

Κρυπτογραφία Ελλειπτικών Καμπυλών

Ειδικά Θέματα: Κρυπτογραφία

Νούλας Δημήτριος
ΑΜ: 7112112100016
dnoulas@math.uoa.gr



ΕΛΛΗΝΙΚΗ ΔΗΜΟΚΡΑΤΙΑ
**Εθνικόν και Καποδιστριακόν
Πανεπιστήμιον Αθηνών**
— ΙΔΡΥΘΕΝ ΤΟ 1837 —

Περιεχόμενα

1	Εισαγωγή	3
2	Κρυπτογραφία Ελλειπτικών Καμπυλών	12
2.1	Πρόβλημα Διακριτού Λογαρίθμου	12
2.2	Πρωτόκολλο ECDH	12
2.3	Κρυπτοσύστημα El Gamal	13
3	Ψηφιακές Υπογραφές	16
3.1	Αλγόριθμος Ψηφιακής Υπογραφής	16
3.2	Πρωτόκολλο ECDSA	17
3.3	Ανάκτηση Δημόσιου Κλειδιού	18
3.4	Κώδικας	18
4	Vulnerabilities	22
4.1	Επίθεση Pohlig-Hellman	22
4.2	Ανώμαλες Καμπύλες	24
4.3	Singular Καμπύλες	25
4.4	Επίθεση MOV	26
4.5	CurveBall	27
5	Ασφαλείς Ελλειπτικές Καμπύλες	28
6	Περαιτέρω μελέτη και Isogenies	32
7	Βιβλιογραφία	37

1 Εισαγωγή

Οι ελλειπτικές καμπύλες οφείλουν το όνομά τους στο πρόβλημα της εύρεσης μήκους τόξου πάνω σε ελλείψεις. Ουσιαστικά, το πρόβλημα αυτό ανάγεται στον υπολογισμό των λεγόμενων ελλειπτικών ολοκληρωμάτων που έχουν την μορφή:

$$\int \frac{1}{\sqrt{x^2 + ax + b}} dx$$

Ο υπολογισμός των παραπάνω ολοκληρωμάτων ήταν ένα από τα κύρια προβλήματα της ανάλυσης στους προηγούμενους αιώνες και διάσημοι μαθηματικοί όπως οι Euler, Weierstrass, Abel, Jacobi ασχολήθηκαν σε αυτόν τον τομέα. Το παραπάνω ολοκλήρωμα οδηγεί στην μελέτη δύο μιγαδικών συναρτήσεων x, y που ικανοποιούν μια εξίσωση της μορφής:

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

ή ισοδύναμα στην μελέτη των μιγαδικών αριθμών $(x, y) \in \mathbb{C}^2$ που ικανοποιούν την παραπάνω εξίσωση. Η σύγχρονη μελέτη αυτού του προβλήματος εντάσσεται στα πλαίσια της μελέτης των επιφανειών Riemann στην γεωμετρία. Στα πλαίσια της κρυπτογραφίας και γενικότερα της επιστήμης των υπολογιστών, αυτές οι εξισώσεις έχουν ιδιαίτερο ενδιαφέρον όταν αντί για το \mathbb{C} , έχοντας κατά νου ότι η μνήμη ενός υπολογιστή είναι πεπερασμένη, τις κοιτάμε πάνω από τα πεπερασμένα σώματα \mathbb{F}_q . Δίνουμε τώρα τον ορισμό μιας ελλειπτικής καμπύλης.

Ορισμός (Ελλειπτική Καμπύλη). Έστω $x^3 + ax + b$ ένα κυβικό πολυώνυμο υπέρ ενός σώματος K το οποίο έχει απλές ρίζες. Μια ελλειπτική καμπύλη υπέρ του σώματος K θα είναι το σύνολο των σημείων $x, y \in K$ που ικανοποιούν την εξίσωση:

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

μαζί με ένα σημείο O στο άπειρο. Η συνθήκη για τις απλές ρίζες είναι ισοδύναμη με την σχέση $\Delta = 16(4a^3 + 27b^2) \neq 0$.

Χωρίς να είναι ακόμα ξεκάθαρο τι είναι το σημείο στο άπειρο, είναι χρήσιμο να βλέπουμε τις ελλειπτικές καμπύλες σε έναν ευρύτερο χώρο, το λεγόμενο προβολικό επίπεδο, το οποίο επιταχύνει τους υπολογισμούς και μπορούμε εύκολα να αναπαριστούμε τα σημεία της ελλειπτικής καμπύλης σαν λίστες $[x : y : z]$. Δίνουμε τον εξή ορισμό:

Ορισμός (Προβολικό Επίπεδο). Έστω K σώμα και στο σύνολο των διατεταγμένων τριάδων $K^3 - (0, 0, 0)$ ορίζουμε την σχέση ισοδυναμίας $(x_1, y_1, z_1) \sim (x_2, y_2, z_2)$ αν και μόνο αν υπάρχει $\lambda \in K^*$ με $(x_1, y_1, z_1) = \lambda(x_2, y_2, z_2)$. Το σύνολο πηλίκο το ονομάζουμε προβολικό επίπεδο και στην βιβλιογραφία συμβολίζεται με

$$\mathbb{P}^2(K) = \frac{K^3 - (0, 0, 0)}{\sim}$$

Ουσιαστικά, βλέπουμε τις ευθείες σαν ένα στοιχείο του χώρου, αφού όλα τα σημεία που βρίσκονται πάνω σε μια ευθεία που περνάει από το $(0, 0, 0)$ είναι ισοδύναμα ως προς την σχέση \sim . Θα συμβολίζουμε την κλάση ισοδυναμίας ενός σημείου $(x, y, z) \neq (0, 0, 0)$ ως $[x : y : z]$. Παρατηρούμε ότι τα σημεία $[x : y : 1]$ είναι σε ένα προς ένα και επί αντιστοιχία με το επίπεδο K^2 , ενώ τα σημεία $[x : y : 0]$ αποτελούν μια ευθεία που την ονομάζουμε ευθεία στο άπειρο. Μπορούμε να χωρίσουμε το προβολικό επίπεδο στα στοιχεία που έχουν αντιπρόσωπο $z = 0$ και σε αυτά που δεν έχουν, δηλαδή στις ευθείες που περνάνε από το $(0, 0, 0)$ στο επίπεδο $\{x, y, 0\}$ και σε αυτές που δεν ζουν μέσα σε αυτό. Από τους ορισμούς, η δεύτερη κατηγορία θα τέμνει το επίπεδο $z = 1$ και άρα κάθε τέτοιο στοιχείο στο προβολικό επίπεδο θα έχει αντιπρόσωπο $[x : y : 1]$. Με την ίδια λογική, οι ευθείες για τις οποίες ισχύει $z = 0$ θα έχουν αντιπρόσωπο

$[x : 1 : 0]$, εκτός από αυτές που είναι παράλληλες στο επίπεδο $\{y = 1, z = 0\}$ που είναι ο άξονας των x , δηλαδή στα σημεία $[x : 0 : 0]$. Συνοπτικά, τα στοιχεία του προβολικού επιπέδου έχουν τρία είδη αντιπροσώπων:

$$[1 : 0 : 0], \quad [x : 1 : 0], \quad [x : y : 0]$$

Για κάθε πολυώνυμο $f(x, y) \in K[x, y]$ βαθμού n :

$$f(x, y) = \sum_{i,j} a_{ij} x^i y^j$$

θα συμβολίζουμε με F το ομογενές πολυώνυμο:

$$F(x, y, z) = \sum_{i,j} a_{ij} x^i y^j z^{n-i-j}$$

και αυτήν την διαδικασία την ονομάζουμε ομογενοποίηση.

Επιστρέφοντας στις ελλειπτικές καμπύλες, το να τις δούμε μέσα στο προβολικό επίπεδο είναι ισοδύναμο με το να ψάχνουμε στοιχεία $[x : y : z]$ που ικανοποιούν την ομογενοποιημένη εξίσωση:

$$E : \quad zy^2 = x^3 + axz^2 + bz^3$$

Βλέπουμε ότι το στοιχείο $[1 : 0 : 0]$ δεν ανήκει πάνω στην καμπύλη, ενώ τα στοιχεία με $z = 1$ μας επιστρέφουν την αρχική εξίσωση $y^2 = x^3 + ax + b$ της ελλειπτικής καμπύλης. Για τα στοιχεία με $z = 0$ έχουμε ότι:

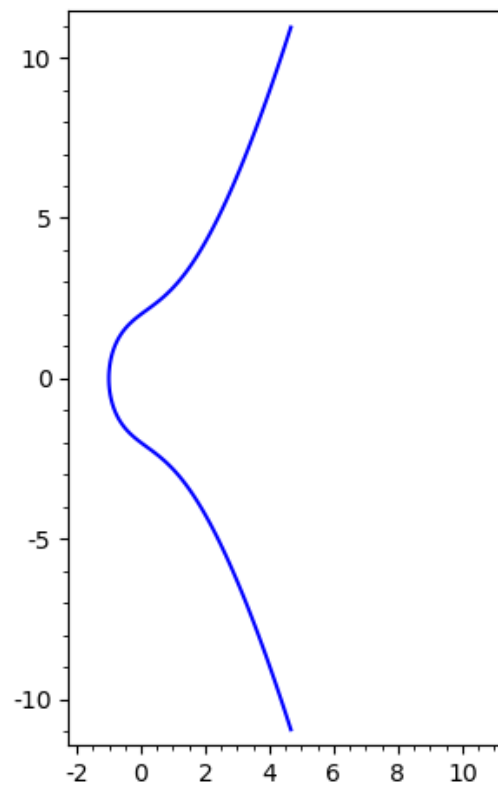
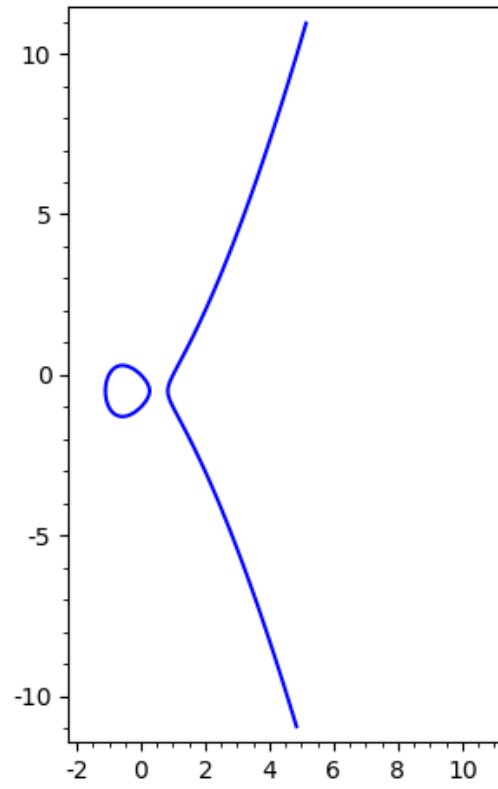
$$z = 0 \implies 0 = y^2 z = x^3 + axz^2 + bz^3 = x^3$$

δηλαδή $x = 0$ και άρα τα σημεία που βρίσκονται πάνω στην ελλειπτική με $z = 0$ είναι αυτά πάνω στον y -άξονα, τα οποία όλα ανήκουν στην κλάση $\mathcal{O} = [0 : 1 : 0]$. Αυτό είναι το στοιχείο που δεν μπορούμε να προσδιορίσουμε με την αρχική εξίσωση της ελλειπτικής καμπύλης, το λεγόμενο σημείο στο άπειρο.

Αξίζει να σημειώσουμε ότι για μια ελλειπτική καμπύλη υπάρχει και η πιο γενική εξίσωση του Weierstrass:

$$y^2 + a_1 xy + a_3 y = x^3 + a_2 x^2 + a_4 x + a_6$$

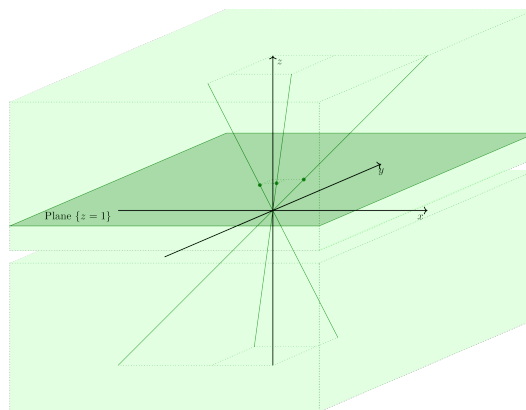
η οποία είναι ισοδύναμη με την αρχική κάνοντας μια γραμμική αλλαγή μεταβλητής. Για να έχουμε μια εικόνα, δύο ελλειπτικές καμπύλες πάνω από τους ρητούς έχουν γραφική παράσταση όπως φαίνεται στην επόμενη σελίδα:



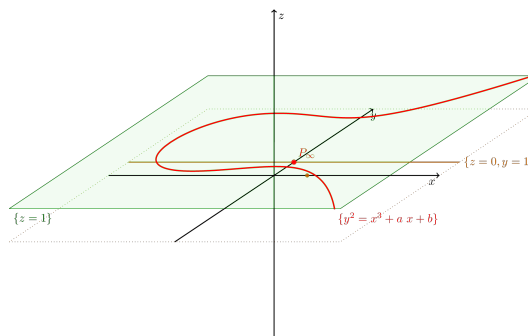
Τις οποίες παράγουμε ως εξής:

```
sage: E1 = EllipticCurve([0,0,1,-1,0])
sage: E1
Elliptic Curve defined by  $y^2 + y = x^3 - x$  over Rational Field
sage: E2 = EllipticCurve([3,4])
sage: E2
Elliptic curve defined by  $y^2 = x^3 + 3x + 4$  over Rational Field
sage: C1 = Curve(E1)
sage: C2 = Curve(E2)
sage: C1.plot()
sage: C2.plot()
```

Ενώ στο προβολικό επίπεδο:



Μια ελλειπτική καμπύλη φαίνεται ως:

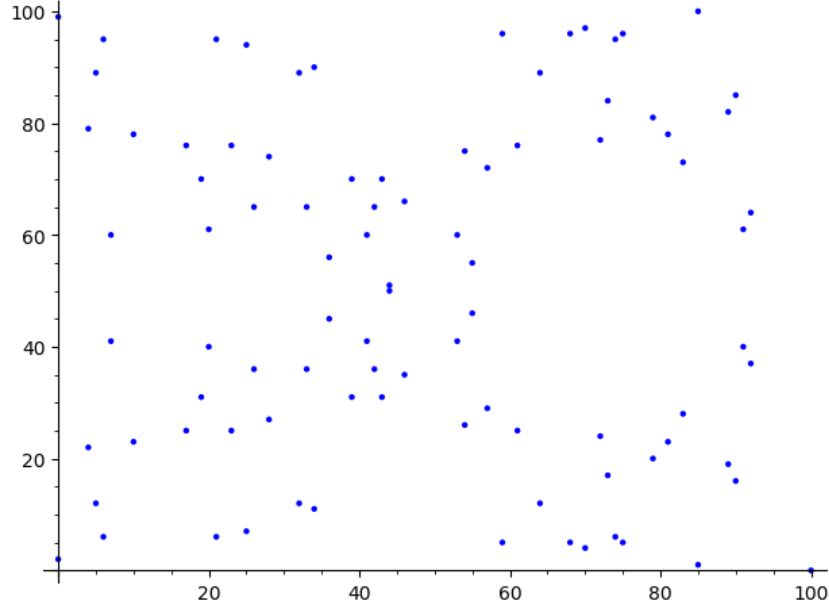


Οι εικόνες είναι από την πηγή: crypto.stackexchange.com/questions/40947/what-is-the-projective-space

Στα πεπερασμένα σώματα που μας ενδιαφέρει στην πράξη, η ελλειπτική καμπύλη έχει γραφική παράσταση σαν την παρακάτω. Εφόσον έχουμε στο σώμα χαρακτηριστική $p > 0$ πρώτο αριθμό, παίρνουμε τα σημεία (x, y) που ικανοποιούν την σχέση:

$$y^2 \equiv x^3 + ax + b \pmod{p}$$

```
sage: E = EllipticCurve(FiniteField(101), [3,4])
sage: P = plot(E, rgbcolor=(0,0,1))
```



Η ελλειπτική καμπύλη $y^2 = x^3 + 3x + 4$ πάνω από το σώμα \mathbb{F}_{101} .

Στην συνέχεια, θα ορίσουμε τον τρόπο πρόσθεσης δύο σημείων πάνω στην ελλειπτική καμπύλη και θα δώσουμε το θεώρημα το οποίο μας επιτρέπει να κάνουμε κρυπτογραφία πάνω σε ελλειπτικές καμπύλες. Έστω $P = (x_1, y_1)$ και $Q = (x_2, y_2)$ δύο σημεία επί της ελλειπτικής καμπύλης. Σχηματίζουμε την ευθεία που ενώνει τα δύο σημεία. Αυτή η ευθεία τέμνει την ευθεία σε ένα τρίτο σημείο PQ . Από το σημείο PQ φέρνουμε την κάθετη στον άξονα των x . Το σημείο τομής της κάθετης με την ελλειπτική καμπύλη το ορίζουμε ως $P + Q$, δηλαδή ως άθροισμα των σημείων P, Q . Στην περίπτωση που θέλουμε να υπολογίσουμε το σημείο $P + P$ θεωρούμε την εφαπτομένη πάνω στο P , αντί για την χορδή όπως στην προηγούμενη περίπτωση. Όταν ένας προσθετέος είναι το \mathcal{O} έχουμε ότι $P + \mathcal{O} = P$, δηλαδή το \mathcal{O} δρα ως ουδέτερο σημείο στην πρόσθεση.

Για να εκφράσουμε αλγεβρικά τους τύπους πρόσθεσης για τα σημεία $P = (x_1, y_1)$ και $Q = (x_2, y_2)$, υποθέτουμε ότι $P, Q \neq \mathcal{O}$ και αν $x_1 = x_2$ και $y_1 = -y_2$ τότε $P + Q = \mathcal{O}$, δηλαδή συμμετρικά σημεία ως προς τον άξονα x έχουν άθροισμα \mathcal{O} . Διαφορετικά θέτουμε:

$$\lambda = \begin{cases} \frac{3x_1 + a}{2y_1}, & \text{αν } P = Q \\ \frac{y_1 - y_2}{x_1 - x_2}, & \text{αν } P \neq Q \end{cases}$$

και βασιζόμενοι στο λ το σημείο $P + Q$ έχει συντεταγμένες:

$$P + Q = (\lambda^2 - x_1 - x_2, \lambda(x_1 - x_2) - y_1)$$

Για να ορίσουμε βαθμωτό πολλαπλασιασμό των σημείων της ελλειπτικής καμπύλης χρησιμοποιούμε τον κανόνα της πρόσθεσης, δηλαδή:

$$n \cdot P = P + P + \dots + P$$

Το θεώρημα στο οποίο βασιζόμαστε είναι το εξής:

Θεώρημα. Τα σημεία της ελλειπτικής καμπύλης με πράξη την πρόσθεση που ορίστηκε και ουδέτερο στοιχείο το \mathcal{O} αποτελούν αβελιανή ομάδα.

Όλα τα παραπάνω μπορούν να υλοποιηθούν απλά με την γλώσσα προγραμματισμού Python ως εξής:

```
from Crypto.Util.number import inverse

def point_add(p, q, E):
    zero = (0, 0)
    if p == zero:
        return q
    elif q == zero:
        return p
    else:
        x1, y1 = p
        x2, y2 = q
        if x1 == x2 and y1 == -y2:
            return zero

        Ea, Ep = E['a'], E['p']
        if p != q:
            lmd = (y2 - y1) * inverse(x2 - x1, Ep)
        else:
            lmd = (3 * (x1**2) + Ea) * inverse(2 * y1, Ep)
        x3 = ((lmd**2) - x1 - x2) % Ep
        y3 = (lmd * (x1 - x3) - y1) % Ep
        return x3, y3

def scalar_mult(n, p, E):
    q, r = p, (0, 0)
    while n > 0:
        if n % 2 == 1:
            r = point_add(r, q, E)
        q = point_add(q, q, E)
        n //= 2
    return r
```

Ωστόσο, δεν χρειάζεται κανείς να ξεκινήσει από το μηδέν. Για τους μαθηματικούς είναι αρκετά χρήσιμο το πακέτο SageMath βασιζόμενο στην γλώσσα προγραμματισμού Python που κάνει τους υπολογισμούς στο προβολικό επίπεδο:


```

sage: E = EllipticCurve(FiniteField(101), [3,4])
sage: P = E.point([0,2])
sage: P
(0:2:1)
sage: P.order()
23
sage: for i in range(1,P.order()+1):
....:   i*P
....:
(0 : 2 : 1)
(70 : 97 : 1)
(55 : 46 : 1)
(83 : 73 : 1)
(32 : 12 : 1)
(4 : 22 : 1)
(21 : 95 : 1)
(79 : 81 : 1)
(23 : 76 : 1)
(20 : 61 : 1)
(44 : 50 : 1)
(44 : 51 : 1)
(20 : 40 : 1)
(23 : 25 : 1)
(79 : 20 : 1)
(21 : 6 : 1)
(4 : 79 : 1)
(32 : 89 : 1)
(83 : 28 : 1)
(55 : 55 : 1)
(70 : 4 : 1)
(0 : 99 : 1)
(0 : 1 : 0)

```

Επιπλέον, υπάρχει και η βιβλιοθήκη PyCryptodome της Python που υποστηρίζει ολόκληρο το Public Key Infrastructure και στο επόμενο παράδειγμα φαίνεται πώς μπορούμε να επιλέξουμε μια ελλειπτική καμπύλη, να δημιουργήσουμε ιδιωτικό και δημόσιο κλειδί και να τα χειριστούμε σε αρχεία:

```

from Crypto.PublicKey import ECC

private_key = ECC.generate(curve='P-256')
public_key = private_key.public_key()
print(public_key)
%EccKey(curve='NIST_P-256', point_x=7656170542301456206
1860150560615631451455188074817536478296461957767192909
320,
point_y=69089698564456934829353568825198745013281366555
012989239449323649065476753678)

f = open('myprivatekey.pem', 'wt')
f.write(key.export_key(format='PEM'))
f.close()

f = open('myprivatekey.pem', 'rt')

```

```
key = ECC.import_key(f.read())
```

Πριν κλείσει το κεφάλαιο αξίζει να αναφερθούμε σε κάποια κομμάτια θεωρίας. Για μια δεδομένη ελλειπτική καμπύλη E θα συμβολίζουμε με $E(K)$ την αβελιανή ομάδα των σημείων της υπεράνω του σώματος K . Ας δούμε τι γίνεται με τις τάξεις των σημείων αυτής της ομάδας.

Θεωρούμε λοιπόν μια ελλειπτική καμπύλη $E : y^2 = x^3 + ax + b$ ορισμένη στο σώμα K . Τα σημεία $(x, 0)$ ανήκουν στην E αν και μόνο αν το x είναι ρίζα του πολυωνύμου $x^3 + ax + b$. Υπάρχει η περίπτωση το K να μην περιέχει τις τρεις ρίζες του πολυωνύμου, αλλά ξέρουμε από την θεωρία σωμάτων ότι υπάρχει κατάλληλη επέκταση σώματος $L \supseteq K$ που θα τις περιέχει και βλέπουμε πλέον την ελλειπτική καμπύλη πάνω από το L . Άρα υπάρχουν τρία σημεία της ελλειπτικής $(x_1, 0), (x_2, 0), (x_3, 0)$ που μαζί με το \mathcal{O} σχηματίζουν ομάδα ισόμορφη με την $\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2\mathbb{Z}$. Αυτό προκύπτει εφόσον αυτά είναι τα μόνα σημεία P της ελλειπτικής για τα οποία $2P = \mathcal{O}$, δηλαδή $P + P = \mathcal{O}$. Ένας άλλος τρόπος που μεταφράζεται αυτό είναι ότι για την y -συντεταγμένη του P έχουμε $y = -y$ και άρα ισχύει $y = 0$.

Γενικότερα, αν επιλέξουμε ένα στοιχείο P στην E , μπορούμε να θεωρήσουμε όλα τα πολλαπλάσια nP για κάθε φυσικό αριθμό n . Η μια περίπτωση είναι να έχουμε $nP = \mathcal{O}$ για κάποιο φυσικό αριθμό n , οπότε το σημείο P θα έχει πεπερασμένη τάξη. Είναι όπως είδαμε με το sage ότι για το σημείο $P = (0, 2)$ στην ελλειπτική καμπύλη $E : y^2 = x^3 + 4x + 4$ πάνω από το σώμα \mathbb{F}_{101} ισχύει ότι $23P = \mathcal{O}$.

Διαφορετικά, ένα σημείο μπορεί να έχει άπειρη τάξη και να παράγει υποομάδα της ελλειπτικής καμπύλης ισόμορφη με την άπειρη κυκλική ομάδα \mathbb{Z} , το οποίο συμβαίνει φυσικά μόνο στα άπειρα σώματα.

Στα πεπερασμένα σώματα \mathbb{F}_{p^h} είναι σαφές ότι η ομάδα $E(\mathbb{F}_{p^h})$ θα είναι πεπερασμένη αβελιανή και ένα χαλαρό φράγμα της τάξης της είναι το $p^{2h} + 1$, αφού έχουμε το πολύ p^h επιλογές για τα y και για τα x . Ένα καλύτερο φράγμα με απόδειξη του H. Hasse είναι το $p^h + 1 \pm s$ με $|s| \leq 2\sqrt{p^h}$, όπου το s στην βιβλιογραφία είναι γνωστό ως το ίχνος του Frobenius.

Δύο ακόμα γνωστά σημαντικά θεωρήματα είναι τα εξής:

Θεώρημα (Mordell). Για μια ελλειπτική καμπύλη E , η αβελιανή ομάδα $E(\mathbb{Q})$ είναι πεπερασμένα παραγόμενη, δηλαδή:

$$E(\mathbb{Q}) = \mathbb{Z}^r \times \prod_{i=1}^s \mathbb{Z}/n_i\mathbb{Z}$$

Στην πραγματικότητα μπορούμε να είμαστε περισσότερο ακριβείς για το κομμάτι της πεπερασμένης τάξης του $E(\mathbb{Q})$ αφού ισχύει και το θεώρημα του Barry Mazur:

Θεώρημα (Mazur). Η ομάδα των σημείων πεπερασμένης τάξης σε μια ελλειπτική καμπύλη είναι ισόμορφη με μια από τις παρακάτω 15 ομάδες:

$$\mathbb{Z}/n\mathbb{Z}, \quad 1 \leq n \leq 10 \text{ ή } n = 12$$

$$\mathbb{Z}/2\mathbb{Z} \times \mathbb{Z}/2n\mathbb{Z}, \quad 1 \leq n \leq 4$$

Επιπλέον, είναι αξιοσημείωτη η συνεισφορά των ελλειπτικών καμπυλών στην θεωρία αριθμών, αφού αποτέλεσαν θεμέλιο για το διάσημο πρόβλημα γνωστό ως το τελευταίο θεώρημα του Fermat:

$$x^n + y^n = z^n, \quad n \geq 3 \quad \text{δεν υπάρχουν λύσεις } (x, y, z) \in \mathbb{Z}^3 \text{ με } xyz \neq 0$$

Η κεντρική ιδέα ήταν ότι για έναν πρώτο αριθμό $p \geq 3$ αν είχαμε μια μη τετριμμένη λύση της εξίσωσης:

$$a^p + b^p = c^p$$

τότε θα παίρναμε την αντίστοιχη ελλειπτική καμπύλη:

$$E : y^2 = x(x - a^p)(x - b^p)$$

της οποίας οι ιδιότητες θα δημιουργούσαν αντίφαση με την εικασία Taniyama-Shimura. Το μεγαλύτερο μέρος της εικασίας, που αρκούσε για αυτήν την ιδέα, αποδείχθηκε από τον Sir Andrew Wiles το 1995, ενώ το υπόλοιπο μέρος της εικασίας αποδείχθηκε το 1999.

2 Κρυπτογραφία Ελλειπτικών Καμπυλών

Η κρυπτογραφία ελλειπτικών καμπυλών είναι ένας μοντέρνος και εύχρηστος τρόπος κρυπτογραφίας ανοιχτού κλειδιού και η ασφάλειά του βασίζεται στην δυσκολία που έχει η επίλυση του προβλήματος του διακριτού λογαρίθμου. Το πλεονέκτημα της κρυπτογραφίας ελλειπτικών καμπυλών σε σχέση με άλλους ασύμμετρους αλγόριθμους όπως τον RSA είναι το σημαντικά μικρότερο μέγεθος κλειδιού, ενώ ταυτόχρονα παρέχοντας ίδιο μέγεθος ασφάλειας. Για παράδειγμα, ένα 3072-bit κλειδί RSA αναλογεί σε 768 bytes ενώ το εξίσου ασφαλές ιδιωτικό κλειδί της ελλειπτικής καμπύλης NIST P-256 αναλογεί σε 32 bytes.

2.1 Πρόβλημα Διακριτού Λογαρίθμου

Δεδομένου μιας ομάδας G και ενός στοιχείου της g με το στοιχείο υψωμένο σε μια δύναμη g^x , το πρόβλημα του διακριτού λογαρίθμου είναι το ακόλουθο: Αν γνωρίζουμε τα (G, g, g^x) να υπολογίσουμε το x . Η ασφάλεια της κρυπτογραφίας με αυτό το πρόβλημα είναι ότι δεν γνωρίζουμε να υπάρχει εύχρηστος αλγόριθμος να υπολογίζει το x , ωστόσο χρειάζεται προσοχή στην επιλογή της ομάδας G και του γεννήτορα g .

Συγκεκριμένα, στις ελλειπτικές καμπύλες πάνω από ένα πεπερασμένο σώμα \mathbb{F}_q η ομάδα $E(\mathbb{F}_q)$ είτε θα είναι κυκλική, αλλιώς διαλέγουμε την κυκλική υποομάδα που παράγει ένα στοιχείο με τάξη πρώτο αριθμό και το πρόβλημα διακριτού λογαρίθμου ελλειπτικών καμπυλών (ECDLP) είναι:

$$(ECDLP) : \text{Δεδομένου γεννήτορα } G \in E(\mathbb{F}_q) \quad \text{και} \quad A = nG \quad \text{βρες το } n$$

Η πρακτική ένδειξη της δυσκολίας του προβλήματος στις ελλειπτικές καμπύλες είναι ότι τα κρυπτοσυστήματα όπως το El Gamal είναι δυσκολότερο να λυθούν από τα αντίστοιχα συστήματα αν τα δούμε στις ομάδες $G = \mathbb{F}_q^*$.

2.2 Πρωτόκολλο ECDH

Ένας από τους τρόπους που χρησιμοποιείται η κρυπτογραφία ελλειπτικών καμπυλών είναι με το πρωτόκολλο ανταλλαγής κλειδιού Elliptic Curve Diffie-Hellman (ECDH) ακολουθώντας το κλασικό πρωτόκολλο ανταλλαγής κλειδιού Diffie-Hellman όπως φαίνεται παρακάτω. Η Alice και ο Bob έχοντας συμφωνήσει στην καμπύλη μαζί με το πεπερασμένο σώμα, τον πρώτο αριθμό p και τον γεννήτορα G τα οποία είναι οι δημόσιοι παράμετροι, διαλέγουν τα ιδιωτικά κλειδιά τους n_A, n_B και υπολογίζουν τα δημόσια κλειδιά που ανταλλάζουν ως εξής:

ECDH

Alice

$$n_A \in \{1, 2, \dots, p-1\}$$

$$A = n_A G$$

Bob

$$n_B \in \{1, 2, \dots, p-1\}$$

$$B = n_B G$$

στέλνει η Alice το A

→

στέλνει ο Bob το B

←

$$S = n_A B$$

$$S = n_B A$$

κοινό κλειδί:

$$S = n_A n_B G = n_A B = n_B A$$

Ουσιαστικά, κάποιος που θα έχει τα δημόσια κλειδιά A, B δεν θα μπορεί να υπολογίσει σε ανθρώπινο χρόνο το S . Οι Alice και Bob έχοντας το $S = (x, y)$ κρατάνε ως κλειδί την πρώτη συντεταγμένη x . Αργότερα, χρησιμοποιούν το x σαν κλειδί σε κάποιον συμμετρικό αλγόριθμο κρυπτογράφησης όπως τον AES. Σαφώς, όπως το κλασικό πρωτόκολλο DH το ECDH δεν είναι ασφαλές από μόνο του σε man in the middle επιθέσεις και χρειάζονται επιπλέον μέτρα ασφαλείας.

2.3 Κρυπτόςυστημα El Gamal

Το κρυπτόςυστημα El Gamal είναι ένας ασύμμετρος αλγόριθμος κρυπτογράφησης ανοιχτού κλειδιού που βασίζεται στο πρωτόκολλο ανταλλαγής κλειδιού Diffie-Hellman. Μπορεί να οριστεί γενικά για οποιαδήποτε κυκλική ομάδα G και η ασφάλειά του βασίζεται ομοίως στην δυσκολία της επίλυσης του προβλήματος διακριτού λογαρίθμου στην G . Η διαφορά με το Diffie-Hellman είναι ότι ο Bob κρυπτογραφεί μήνυμα μέσα στο πρωτόκολλο, το οποίο στέλνει στην Alice στην παραπάνω ανταλλαγή και δεν της στέλνει κάποιο δημόσιο κλειδί του. Έχοντας μια κυκλική ομάδα G τάξης q με γεννήτορα g , διαγραμματικά το πρωτόκολλο λειτουργεί ως εξής:

El Gamal

Alice

Υπολόγισε τυχαία

$x \in \{1, \dots, q-1\}$

x ιδιωτικό κλειδί

Υπολόγισε $h := g^x$

Δημόσιο κλειδί: (G, q, g, h)

Bob

Μήνυμα M

Αντιστοίχισε M με στοιχείο m της G
με απεικόνιση που αντιστρέφεται

Υπολόγισε τυχαία

$y \in \{1, \dots, q-1\}$

Υπολόγισε

$s := h^y$

s κοινό μυστικό

Υπολόγισε τα

$c_1 := g^y$

$c_2 := m \cdot s$

(c_1, c_2) κρυπτογραφημένο μήνυμα

στέλνει ο Bob το (c_1, c_2)

Αποκρυπτογράφηση

$s = c_1^x$

το κοινό μυστικό

Υπολόγισε s^{-1}

ως $s^{-1} = c_1^{q-x}$

Υπολόγισε m

ως $m = c_2 \cdot s^{-1}$

Αντιστοίχισε πίσω στο M

Σαφώς, όλο το παραπάνω δουλεύει καλά σε μια ελλειπτική καμπύλη, όπου η κυκλική ομάδα ορίζεται από ένα σημείο P της καμπύλης και οι δυνάμεις g^x είναι τα πολλαπλάσια xP . Ας δούμε ένα παράδειγμα που δουλεύει αυτό το κρυπτοσύστημα με ελλειπτικές καμπύλες, καθώς και σε τι άλλο κομμάτι έξω από το κρυπτοσύστημα υπήρχε κενό ασφάλειας ώστε αυτό να σπάσει. Το παρακάτω παράδειγμα είναι ένα σύστημα Digital Rights Management (DRM) της Microsoft που έσπασε ο hacker Beale Screamer. Ακολουθούμε την περιγραφή του William Stein [2].

Πεπερασμένο Σώμα $K = \mathbb{Z}/p\mathbb{Z}$

Πρώτος $p = 85963102379428822376694789446897396207498568951$

και στο δεκαεξαδικό σύστημα:

$p = 89ABCDEF012345672718281831415926141424F7$

Ορίζουμε την ελλειπτική καμπύλη:

$$y^2 = x^3 + 317689081251325503476317476413827693272746955927x + 79052896607878758718120572025718535432100651934$$

για την οποία ισχύει:

$$|E(K)| = 785963102379428822376693024881714957612686157429$$

η οποία είναι κυκλική με γεννήτορα:

$$B = (771507216262649826170648268565579889907769254176, 390157510246556628525279459266514995562533196655)$$

Η Alice και ο Bob θέλουν να μοιραστούν την μουσική τους και μόλις η Alice εγκατέστησε το λογισμικό DRM αυτό παρήγαγε στον υπολογιστή της το ιδιωτικό της κλειδί:

$$n = 670805031139910513517527207693060456300217054473$$

Για να παίξει η Alice λοιπόν το song.wma αφού κάνει την αγορά online, η ιστοσελίδα της συναλλαγής της κατεβάζει στον υπολογιστή της ένα αρχείο άδειας license file που την αφήνει να αναπαράγει το song.wma. Αυτό το αρχείο παράχθηκε από το κρυπτοσύστημα El Gamal στην ομάδα $E(K)$. Η Alice τώρα μοιράζεται με τον Bob το τραγούδι song.wma αλλά και το αρχείο άδειας που έλαβε, ωστόσο ο Bob δεν μπορεί να αναπαράγει το τραγούδι καθώς δεν έχει το ιδιωτικό κλειδί n της Alice που είναι αποθηκευμένο στον υπολογιστή της. Άρα δεν μπορεί να αποκρυπτογραφήσει ο υπολογιστής του το αρχείο άδειας και για αυτό δεν μπορεί να αναπαράγει το τραγούδι.

Στην πραγματικότητα, η Alice δημοσίευσε το δημόσιο κλειδί $(E(K), p, B, nB)$ και το αρχείο άδειας που της έστειλε η Microsoft ήταν ένα σημείο $Q = (x, y)$ στην ελλειπτική καμπύλη. Αυτό το x είναι το κλειδί που χρειάζεται για να παίξει το song.wma. Η Microsoft διαλέγει έναν πρώτο αριθμό r και στέλνει στην Alice τα σημεία της ελλειπτικής καμπύλης:

$$C_1 := rB$$

$$C_2 := Q + rnB$$

και ο υπολογιστής της Alice διαβάζει το n που έχει αποθηκευμένο και υπολογίζει το $nC_1 = n(rB) = nrB$. Έτσι έχει όσα χρειάζεται για να υπολογίσει:

$$Q = C_2 - nrB = Q + nrB - nrB$$

Στο παράδειγμά μας, η Microsoft έστειλε τα σημεία:

$$rB = (179671003218315746385026655733086044982194424660, \\ 697834385359686368249301282675141830935176314718)$$

και

$$Q + r(nB) = (137851038548264467372645158093004000343639118915, \\ 110848589228676224057229230223580815024224875699)$$

και όταν η Alice επιχειρεί να αναπαράγει το song.wma φορτώνεται στην μνήμη το

$$n = 670805031139910513517527207693060456300217054473$$

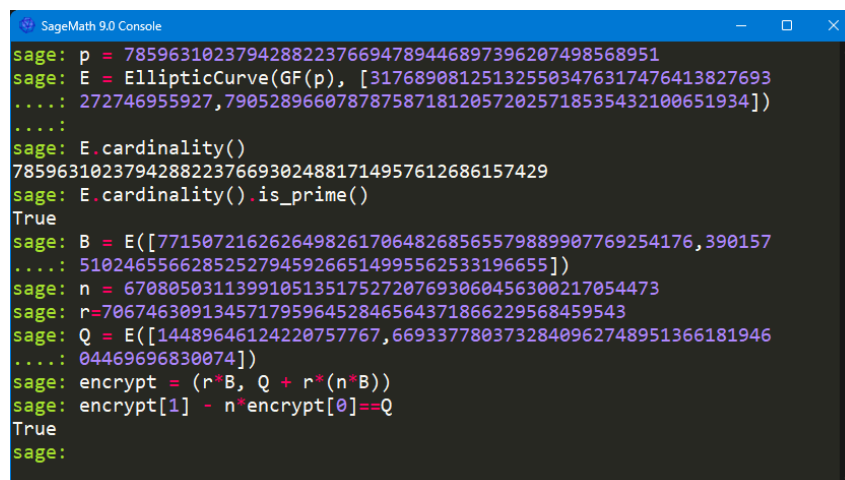
και υπολογίζεται το

$$n(rB) = (328901393518732637577115650601768681044040715701, \\ 586947838087815993601350565488788846203887988162)$$

το οποίο αφαιρεί από το $Q + r(nB)$ για να πάρει:

$$Q = (14489646124220757767, 669337780373284096274895136618194604469696830074)$$

και η x -συντεταγμένη 14489646124220757767 είναι το κλειδί που επιτρέπει στον υπολογιστή να παίξει το song.wma. Ας επαληθεύσουμε στο sage τα παραπάνω:



```
sage: p = 785963102379428822376694789446897396207498568951
sage: E = EllipticCurve(GF(p), [317689081251325503476317476413827693
....: 272746955927, 79052896607878758718120572025718535432100651934])
....:
sage: E.cardinality()
785963102379428822376693024881714957612686157429
sage: E.cardinality().is_prime()
True
sage: B = E([771507216262649826170648268565579889907769254176, 390157
....: 510246556628525279459266514995562533196655])
sage: n = 670805031139910513517527207693060456300217054473
sage: r=70674630913457179596452846564371866229568459543
sage: Q = E([14489646124220757767, 6693377803732840962748951366181946
....: 04469696830074])
sage: encrypt = (r*B, Q + r*(n*B))
sage: encrypt[1] - n*encrypt[0]==Q
True
sage:
```

Αν η Alice ήξερε η ίδια το n θα μπορούσε να υπολογίσει το Q μόνη της και έτσι να μοιραστεί την μουσική με τον Bob. Αυτό το κρυπτοσύστημα έσπασε, όχι γιατί δεν ήταν ασφαλές από μαθηματικής άποψης, αλλά γιατί είχε φτωχή υλοποίηση καθώς η ιδέα του ήταν να αποθηκεύει στον υπολογιστή του χρήστη το ιδιωτικό κλειδί. Περισσότερα για το συγκεκριμένο κενό ασφαλείας μπορεί να διαβάσει κανείς στο cryptome.org/ms-drm.htm.

3 Ψηφιακές Υπογραφές

Οι ψηφιακές υπογραφές χρησιμοποιούν τα ίδια κρυπτοσυστήματα για την κρυπτογράφηση μηνυμάτων αλλά με διαφορετικό σκοπό. Δεν είναι το κίνητρο να περαστεί ασφαλώς το μήνυμα στον παραλήπτη, αλλά όπως το λέει η λέξη υπογραφή, είναι για να διασφαλιστούν τα ακόλουθα τρία πράγματα. Αρχικά είναι για την αυθεντικότητα των μηνυμάτων, να υπάρχει εξασφάλιση ότι βρίσκεται πράγματι το ίδιο άτομο που αναγράφεται πίσω από τα μηνύματα. Ο δεύτερος λόγος είναι μια απόδειξη ότι το αρχικό μήνυμα δεν έχει αλλοιωθεί και τελευταίος λόγος είναι ότι αυτός που υπέγραψε το μήνυμα δεν μπορεί στο μέλλον να το αρνηθεί.

Το πρωτόκολλο λειτουργεί ως εξής, έχοντας συμφωνήσει σε ένα ασύμμετρο αλγόριθμο κρυπτογράφησης ο Bob θέλει η Alice να του υπογράψει ένα μήνυμα m . Στέλνει το m στην Alice και εκείνη μπλέκει το μήνυμα με το ιδιωτικό της κλειδί και του επιστρέφει το μήνυμα. Ο Bob με την σειρά του μπλέκει το μήνυμα που έλαβε με το δημόσιο κλειδί της Alice και του επιστρέφεται το αρχικό μήνυμα m . Φυσικά αυτό μπορεί να το κάνει οποιοσδήποτε κρυφακούει την συνομιλία. Δηλαδή, ακόμα και κάποιος που κρυφάκουσε μπορεί να επαληθεύσει ότι το μήνυμα το υπέγραψε η ίδια η Alice.

3.1 Αλγόριθμος Ψηφιακής Υπογραφής

Ο αλγόριθμος ψηφιακής υπογραφής γνωστός ως DSA χρησιμοποιείται για υπογραφή και επαλήθευση υπογραφής. Ομοίως, η ασφάλεια του βασίζεται στην δυσκολία επίλυσης του προβλήματος διακριτού λογαρίθμου. Σαν αλγόριθμος, είναι μια παραλλαγή του κρυπτοσυστήματος El Gamal. Λειτουργεί ως εξής:

- Η χρήση του DSA για υπογραφή αρχικά υπολογίζει το hash του μηνύματος και παράγει έναν τυχαίο ακέραιο k . Με αυτά τα δύο υπολογίζει την υπογραφή $\{r, s\}$. Τα r, s είναι ακέραιοι και το r εξαρτάται μόνο από το k , ενώ το s υπολογίζεται από το hash του μηνύματος, το ιδιωτικό κλειδί του ασύμμετρου αλγορίθμου που χρησιμοποιούμε και τον τυχαίο αριθμό k . Εφόσον έχουμε τυχαιότητα με το k αυτή η διαδικασία υπογραφής θεωρείται μη-ντετερμινιστική.
- Για την επαλήθευση της υπογραφής γίνονται υπολογισμοί, δεχόμενοι σαν είσοδο το hash του μηνύματος, το δημόσιο κλειδί του ασύμμετρου αλγορίθμου καθώς και της ίδιας της υπογραφής $\{r, s\}$.

Ο τυχαίος αριθμός k (που παράγεται όταν δημιουργείται η υπογραφή) αφήνει ένα κενό ασφάλειας. Αν δύο διαφορετικά μηνύματα υπογραφούν με το ίδιο ιδιωτικό κλειδί και το ίδιο k τότε ένας μπορεί να επιτεθεί στο πρωτόκολλο και να βρει το ιδιωτικό κλειδί του υπογραφέα. (Δες github.com/tintinweb/ecdsa-private-key-recovery).

Υπάρχει και ο περισσότερο ασφαλής ντετερμινιστικός DSA, που δεν υπολογίζει το k τυχαία αλλά σαν hash message authentication code (HMAC) που εξαρτάται από το ιδιωτικό κλειδί, το hash του μηνύματος που αποστέλλεται και άλλες παραμέτρους. Όπως θα δούμε, χρησιμοποιούνται οι ελλειπτικές καμπύλες με τέτοιο αλγόριθμο, βασιζόμενοι στην δυσκολία του ECDLP καθώς και το μικρότερο μέγεθος σε μνήμη που έχουν τα κλειδιά και οι υπογραφές, ενώ ταυτόχρονα παρέχεται το ίδιο επίπεδο ασφάλειας και καλύτερη επίδοση.

Επιπλέον, το δημόσιο κλειδί (x, y) της ελλειπτικής μπορεί πάντα να συμπτυκνωθεί στο x . Αν για παράδειγμα έχουμε την καμπύλη secp256k1 που έχει ιδιωτικά κλειδιά με 256-bit ακεραίους (32 bytes) τότε γνωρίζοντας το x , το $x^3 + ax + b := m$ είναι γνωστό και έχουμε να υπολογίσουμε το τετραγωνικό υπόλοιπο:

$$y^2 = m \mod p$$

οπότε απλά κρατάμε ένα επιπλέον bit για το πρόσθετο μέσα στο τετράγωνο. Έτσι, έχουμε δημόσιο κλειδί με 257-bit.

3.2 Πρωτόκολλο ECDSA

Ο αλγόριθμος ψηφιακής υπογραφής ελλειπτικών καμπυλών (ECDSA) είναι μια παραλλαγή του κλασικού DSA και ομοίως στηρίζεται στο κρυπτοσύστημα El Gamal. Για τον αλγόριθμο έχουμε ως είσοδο (E, n, G, dG) , όπου E είναι μια ελλειπτική καμπύλη πάνω από κάποιο πεπερασμένο σώμα \mathbb{F}_q , n είναι η τάξη της ομάδας που παράγει το $G \in E$ και είναι πρώτος αριθμός και $d \in \{2, \dots, n-1\}$ το ιδιωτικό κλειδί το οποίο με την σειρά του δίνει το δημόσιο κλειδί dG .

- **Υπογραφή:**

- (1) Υπολογίζουμε το hash του μηνύματος m , με κάποιον hash αλγόριθμο όπως ο SHA-256, παίρνουμε $h = \text{hash}(m)$
- (2) Παράγουμε τυχαία έναν ακέραιο αριθμό $k < n$, ή στην ντετερμινιστική περίπτωση παίρνουμε το HMAC των h, d .
- (3) Υπολογίζουμε το σημείο στην καμπύλη $R = kG$ και παίρνουμε την x συντεταγμένη του ως $r = R.x$
- (4) Υπολογίζουμε το $s = k^{-1}(h + rd) \pmod n$, όπου k^{-1} είναι το αντίστροφο του k στο σώμα $\mathbb{Z}/n\mathbb{Z}$, δηλαδή $kk^{-1} = 1 \pmod n$.
- (5) Επιστρέφουμε την υπογραφή $\{r, s\}$.

- **Επαλήθευση Υπογραφής:** Παίρνουμε σαν είσοδο το μήνυμα m που στάλθηκε για υπογραφή, την υπογραφή $\{r, s\}$ και το δημόσιο κλειδί dG του υπογραφέα που αντιστοιχεί στο ιδιωτικό κλειδί d . Κατόπιν, εκτελούμε τα ακόλουθα:

- (1) Υπολογίζουμε το hash του μηνύματος m , με τον ίδιο αλγόριθμο με πριν και έχουμε $h = \text{hash}(m)$.
- (2) Υπολογίζουμε το $s^{-1} \in \mathbb{Z}/n\mathbb{Z}$.
- (3) Υπολογίζουμε το σημείο στην ελλειπτική $R' = (hs^{-1}) \cdot G + (rs^{-1}) \cdot dG = s^{-1}(h + rd) \cdot G$.
- (4) Παίρνουμε ως r' την x -συντεταγμένη $r' = R'.x$.
- (5) Επαληθεύουμε αν $r = r'$.

Ουσιαστικά, υπογράφοντας κωδικοποιείται ένα στοιχείο R της ελλειπτικής καμπύλης (που αναπαρίσταται μόνο με την x -συντεταγμένη) ως r και με πράξεις στην ελλειπτική καμπύλη χρησιμοποιώντας τα d, h παίρνουμε το s που είναι το αποδεικτικό στοιχείο ότι αυτός που υπογράφει το μήνυμα γνωρίζει το ιδιωτικό κλειδί d . Από την υπογραφή $\{r, s\}$ και μόνο είναι υπολογιστικά δύσκολο να βρεθεί το d λόγω της δυσκολίας του ECDLP.

Για την επαλήθευση, αποδικοποιείται το s πίσω στο αρχικό σημείο R της ελλειπτικής καμπύλης χρησιμοποιώντας το δημόσιο κλειδί dG και το hash h και γίνεται η σύγκριση της x -συντεταγμένης με το r της υπογραφής.

Για του λόγου το αληθές, κάνουμε τις πράξεις για την ορθότητα της διαδικασίας. Έχουμε:

$$R' = (hs^{-1}) \cdot G + (r \cdot s^{-1}) \cdot dG = s^{-1}(h + rd) \cdot G$$

και το s κατά την διαδικασία της υπογραφής το έχουμε υπολογίσει ως

$$s = k^{-1}(h + rd) \pmod n$$

άρα

$$s^{-1} = (k^{-1}(h + rd))^{-1} = k(h + rd)^{-1} \pmod n$$

και μπορούμε να κάνουμε την αντικατάσταση $s^{-1} \leftrightarrow k(h + rd)^{-1}$ στο R' , τα αντίστροφα είναι απλά αθέμιτοι $\mod n$ και εφόσον τα $s^{-1}, k(h + rd)^{-1}$ είναι ισοδύναμα $\mod n$ τότε θα δίνουν και το ίδιο σημείο στην καμπύλη ως συντελεστές του G , αφού η ομάδα στην οποία δουλεύουμε είναι η $\{O, G, 2G, \dots, (n-1)G\}$ που παράγεται από το G με τάξη n . Άρα πράγματι μπορούμε να κάνουμε την αντικατάσταση:

$$R' = s^{-1}(h + rd) \cdot G = k(h + rd)^{-1}(h + rd) \cdot G = kG = R$$

και άρα πράγματι ο υπογραφέας είναι αυτός που υποστηρίζει αν $r = r'$ που σημαίνει ότι επαληθεύεται η σχέση $R = R'$ και δεν έχει συμβεί κάποιο σφάλμα.

3.3 Ανάκτηση Δημόσιου Κλειδιού

Είναι αρκετά σημαντικό ότι με το πρωτόκολλο ECDSA μπορεί το δημόσιο κλειδί να ανακτηθεί από το μήνυμα m μαζί με την υπογραφή $\{r, s\}$. Αυτό επιτρέπει σε συστήματα να μην το μεταφέρουν, μειώνοντας έτσι το μέγεθος της πληροφορίας που ανταλλάσσεται και αυτό είναι ιδιαίτερα χρήσιμο εκεί που η χρήση πόρων είναι αρκετά αυστηρή και δεν θα θέλαμε να κρατάμε επιπλέον πληροφορία. Αυτό συμβαίνει ακόμα και στα blockchain συστήματα και έτσι τα δημόσια κλειδιά δεν αποθηκεύονται και ανακτούνται την στιγμή της επαλήθευσης. Ωστόσο, η διαδικασία της ανάκτησης με είσοδο $\{r, s\}$ επιτρέπει σε 0, 1 ή 2 πιθανά δημόσια κλειδιά. Για να αποφεύγεται αυτό γίνεται μια επέκταση του ECDSA που επιστρέφει σαν υπογραφή $\{r, s, v\}$ ένα επιπλέον bit v με το οποίο ανακτάται το δημόσιο κλειδί με σιγουριά. Με αυτόν τον τρόπο, το Ethereum για παράδειγμα χρησιμοποιεί υπογραφές $\{r, s, v\}$ για τις υπογεγραμμένες συναλλαγές επιβαρύνοντας λιγότερο το δίκτυο και δεσμεύοντας λιγότερο αποθηκευτικό χώρο.

Έχοντας ως παραμέτρους τα (E, G, n, k) με E την ελλειπτική καμπύλη, G το στοιχείο που διαλέγουμε σαν γεννήτορα τάξης n και $k = |E|/n$ τότε για ένα μήνυμα m με hash h και υπογραφή $\{r, s\}$ η ανάκτηση γίνεται ως εξής:

(1) Για $j = 0$ έως k :

- i) Θέτουμε $x = r + jn$.
- ii) Μετατροπή του ακεραίου x σε octet string X όπως περιγράφεται στο SEC1 [10].
- iii) Μετατροπή του octet string $02_{16}||X$ σε σημείο R της ελλειπτικής καμπύλης, όμοια όπως στο [10]. Αν αποτυγχάνει αυτό γύρνα στο βήμα (1).
- iv) Αν $nR \neq O$ τότε γύρνα στο βήμα (1).
- v) Πιθανό δημόσιο κλειδί:

$$Q = r^{-1}(sR - hG)$$

δοκιμή της επαλήθευσης αν μας επιστρέφει το μήνυμα m , αν ναι τέλος του αλγορίθμου και εκτυπώνεται το Q . Αν όχι, δοκιμάζουμε το ίδιο με όπου R το $-R$.

(2) Αν τελειώσει η επανάληψη Για χωρίς να τελειώσει ο αλγόριθμος, εκτύπωσε ότι δεν υπάρχει δημόσιο κλειδί.

3.4 Κώδικας

Θα βασιστούμε στην βιβλιοθήκη ecdsa της γενικότερης pycoin της γλώσσας προγραμματισμού Python. Θα χρησιμοποιήσουμε τον αλγόριθμο hashing SHA3-256 και έναν γεννήτορα της ελλειπτικής καμπύλης Secp256k1. Οι παραπάνω πράξεις και ελέγχοι που γίνονται για την υπογραφή και επαλήθευση είναι ήδη υλοποιημένες στις βιβλιοθήκες και τις χρησιμοποιούμε ως `generator.sign()`, `generator.verify()`. Άρα μπορούμε να δούμε παρακάτω ένα παράδειγμα που υπογράφουμε ένα μήνυμα, κάνουμε επαλήθευση της υπογραφής και τι γίνεται αν παραλλάξουμε το μήνυμα:

```

from pycoin.ecdsa.secp256k1 import secp256k1_generator
import hashlib, secrets

def sha3_256Hash(msg):
    hashBytes = hashlib.sha3_256(msg.encode("utf8")).digest()
    return int.from_bytes(hashBytes, byteorder="big")

def signECDSAsecp256k1(msg, privKey):
    msgHash = sha3_256Hash(msg)
    signature = secp256k1_generator.sign(privKey, msgHash)
    return signature

def verifyECDSAsecp256k1(msg, signature, pubKey):
    msgHash = sha3_256Hash(msg)
    valid = secp256k1_generator.verify(pubKey, msgHash, \
signature)
    return valid

# ECDSA sign message (using the curve secp256k1 + SHA3-256)
msg = "Message_for_ECDSA_signing"
privKey = secrets.randbelow(secp256k1_generator.order())
signature = signECDSAsecp256k1(msg, privKey)
print("Message:", msg)
print("Private_key:", hex(privKey))
print("Signature:_r=" + hex(signature[0]) + ",_s=" + \
hex(signature[1]))

# ECDSA verify signature (using the curve secp256k1 + SHA3-256)
pubKey = (secp256k1_generator * privKey)
valid = verifyECDSAsecp256k1(msg, signature, pubKey)
print("_nMessage:", msg)
print("Public_key:_(" + hex(pubKey[0]) + ",_" + \
hex(pubKey[1]) + ")")
print("Signature_valid?", valid)

# ECDSA verify tampered signature
 #(using the curve secp256k1 + SHA3-256)
msg = "Tampered_message"
valid = verifyECDSAsecp256k1(msg, signature, pubKey)
print("_nMessage:", msg)
print("Signature_(tampered_msg)_valid?", valid)

```

```
algebro@DESKTOP-PQB10LR: /mnt/c/Users/dimit/Desktop/TeX/EC
algebro@DESKTOP-PQB10LR:/mnt/c/Users/dimit/Desktop/TeX/EC$ python3 sign_exa.py
Message: Message for ECDSA signing
Private key: 0x2e589b374d0b28c7fb38fd5d29a60c089cd0654aa23afc116b7d60bb98d6501
Signature: r=0x497a83a8361348b5b41c6f7e5af299c758910624eabaf68bbad7c58a1706d116, s=0x8520
745b31a38242dd63d588cea076895c8a7747b5f8509b61f4eb278509c205

Message: Message for ECDSA signing
Public key: (0xf0a96eb3cdeffcecf9d8833dad73bd8abc0ce0f6fb6114fe25e871b17633e927, 0x4833c3
fd9f66769361042a1d3c581d4c0e8efc7c1a301ef24c38b6a42d13bf2f)
Signature valid? True

Message: Tampered message
Signature (tampered msg) valid? False
algebro@DESKTOP-PQB10LR:/mnt/c/Users/dimit/Desktop/TeX/EC$
```

Στα παραπάνω, αν ορίσουμε την ακόλουθη συνάρτηση:

```
def recoverPubKeyFromSignature(msg, signature):
    msgHash = sha3_256Hash(msg)
    recoveredPubKeys =
        secp256k1_generator.possible_public_pairs_for_signature(
            msgHash, signature)
    return recoveredPubKeys
```

και τρέξουμε το ακόλουθο κομμάτι:

```
msg = "Message_for_ECDSA_signing"
privKey = secrets.randbelow(secp256k1_generator.order())
signature = signECDSAsecp256k1(msg, privKey)

recoveredPubKeys = recoverPubKeyFromSignature(msg, signature)
print("\nMessage:", msg)
print("Signature: r=" + hex(signature[0]) + ", s=" + \
hex(signature[1]))
for pk in recoveredPubKeys:
    print("Recovered_public_key_from_signature: (" +
        hex(pk[0]) + ", " + hex(pk[1]) + ")")
```

παίρνουμε ένα αποτέλεσμα σαν το ακόλουθο:

```
algebro@DESKTOP-PQB10LR: /mnt/c/Users/dimit/Desktop/TeX/EC
algebro@DESKTOP-PQB10LR:/mnt/c/Users/dimit/Desktop/TeX/EC$ python3 pubkey_rec.py
Message: Message for ECDSA signing
Signature: r=0xabd79116454a17e221eb6b69b1ab3b8e15db00ba4a5557a1c4d348e3a63963cb, s=0xba556c
eab69f1f267f65995de4feb94178bf6ac3a39136fda4e1357e4671864d
Recovered public key from signature: (0xe75f455d66afd9dfadb7034f549baf12aeb132f9d34faecc82e
0e0c873a76cf4, 0xe4f70b2c40c6c26c7f504c31385246b3c7d32122f1f55a83ec02d61d34e06f67)
Recovered public key from signature: (0x9168ed53f577c298d6e44144ad55b6ae4ceb66c9a927e083985
5728c6bf1865e, 0x5bfb2b20a6d55877e2a6ca862910f6ac279cc5a3671727437b2d242edb63a4de)
algebro@DESKTOP-PQB10LR:/mnt/c/Users/dimit/Desktop/TeX/EC$
```

Όπως είναι αναμενόμενο, θα είναι δύο τα δημόσια κλειδιά που ταιριάζουν με την υπογραφή και το μήνυμα ταυτόχρονα. Το ένα θα είναι το σωστό που θα ταιριάζει και με το ιδιωτικό κλειδί, ενώ το άλλο απλά ταιριάζει στα μαθηματικά της ανάκτησης του δημοσίου κλειδιού. Δηλαδή,

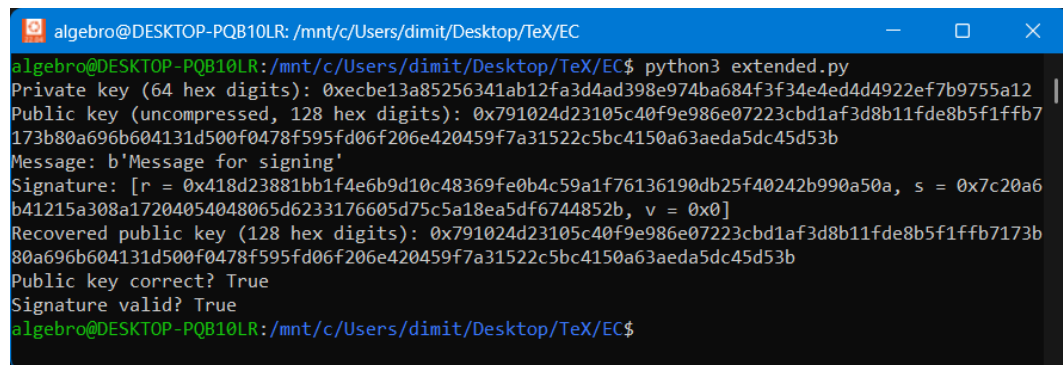
όπως παραπάνω που δοκιμάζουμε το R ή το $-R$ ουσιαστικά αν $R = (x, y)$ τότε $-R = (x, -y)$ πάνω στην ελλειπτική καμπύλη, δηλαδή δεν ξέρουμε το πρόσημο του y το οποίο κρατάμε ως v στην γενικευμένη υπογραφή $\{r, s, v\}$. Ας δούμε ένα παράδειγμα με extended ecdsa. Θα χρησιμοποιήσουμε την βιβλιοθήκη eth keys που είναι μέρος του Ethereum και χρησιμοποιεί κρυπτογραφία ελλειπτικών καμπυλών με βάση την secp256k1 για ιδιωτικά και δημόσια κλειδιά, καθώς και γενικευμένες ECDSA υπογραφές $\{r, s, v\}$.

```
import eth_keys, os

# Generate the private + public key pair
# (using the secp256k1 curve)
signerPrivKey = eth_keys.keys.PrivateKey(os.urandom(32))
signerPubKey = signerPrivKey.public_key
print('Private key (64 hex digits):', signerPrivKey)
print('Public key (uncompressed, 128 hex digits):', signerPubKey)

# ECDSA sign message (using the curve secp256k1 + Keccak-256)
msg = b'Message for signing'
signature = signerPrivKey.sign_msg(msg)
print('Message:', msg)
print('Signature: [r = {0}, s = {1}, v = {2}]'.format(
    hex(signature.r), hex(signature.s), hex(signature.v)))

# ECDSA public key recovery from signature + verify signature
# (using the curve secp256k1 + Keccak-256 hash)
msg = b'Message for signing'
recoveredPubKey = signature.recover_public_key_from_msg(msg)
print('Recovered public key (128 hex digits):', recoveredPubKey)
print('Public key correct?', recoveredPubKey == signerPubKey)
valid = signerPubKey.verify_msg(msg, signature)
print("Signature valid?", valid)
```



```
algebro@DESKTOP-PQB10LR: /mnt/c/Users/dimit/Desktop/TeX/EC
algebro@DESKTOP-PQB10LR:/mnt/c/Users/dimit/Desktop/TeX/EC$ python3 extended.py
Private key (64 hex digits): 0xecbe13a85256341ab12fa3d4ad398e974ba684f3f34e4ed4d4922ef7b9755a12
Public key (uncompressed, 128 hex digits): 0x791024d23105c40f9e986e07223cbd1af3d8b11fde8b5f1ffb7173b80a696b604131d500f0478f595fd06f206e420459f7a31522c5bc4150a63aeda5dc45d53b
Message: b'Message for signing'
Signature: [r = 0x418d23881bb1f4e6b9d10c48369fe0b4c59a1f76136190db25f40242b990a50a, s = 0x7c20a6b41215a308a17204054048065d6233176605d75c5a18ea5df6744852b, v = 0x0]
Recovered public key (128 hex digits): 0x791024d23105c40f9e986e07223cbd1af3d8b11fde8b5f1ffb7173b80a696b604131d500f0478f595fd06f206e420459f7a31522c5bc4150a63aeda5dc45d53b
Public key correct? True
Signature valid? True
algebro@DESKTOP-PQB10LR:/mnt/c/Users/dimit/Desktop/TeX/EC$
```

Εδώ φυσικά θα παίρνουμε πίσω το δημόσιο κλειδί με σιγουριά και η επαλήθευση της υπογραφής θα είναι επιτυχής, εκτός αν φυσικά πειράζουμε το μήνυμα, το δημόσιο κλειδί ή την ίδια την υπογραφή.

4 Vulnerabilities

Παραπάνω έχουμε δει ότι χρησιμοποιούνται συγκεκριμένες καμπύλες όπως η NIST-256 και η secp256k1 που έχουν συγκεκριμένες παραμέτρους. Θα μπορούσαμε να ορίζουμε οποιαδήποτε ελλειπτική καμπύλη και να κάνουμε κρυπτογραφία? Η αλήθεια είναι πως όχι. Υπάρχουν διάφορα κρυπτογραφικά standard που πρέπει να ικανοποιεί μια καμπύλη για να θεωρείται ασφαλής. Ας δούμε μερικά σενάρια ελλειπτικών καμπυλών με κακές παραμέτρους.

4.1 Επίθεση Pohlig-Hellman

Ας υποθέσουμε ότι έχουμε μια ελλειπτική καμπύλη $E(\mathbb{F}_p)$ και γεννήτορα P με τάξη $n = p_1^{r_1} p_2^{r_2} \cdots p_s^{r_s}$, όπου οι p_i είναι πρώτοι αριθμοί και σχετικά μικροί σε μέγεθος. Η επίθεση Pohlig - Hellman ουσιαστικά μεταφέρει το πρόβλημα ECDLP από το $E(\mathbb{F}_p)$ που είναι δύσκολο στις υποομάδες της $\langle G \rangle$ με τάξη πρώτο αριθμό, που είναι αρκετά πιο εύκολο εκεί και στο τέλος λύνει το αρχικό πρόβλημα χρησιμοποιώντας το κινέζικο θεώρημα υπολοίπων.

Υποθέτουμε ότι η τάξη του P έχει την προηγούμενη παραγοντοποίηση και έχοντας το σημείο $Q = kP$ θέλουμε να υπολογίσουμε το k . Αρκεί να βρούμε αριθμούς k_i τέτοιους ώστε:

$$k_i = k \mod p_i^{r_i}$$

και αυτό θα γίνει με το να τους γράψουμε σαν

$$k_i = z_0 + z_1 p_i + \cdots + z_{r_i-1} p_i^{r_i-1}$$

και να υπολογίσουμε τα z_j που αντιστοιχούν στο κάθε k_i . Θέτουμε

$$P_0 = \frac{n}{p_i} P \text{ και } Q_0 = \frac{n}{p_i} Q$$

Το P_0 έχει τάξη p_i αφού $p_i P_0 = nP = \mathcal{O}$ και το p_i είναι πρώτος, άρα δεν έχει μικρότερο διαιρέτη σαν πιθανή τάξη.

$$Q_0 = \frac{n}{p_i} kP = k \left(\frac{n}{p_i} P \right) = kP_0$$

και καθώς η τάξη του P_0 είναι p_i και z_0 ο πρώτος συντελεστής του k_i στην p_i -αδική αναπαράσταση έχουμε ότι

$$kP_0 = k_i P_0 = z_0 P_0 + z_1 p_i P_0 + \cdots + z_{r_i-1} p_i^{r_i-1} P_0 = z_0 P_0 = Q_0$$

όπου η προηγούμενη αντικατάσταση $k = k_i = z_0$ ισχύει αφού σε όλα γίνονται $\mod p_i$ στην κυκλική $\langle P_0 \rangle$ τάξης p_i . Οπότε σε αυτήν την κυκλική ομάδα τάξης p_i έχει να λυθεί το αρκετά πιο εύκολο ECDLP $P_0 = z_0 Q_0$. Γενικεύοντας αυτό το επιχείρημα παίρνουμε όλα τα z_j που εμφανίζονται στην p_i -αδική αναπαράσταση του k_i ως εξής:

$$Q_j = \frac{n}{p_i^{j+1}} \left(Q - z_0 P - z_1 p_i P - z_2 p_i^2 P - \cdots - z_{j-1} p_i^{j-1} P \right)$$

οπότε κάνοντας την ίδια διαδικασία για τα διαφορετικά p_i παίρνουμε το σύστημα εξισώσεων:

$$\begin{aligned} k &= k_1 \mod p_1^{r_1} \\ k &= k_2 \mod p_2^{r_2} \\ k &= k_3 \mod p_3^{r_3} \\ &\vdots \\ k &= k_s \mod p_s^{r_s} \end{aligned}$$

Το οποίο μπορούμε να λύσουμε με το κινέζικο θεώρημα υπολοίπων αφού όλα τα $p_i^{r_i}$ είναι σχετικά πρώτα ανά δύο. Έτσι υπολογίζουμε το k και λύνουμε σε αυτή τη περίπτωση το ECDLP. Ας δούμε αυτό το παράδειγμα υπολογιστικά. Ο παρακάτω κώδικας για το SageMath είναι από το (reference Novotney).

```
def PohligHellman(P,Q):
....:     zList = list()
....:     conjList = list()
....:     rootList = list()
....:     n = P.order()
....:     factorList = n.factor()
....:     for facTuple in factorList:
....:         P0 = (ZZ(n/facTuple[0]))*P
....:         conjList.append(0)
....:         rootList.append(facTuple[0]^facTuple[1])
....:         for i in range(facTuple[1]):
....:             Qpart = Q
....:             for j in range(1,i+1):
....:                 Qpart = Qpart - (zList[j-1]*(facTuple[0]^(j-1))*P)
....:             Qi = (ZZ(n/(facTuple[0]^(i+1))))*Qpart
....:             zList.insert(i,discrete_log(Qi,P0,operation='+'))
....:             conjList[-1]=conjList[-1]+zList[i]*(facTuple[0]^i)
....:     return crt(conjList,rootList)
```

Και ας δούμε ένα μικρό και ένα λίγο πιο μεγάλο παράδειγμα:

```
SageMath 9.0 Console
sage: k = 21345332
sage: p = 4516284508517
sage: E = EllipticCurve(GF(p), [7,1])
sage: P = E.gens()[0]
sage: P
(33660320910 : 61691498829 : 1)
sage: Q = k*P
sage: P.order().factor()
11 * 13 * 31582419389
sage: time findk = PohligHellman(P,Q)
CPU times: user 18.2 s, sys: 375 ms, total: 18.5 s
Wall time: 18.8 s
sage: findk
21345332
sage: 
```

```
SageMath 9.0 Console
sage: p = 310717010502520989590157367261876774703
sage: E = EllipticCurve(GF(p), [2,3])
sage: P = E(179210853392303317793440285562762725654, 1052686714999
....: 42631758568591033409611165)
sage: P.order().factor()
2 * 3^7 * 139 * 165229 * 31850531 * 270778799 * 179317983307
sage: Q = E(280810182131414898730378982766101210916, 29150649076805
....: 4478159835604632710368904)
sage: time findk = PohligHellman(P,Q)
CPU times: user 1min 19s, sys: 1.2 s, total: 1min 20s
Wall time: 1min 22s
sage: findk * P == Q
True
sage: 
```

Όπως βλέπουμε, σε μια τέτοια κακή επιλογή παραμέτρων το ECDLP είναι όσο δύσκολο είναι για την κυκλική ομάδα με τάξη τον μεγαλύτερο διακεκριμένο πρώτο αριθμό που εμφανίζεται στην παραγοντοποίηση της τάξης του γεννήτορα P .

Ας κάνουμε μια παραλλαγή και ας υποθέσουμε ότι στην παραγοντοποίηση $n = p_1^{r_1} p_2^{r_2} \cdots p_s^{r_s}$ ο πρώτος p_s είναι αρκετά μεγάλος για να λύνεται γρήγορα το πρόβλημα του διακριτού λογαρίθμου στην αντίστοιχη κυκλική ομάδα, ενώ οι υπόλοιποι διακεκριμένοι πρώτοι είναι αρκετοί και με μέγεθος που λύνεται το αντίστοιχο πρόβλημα σχετικά γρήγορα στον καθένα. Αν είχαμε την πληροφορία ότι το ιδιωτικό κλειδί είναι σχετικά μικρό σε μέγεθος, γιατί θα μπορούσε να είναι φτωχή υλοποίηση ή να απευθύνεται σε μικρές συσκευές, τότε θα υπήρχε μεγάλη πιθανότητα στην παραπάνω διαδικασία να μπορούσαμε να παραλείψουμε τον πρώτο p_s . Δηλαδή, να κρατούσαμε τις εξισώσεις:

$$\begin{aligned} k &= k_1 \pmod{p_1} \\ k &= k_2 \pmod{p_2} \\ &\vdots \\ k &= k_{s-1} \pmod{p_{s-1}} \end{aligned}$$

και με το κινέζικο θεώρημα να προέκυπτε η λύση $k = a \pmod{M}$ αλλά εν τέλει το M να είναι αρκετά μεγάλο που να μας διασφάλιζε ότι $k = a$ και αν θεωρητικά είχαμε συμπεριλάβει την εξίσωση $k = k_s \pmod{p_s}$ στο κινέζικο θεώρημα απλά να παίρναμε $k = a \pmod{M'}$ για κάποιο $M' > M$. Σε αυτήν την παραλλαγή, δηλαδή με την έξτρα πληροφορία ότι το ιδιωτικό κλειδί είναι σχετικά μικρό σε μέγεθος, μπορούμε να το βρούμε χωρίς να λύσουμε το ECDLP για κάθε υποομάδα με τάξη πρώτο, αλλά ακριβώς για όσες αρκεί και ενδεχομένως αποφεύγοντας τους πολύ μεγάλους πρώτους στην παραγοντοποίηση.

4.2 Ανώμαλες Καμπύλες

Είναι όπως είδαμε στα προηγούμενα κεφάλαια αρκετά βολικό να είναι πρώτος η τάξη του γεννήτορα P , και όπως φαίνεται από την επίθεση Pohlig-Hellman να είναι αυτός ο πρώτος σχετικά μεγάλος. Τι γίνεται αν δοκιμάσει κανείς να κάνει κρυπτογραφία με μια ελλειπτική καμπύλη $E(\mathbb{F}_p)$ που είναι κυκλική και έχει γεννήτορα P με τάξη το ίδιο το p του σώματος? Μια τέτοια καμπύλη ονομάζεται στην βιβλιογραφία ανώμαλη ελλειπτική καμπύλη.

Η ιδέα εδώ είναι ότι το ECDLP θα μεταφερθεί στο σώμα \mathbb{F}_p στο οποίο λύνεται τετριμμένα. Αρκεί δηλαδή να βρεθεί ένας ισομορφισμός ομάδων:

$$\phi : E(\mathbb{F}_p) \longrightarrow \mathbb{F}_p$$

Ορίζουμε ένα ευθύ άθροισμα:

$$E(\mathbb{F}_p) \oplus \mathbb{F}_p$$

με πράξη

$$(Q_1, a) \oplus (Q_2, b) = (Q_1 + Q_2, a + b + \lambda(Q_1, Q_2))$$

όπου $\lambda(Q_1, Q_2)$ είναι η κλίση της ευθείας που περνάει από τα Q_1, Q_2 ή της εφαπτομένης αν $Q_1 = Q_2$ και αν ένα από τα δύο είναι το \mathcal{O} τότε δίνει 0. Έχοντας ότι $pQ = \mathcal{O}$ για κάθε $Q \in E$ τότε:

$$\begin{aligned} p[Q, 0] &= [Q, 0] \oplus [Q, 0] \oplus \cdots \oplus [Q, 0] = \\ &[\mathcal{O}, \lambda(Q, Q) + \lambda(2Q, Q) + \cdots + \lambda((p-1)Q, Q)] := [\mathcal{O}, a] \\ \phi(Q) &:= a \end{aligned}$$

και από την βιβλιογραφία [17] αυτή η ϕ είναι καλά ορισμένη και είναι ο ζητούμενος ισομορφισμός ομάδων. Τώρα αφού έχουμε:

$$\begin{aligned}
kP = Q &\implies \\
k\phi(P) = \phi(Q) &\implies \\
k = \phi(Q)\phi(P)^{-1} &\in \mathbb{F}_p
\end{aligned}$$

το οποίο λύνεται τετριμμένα.

4.3 Singular Καμπύλες

Στον ορισμό μιας ελλειπτικής καμπύλης $y^2 = x^3 + c_1x + c_2$ δώσαμε την σχέση $16(4c_1^3 + 27c_2^2) \neq 0$, δηλαδή της διακρίνουσας ενός πολυώνυμου τρίτου βαθμού να μην είναι 0. Αυτό είναι ισοδύναμο με το πολυώνυμο $x^3 + c_1x + c_2$ να έχει μόνο απλές ρίζες στο σώμα που το βλέπουμε και σε κάθε επέκτασή του. Τι θα γινόταν αν κάποιος επιχειρούσε να κάνει κρυπτογραφία με μια τέτοια καμπύλη που μοιάζει με ελλειπτική και το πολυώνυμο $x^3 + c_1x + c_2$ έχει πολλαπλές ρίζες; Τέτοιες καμπύλες τις συναντά κανείς στην αλγεβρική γεωμετρία και λέγονται singular και τα αντίστοιχα σημεία με πολλαπλές ρίζες λέγονται singular points. Τέτοιες καμπύλες προκύπτουν εντελώς φυσιολογικά κοιτώντας ελλειπτικές καμπύλες πάνω από το \mathbb{Q} με $\Delta \neq 0$ και κάνοντας αναγωγή mod p με $\Delta \equiv 0 \pmod{p}$ προκύπτει πλέον μια καμπύλη που δεν είναι ελλειπτική πάνω από το \mathbb{F}_p .

Ουσιαστικά, σε ένα singular σημείο η καμπύλη θα διασταυρώνεται μεταξύ της και έτσι δεν θα μπορεί να οριστεί η εφαπτομένη για να χρησιμοποιήσουμε τον κανόνα πρόσθεσης σημείων. Ωστόσο, αν πετάξουμε τα singular σημεία και κρατήσουμε όλα τα υπόλοιπα της ελλειπτικής μας μένει πάλι μια αβελιανή ομάδα που την συμβολίζουμε με E_{ns} . Έχουμε δύο περιπτώσεις:

$$y^2 = x^3 + c_1x + c_2 = f(x), \quad f(x) \in \mathbb{F}_p[x]$$

$$f(x) = (x-a)^3 \quad \text{ή} \quad f(x) = (x-a)^2(x-b)$$

Η πρώτη περίπτωση είναι τετριμμένη καθώς έχουμε το μόνο singular σημείο $(a, 0)$ και έτσι για την ομάδα E_{ns} ισχύει ότι:

$$E_{ns} \simeq (\mathbb{F}_p, +)$$

η οποία είναι προσθετική, πράγμα που κάνει άμεση την επίλυση του διακριτού αλγορίθμου.

Στην δεύτερη περίπτωση, η ομάδα E_{ns} θα έχει τάξη $p-1$ ή $p+1$, δηλαδή θα είναι ισόμορφη με μια από τις πολλαπλασιαστικές \mathbb{F}_p^\times ή $\mathbb{F}_{p^2}^\times$. Αυτό γίνεται ως εξής, σε αυτήν την περίπτωση θα ορίζονται δύο ευθείες που προσπαθούν να πάρουν τον ρόλο της εφαπτομένης, οι ευθείες:

$$\varepsilon_1 : y = \sqrt{a-b}(x-a) \quad \text{και} \quad \varepsilon_2 : y = -\sqrt{a-b}(x-a)$$

και με βάση αυτές ορίζεται ο ομομορφισμός ομάδων:

$$\begin{aligned}
\phi : E_{ns} &\longrightarrow \overline{\mathbb{F}_p} = \bigcup_{i \geq 1} \mathbb{F}_{p^i} \\
(x, y) &\longmapsto \frac{y + \sqrt{a-b}(x-a)}{y - \sqrt{a-b}(x-a)}
\end{aligned}$$

όπου $\overline{\mathbb{F}_p}$ είναι η αλγεβρική θήκη του \mathbb{F}_p με πράξη τον πολλαπλασιασμό. Άρα αν θέλουμε να λύσουμε τον διακριτό λογάριθμο $P = kQ$ τότε θα έχουμε:

$$\phi(P) = \phi(Q)^k$$

και άρα μένει να λυθεί ο διακριτός λογάριθμος στην πολλαπλασιαστική ομάδα \mathbb{F}_p^\times ή στην $\mathbb{F}_{p^2}^\times$, ανάλογα σε ποιο από τα $\mathbb{F}_p, \mathbb{F}_{p^2}$ βρίσκονται τα $\phi(P), \phi(Q) \in \overline{\mathbb{F}_p}$. Αν το $a-b$ είναι τετράγωνο τότε το $\sqrt{a-b}$ θα βρίσκεται στο μικρότερο \mathbb{F}_p , διαφορετικά θα είναι στο \mathbb{F}_{p^2} . Εν κατακλείδι, εφόσον οι πολλαπλασιαστικές ομάδες πεπερασμένων σωμάτων είναι κυκλικές, η δυσκολία του

διακριτού λογαρίθμου σε singular καμπύλες ανάγεται στην δυσκολία που έχει ο διακριτός λογαρίθμος σε μια κυκλική ομάδα τάξης $p - 1$ ή αντίστοιχα $p + 1$, το οποίο υπολογίζεται γρήγορα εφόσον το $p - 1$ ή αντίστοιχα το $p + 1$ δεν έχει σχετικά μεγάλους πρώτους παράγοντες. Περισσότερες λεπτομέρειες στους Sivlerman [7] και Washington [8] στις σελίδες 55-58 και 59-64 αντίστοιχα.

4.4 Επίθεση MOV

Μια ιδιαίτερη επίθεση στο πρόβλημα ECDLP είναι αυτή των A. Menezes, T. Okamoto και S. Vanstone reference. Η ιδέα είναι να μεταφέρουμε το πρόβλημα διακριτού λογαρίθμου από το $E(\mathbb{F}_p)$ στο \mathbb{F}_p^k για μικρό k ώστε να λύνεται εύκολα. Για παράδειγμα, μια ελλειπτική καμπύλη πάνω από το \mathbb{F}_p με p να είναι της τάξης των 256 bit προσφέρει ασφάλεια 128 bit, δηλαδή χρειάζονται 2^{128} βήματα στην επίθεση. Αλλά, αν το k είναι δύο, τότε το πρόβλημα εμφυτεύεται στο \mathbb{F}_p^2 που παρέχει μόνο 60 bit ασφάλειας. Ουσιαστικά, η επίθεση βασίζεται σε μια συγκεκριμένη μη-εκφυλισμένη, εναλλάσσουσα, πολλαπλασιαστικά διγραμμική μορφή με το όνομα ζευγάρισμα Weil. Η διγραμμική μορφή αυτή έχει περίπλοκους ορισμούς στην βιβλιογραφία, αλλά στην ουσία παίρνει στοιχεία από τις ομάδες στρέψης $E[n] = \{P \in E : nP = \mathcal{O}\}$ και τα απεικονίζει σε ρίζες της μονάδας στην αλγεβρική θήκη του σώματος K με χαρακτηριστική μεγαλύτερη του 0 που βρισκόμαστε, εφόσον $\text{mcd}(n, \text{char}(K)) = 1$. Δηλαδή:

$$e : E[n] \times E[n] \longrightarrow \{x \in \overline{K} : x^n = 1\}$$

με τις ιδιότητες:

- $e(rP, sQ) = (e(P, Q))^{rs}$ για κάθε $P, Q \in E[n]$ και $r, s \in \mathbb{N}$.
- $e(P_1 + P_2, Q) = e(P_1, Q) \cdot e(P_2, Q)$
- $e(P, Q_1 + Q_2) = e(P, Q_1) \cdot e(P, Q_2)$
- $e(P, P) = 1$
- $e(P, Q) = e(Q, P)^{-1}$
- $e(P, Q) = 1$ για κάθε Q αν και μόνο αν $P = \mathcal{O}$.
- $e(P, Q) = 1$ για κάθε P αν και μόνο αν $Q = \mathcal{O}$.

Υποθέτουμε ότι θέλουμε να λύσουμε το ECDLP με γεννήτορα P με τάξη n και σημείο xP ψάχνοντας το x και έχουμε ένα $Q \in E[n]$ που δεν ανήκει στην κυκλική ομάδα που ορίζει το P . Με άλλα λόγια, τα P, Q είναι γραμμικά ανεξάρτητα με την έννοια ότι δεν υπάρχει φυσικός m με $Q = mP$. Τότε μπορούν να υπολογιστούν τα ακόλουθα:

$$e(P, Q) \text{ και } e(xP, Q) = e(P, Q)^x$$

και είναι και τα δύο n -οστές ρίζες της μονάδας. Εφόσον, τα P, Q είναι γραμμικά ανεξάρτητα και η e μη-εκφυλισμένη τότε $e(P, Q) \neq 1$. Άρα πράγματι μεταφέρθηκε το πρόβλημα από ECDLP σε DLP στην πολλαπλασιαστική ομάδα του σώματος K . Για $K = \mathbb{F}_p$ έχουμε $\mathbb{F}_p = \cup_{i \geq 1} \mathbb{F}_{p^i}$ και θέλουμε να περιοριστούμε όσο το δυνατόν γίνεται σε μικρότερο σώμα, το οποίο το πετυχαίνουμε στο

$$\mathbb{F}_{p^\ell}, \quad \text{όπου } \ell = \min\{m \in \mathbb{N} : p^m = 1 \pmod{|E|}\}$$

και εκεί γίνεται η επίθεση με μεθόδους όπως στην Pohlig-Hellman που σπάει το πρόβλημα διακριτού λογαρίθμου σε μικρότερα προβλήματα και στο τέλος γίνεται σύνθεση της λύσης με το κινέζικο θεώρημα υπολοίπων. Επιπλέον, για μια τάξη ελλειπτικών καμπυλών με το όνομα υπεριδιάζουσες (supersingular) ενώ το αρχικό πρόβλημα ECDLP είναι αρκετά δύσκολο, με την παραπάνω διαδικασία γίνεται αρκετά εύκολο καθώς πετυχαίνεται η τιμή $\ell = 2$.

4.5 CurveBall

Τα προηγούμενα Vulnerabilities αποτελούν πλέον εκπαιδευτικά παραδείγματα παρά κάτι που θα συναντήσει κανείς εφόσον υπάρχουν τα στάνταρ που ελέγχονται στις ελλειπτικές καμπύλες που χρησιμοποιούνται. Ωστόσο, υπήρξε ένα ιδιαίτερα εύκολο από μαθηματικής άποψης κενό ασφαλείας σε υλοποίηση της Microsoft που βρέθηκε μόλις το 2020. Στο κενό δώθηκαν τα ονόματα CurveBall και Chain of Fools, το πρώτο κυρίως για την μαθηματική πράξη πάνω στην καμπύλη, ενώ το δεύτερο καθώς το κενό έχει να κάνει με πολλές ψηφιακές υπογραφές.

Αυτό που συνέβη ήταν ότι υπήρχε κενό στον έλεγχο και στον τρόπο που επαλήθευε τις ECC υπογραφές το Windows CryptoAPI στο αρχείο Crypt32.dll. Ένας κακόβουλος χρήστης μπορούσε να εκμεταλλευτεί το κενό και να υπογράψει ένα δικό του κακόβουλο αρχείο και να το κάνει να φαίνεται σαν η υπογραφή να ήταν από μια έμπιστη πηγή. Έτσι ο οποιοσδήποτε χρήστης μετά θα βασιζόταν στην πλέον έμπιστη υπογραφή και δεν θα μπορούσε να ξέρει ότι το λογισμικό που έχει λάβει είναι κακόβουλο. Το κενό που υπήρχε πραγματικά ήταν ότι τα Windows έκαναν έλεγχο αν είναι η δεδομένη γνωστή καμπύλη, γινόταν έλεγχος για το δημόσιο κλειδί και άλλες παραμέτρους της ελλειπτικής καμπύλης, αλλά δεν γινόταν έλεγχος ότι όλοι χρησιμοποιούν τον ίδιο γεννήτορα.

Τα μαθηματικά είναι πιο απλά από όλα τα προηγούμενα παραδείγματα, παρόλο που αυτό είναι πραγματικό σενάριο. Ας υποθέσουμε ότι χρησιμοποιείται η καμπύλη $P-256$ με τον γεννήτορα G και ότι η Microsoft έχει δημόσιο κλειδί $Q = xG$. Αν πούμε ότι ο A ήθελε να επιτεθεί, θα μπορούσε να ορίσει την ίδια ακριβώς καμπύλη και παραμέτρους της περνώντας κάθε έλεγχο έτσι, αλλά με διαφορετικό γεννήτορα. Ο A δεν έχει καμία ελπίδα να ξέρει το ιδιωτικό κλειδί x της Microsoft, αλλά φτιάχνει το δικό του ιδιωτικό κλειδί x' και δίνει στο CryptoAPI τον γεννήτορα:

$$G' = (x')^{-1}Q$$

Έτσι, το CryptoAPI βλέπει ότι

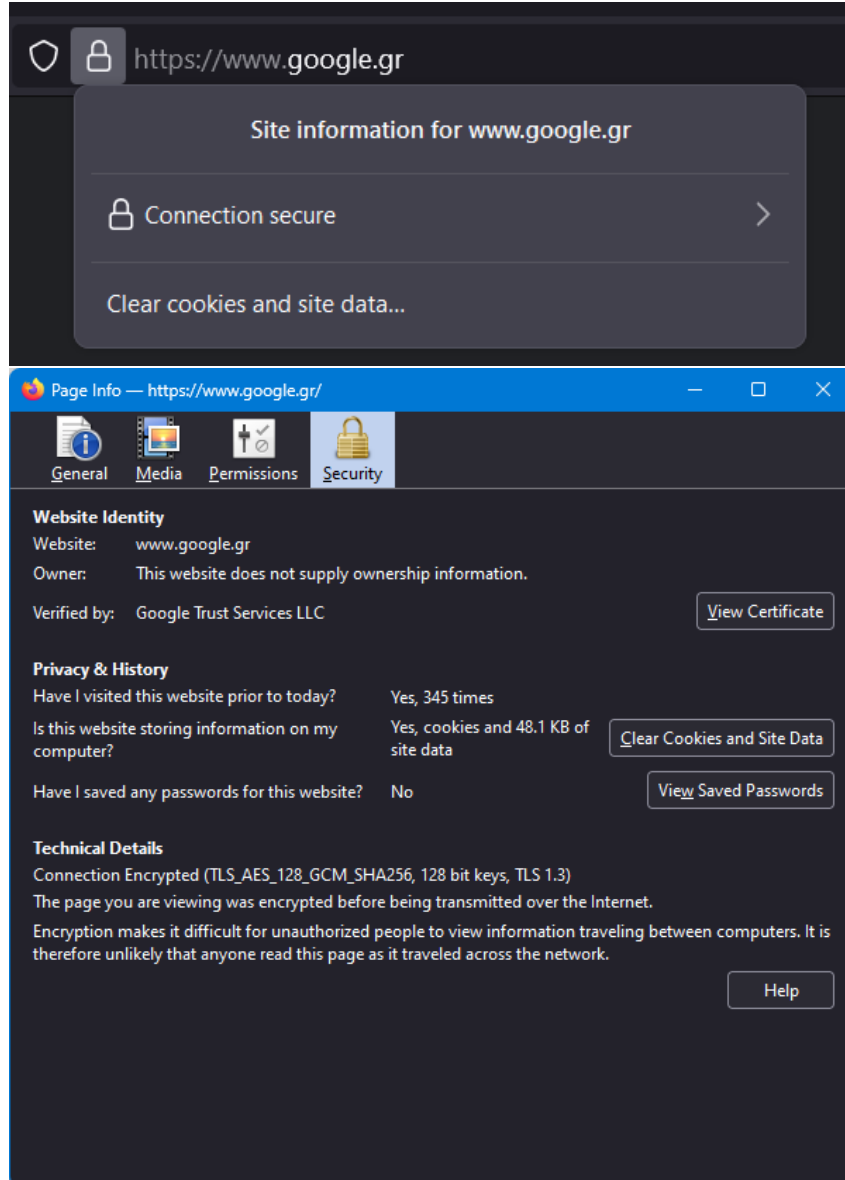
$$Q = x'G'$$

και άρα νομίζει ότι το ιδιωτικό κλειδί που αντιστοιχεί στο Q είναι το x' που ξέρει ο A , ενώ είναι το x της Microsoft. Άρα πλέον ο A θεωρείται έμπιστος ως η ίδια η Microsoft. Μια περιγραφή των γεγονότων μπορεί να διαβάσει κανείς στο blog.lessonslearned.org/chain-of-fools/.

Εδώ είναι ακόμα πιο ξεκάθαρο, ότι όσο εκλεπτυσμένα και να είναι τα μαθηματικά πίσω από την κρυπτογραφία, το πρόβλημα μαθηματικής δυσκολίας είτε θα σπάει από εξωτερικούς παράγοντες, είτε θα μπορεί να παραβλέπεται τελείως και η κρυπτογραφία θα αποτυγχάνει λόγω λανθασμένων υλοποιήσεων.

5 Ασφαλείς Ελλειπτικές Καμπύλες

Αν πάρουμε για παράδειγμα την Google και ψάξουμε το certificate της κάνοντας click στο λουκέτο:



Θα δούμε ότι χρησιμοποιεί την ακόλουθη καμπύλη με το ακόλουθο δημόσιο κλειδί:

Public Key Info	
Algorithm	Elliptic Curve
Key Size	256
Curve	P-256
Public Value	04:28:63:53:78:1D:83:FC:EA:47:0F:54:66:B6:86:56:BA:23:85:EE:33:37:FC:4B:88:B7:2 5:CE:E3:B9:8E:A4:90:4D:19:59:63:9C:A0:0E:91:11:9C:C1:FA:21:B2:5C:BC:F9:C1:7E:D 4:DE:7E:F7:9B:A4:40:C4:9B:E6:58:EB:EE

Η οποία είναι η $P - 256$ από το Εθνικό Ινστιτούτο Προτύπων και Τεχνολογίας (NIST) της Αμερικής. Όπως βλέπουμε εδώ αλλά και στα παραδείγματα που υπάρχουν έτοιμες οι καμπύλες στις βιβλιοθήκες της Python, δεν είναι ασφαλές να φτιάχνει κανείς τις δικές του ελλειπτικές καμπύλες καθώς πρέπει αφενός να αντιστέκονται στις επιθέσεις και αφετέρου να μην υπάρχουν άλλα κενά ασφαλείας. Για την αντοχή στις επιθέσεις, υπάρχουν πολλά πρότυπα για την επιλογή ελλειπτικών καμπυλών που θεωρούνται ασφαλείς όπως τα:

- ANSI X9.62 (1999).
- IEEE P1363 (2000).
- SEC 2 (2000).
- NIST FIPS 186-2 (2000).
- ANSI X9.63 (2001).
- Brainpool (2005).
- NSA Suite B (2005).
- ANSSI FRP256V1 (2011).

και στον σύνδεσμο www.rfc-editor.org/rfc/rfc5639 είναι δημοσιευμένος ο αλγόριθμος για την επιλογή παραμέτρων των NIST και Brainpool καμπυλών. Η εύρεση ασφαλών ελλειπτικών καμπυλών είναι κάτι που μελετάται συνεχώς. Η Certicom ανανεώνει την λίστα Standards for Efficient Cryptography: Recommended Elliptic Curve Parameters (SEC 2) κάθε πέντε χρόνια reference και γίνονται συνεχώς νέες προσπάθειες όπως η πρόσφατη στο [20].

Δυστυχώς, υπάρχει ένα χάσμα μεταξύ της δυσκολίας του ECDLP και της ασφάλειας του ECC. Το ECDLP έχει να κάνει με το να ανακτήσει κάποιος το ιδιωτικό κλειδί του ECC χρήση με δεδομένο το δημόσιο κλειδί. Ακόμα και κάποιος να χρησιμοποιεί τις παραπάνω καμπύλες, οι πιθανότητες να τις χρησιμοποιεί θεωρητικά σωστά είναι εναντίων του. Υπάρχουν πολλές πραγματικές επιθέσεις στο ECC που δεν χρειάζεται να λύσουν το ECDLP. Για παράδειγμα, σε μια υλοποίηση μπορεί να συμβαίνουν τα ακόλουθα:

- Να παράγει λάθος αποτέλεσμα για σημεία της καμπύλης που προκύπτουν σπάνια.
- Να διαρρέει εσωτερική πληροφορία όταν δωθεί σαν είσοδος κάτι που δεν είναι σημείο στην καμπύλη.
- Να δέχεται timing επιθέσεις, δηλαδή να εκμεταλλεύεται κάποιος επιτιθέμενος τον χρόνο που χρειάζεται η υλοποίηση να τρέξει τους κρυπτογραφικούς αλγόριθμους και έτσι να διαρρέει εσωτερική πληροφορία.

Οι επιτιθέμενοι ουσιαστικά εκμεταλλεύονται αυτό το χάσμα του ECDLP και του ECC για τους ακόλουθους λόγους:

- Το ECDLP είναι θεωρητικό πρόβλημα, ενώ το ECC αντιδρά με τις εισόδους του χρήστη που μπορεί να είναι κακόβουλες.
- Το ECDLP δίνει μόνο την πληροφορία nP , ενώ από το ECC μπορεί να ξεφεύγει με διάφορους τρόπους πληροφορία από το κανάλι και από το εσωτερικό σύστημα, καθώς και χρόνοι εκτέλεσης.
- Το ECDLP απλά υπολογίζει το nP θεωρητικά σωστά, ενώ στην πραγματικότητα στο ECC μπορεί να υπάρξουν σφάλματα.

Ένα νέο έργο που προσπαθεί να προσφέρει και ECC ασφάλεια πέρα από ECDLP είναι το SafeCurves των Daniel Bernstein και Tanja Lange reference. Το συγκεκριμένο έργο περνάει ένα επιπλέον φιλτράρισμα για ECC ασφάλεια στις λίστες των καμπυλών που αναφέρθηκαν προηγουμένως.

Curve	Safe?	Details
Anomalous	False	$y^2 = x^3 + 15347898055371580590890576721314318823207531963035637503096292x + 7444386449934505970367865204569124728350661870959593404279615$ modulo $p = 17676318486848893030961583018778670610489016512983351739677143$ Created as an illustration of additive transfer and small discriminant.
M-221	True ✓	$y^2 = x^3 + 117050x^2 + x$ modulo $p = 2^{221} - 3$ 2013 Aranha-Barreto-Pereira-Ricardini (formerly named Curve2213)
E-222	True ✓	$x^2 + y^2 = 1 + 160102x^2y^2$ modulo $p = 2^{222} - 117$ 2013 Aranha-Barreto-Pereira-Ricardini
NIST P-224	False	$y^2 = x^3 - 3x + 18958286285566608000408668544493926415504680968679321075787234672564$ modulo $p = 2^{224} - 2^{96} + 1$ 2000 NIST ; also in SEC 2
Curve1174	True ✓	$x^2 + y^2 = 1 - 1174x^2y^2$ modulo $p = 2^{251} - 9$ 2013 Bernstein-Hamburg-Krasnova-Lange
Curve25519	True ✓	$y^2 = x^3 + 486662x^2 + x$ modulo $p = 2^{255} - 19$ 2006 Bernstein
BN(2,254)	False	$y^2 = x^3 + 0x + 2$ modulo $p = 16798108731015832284940804142231733090889187121439069848933715426072753864723$ 2011 Pereira-Simplicio-Naehrig-Barreto pairing-friendly curve. Included as an illustration of multiplicative transfer and small discriminant.
brainpoolP256t1	False	$y^2 = x^3 - 3x + 46214326585032579593829631435610129746736367449296220983687490401182983727876$ modulo $p = 76884956397045344220809746629001649093037950200943055203735601445031516197751$ 2005 Brainpool
ANSSI FRP256v1	False	$y^2 = x^3 - 3x + 107744541122042688792155207242782455150382764043089114141096634497567301547839$ modulo $p = 109454571331697278617670725030735128145969349647868738157201323556196022393859$ 2011 ANSSI
NIST P-256	False	$y^2 = x^3 - 3x + 41058363725152142129326129780047268409114441015993725554835256314039467401291$ modulo $p = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$ 2000 NIST ; also in SEC 2 and NSA Suite B
secp256k1	False	$y^2 = x^3 + 0x + 7$ modulo $p = 2^{256} - 2^{32} - 977$ SEC2
E-382	True ✓	$x^2 + y^2 = 1 - 67254x^2y^2$ modulo $p = 2^{382} - 105$ 2013 Aranha-Barreto-Pereira-Ricardini
M-383	True ✓	$y^2 = x^3 + 2065150x^2 + x$ modulo $p = 2^{383} - 187$ 2013 Aranha-Barreto-Pereira-Ricardini
Curve383187	True ✓	$y^2 = x^3 + 229969x^2 + x$ modulo $p = 2^{383} - 187$ 2013 Aranha-Barreto-Pereira-Ricardini ; authors subsequently recommended switching to M-383
brainpoolP384t1	False	$y^2 = x^3 - 3x + 19596161053329239268181228455226581162286252326261019516900162717091837027531392576647644262320816848087868142547438$ modulo $p = 2165927077011931617306923684233260497979611638701764860008161850382108934025961822236561982844534088440708417973331$ 2005 Brainpool
NIST P-384	False	$y^2 = x^3 - 3x + 27580193559595705877849011840389048093056905856361568521428707301988689241309860865136260764883745107765439761230575$ modulo $p = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$ 2000 NIST ; also in SEC 2 and NSA Suite B
Curve41417	True ✓	$x^2 + y^2 = 1 + 3617x^2y^2$ modulo $p = 2^{414} - 17$ 2013 Bernstein-Lange (formerly named Curve3617)
Ed448-Goldilocks	True ✓	$x^2 + y^2 = 1 - 39081x^2y^2$ modulo $p = 2^{448} - 2^{224} - 1$ 2014 Hamburg
M-511	True ✓	$y^2 = x^3 + 530438x^2 + x$ modulo $p = 2^{511} - 187$ 2013 Aranha-Barreto-Pereira-Ricardini (formerly named Curve511187)
E-521	True ✓	$x^2 + y^2 = 1 - 376014x^2y^2$ modulo $p = 2^{521} - 1$ 2013 Bernstein-Lange; independently 2013 Hamburg; independently 2013 Aranha-Barreto-Pereira-Ricardini

Curve	Safe?	Parameters:			ECDLP security:				ECC security:			
		field	equation	base	rho	transfer	disc	rigid	ladder	twist	complete	ind
Anomalous	False	True✓	True✓	True✓	True✓	False	False	True✓	False	False	False	False
M-221	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
E-222	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
NIST P-224	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False	False
Curve1174	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Curve25519	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
BN(2,254)	False	True✓	True✓	True✓	True✓	False	False	True✓	False	False	False	False
brainpoolP256t1	False	True✓	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False
ANSSI FRP256v1	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	False	False	False
NIST P-256	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	True✓	False	False
secp256k1	False	True✓	True✓	True✓	True✓	True✓	False	True✓	False	True✓	False	False
E-382	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
M-383	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Curve383187	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
brainpoolP384t1	False	True✓	True✓	True✓	True✓	True✓	True✓	True✓	False	True✓	False	False
NIST P-384	False	True✓	True✓	True✓	True✓	True✓	True✓	False	False	True✓	False	False
Curve41417	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
Ed448-Goldilocks	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
M-511	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓
E-521	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓	True✓

πίνακες από reference

Όπως βλέπουμε, οι διάσημες ελλειπτικές καμπύλες NIST P-256 και secp256k1 δεν θεωρούνται ασφαλείς από τα πρότυπα του SafeCurves.

6 Περαιτέρω μελέτη και Isogenies

Καθώς το πρόβλημα ECDLP παραμένει σήμερα υπολογιστικά δύσκολο, στο οποίο βασίζεται και η ανταλλαγή κλειδιού ECDH, δεν θα ισχύει το ίδιο στο μέλλον όταν ένας επιτιθέμενος θα μπορεί να χρησιμοποιήσει έναν μεγάλης κλίμακας κβαντικό υπολογιστή. Ο μαθηματικός Peter Shor έδειξε με δικό του αλγόριθμο *reference* ότι ένας κβαντικός υπολογιστής που καταφέρνει να λειτουργεί ορθά με αρκετά q -bits μπορεί να σπάσει σε πολυωνυμικό χρόνο τα ακόλουθα:

- Το σχήμα RSA.
- Το πρωτόκολλο ανταλλαγής κλειδιού Diffie-Hellman σε πεπερασμένα σώματα.
- Το πρωτόκολλο ανταλλαγής κλειδιού ECDH.

Μέχρι να υπάρξει ένας τέτοιος υπολογιστής, καλούνται οι μαθηματικοί να έχουν έτοιμους κρυπτογραφικούς αλγόριθμους που αποδεδειγμένα να έχουν αντοχή ενάντια σε συγκεκριμένο πλήθος q -bits. Ένας τέτοιος μετακβαντικός αλγόριθμος είναι ο Supersingular Isogeny-Based Diffie-Hellman (SIDH) που βασίζεται σε isogenies και supersingular ελλειπτικές καμπύλες. Η ιδέα είναι ότι θέλουμε ένα πρωτόκολλο σαν το ECDH που να είναι ασφαλές από κβαντικούς υπολογιστές. Σκεφτόμαστε τον βαθμωτό πολλαπλασιασμό:

$$nP = P + P + \dots + P$$

σαν μια απεικόνιση μεταξύ της ελλειπτικής καμπύλης E :

$$E \longrightarrow E$$

$$P \longmapsto nP$$

και ουσιαστικά θέλουμε να έχουμε γενικότερες απεικονίσεις που αντιστοιχούν ελλειπτικές καμπύλες σε ελλειπτικές καμπύλες. Επιπλέον, αντίθετα από τον βαθμωτό πολλαπλασιασμό, δεν χρειάζεται αυτές οι απεικονίσεις να ξεκινούν και να τελειώνουν στην ίδια καμπύλη E .

Ορισμός (Ρητή Απεικόνιση). Μια ρητή απεικόνιση μεταξύ ελλειπτικών καμπύλων είναι μια απεικόνιση που η κάθε συντεταγμένη είναι ο λόγος πολυωνύμων. Δηλαδή:

$$\Phi(P) = \phi(x, y) = \left(\frac{p_1(x, y)}{q_1(x, y)}, \frac{p_2(x, y)}{q_2(x, y)} \right)$$

με p_1, p_2, q_1, q_2 πολυώνυμα δύο μεταβλητών πάνω από το σώμα που ορίζεται η ελλειπτική καμπύλη.

Καθώς ο πολλαπλασιασμός διατηρεί την πρόσθεση, δηλαδή:

$$n(P + Q) = nP + nQ$$

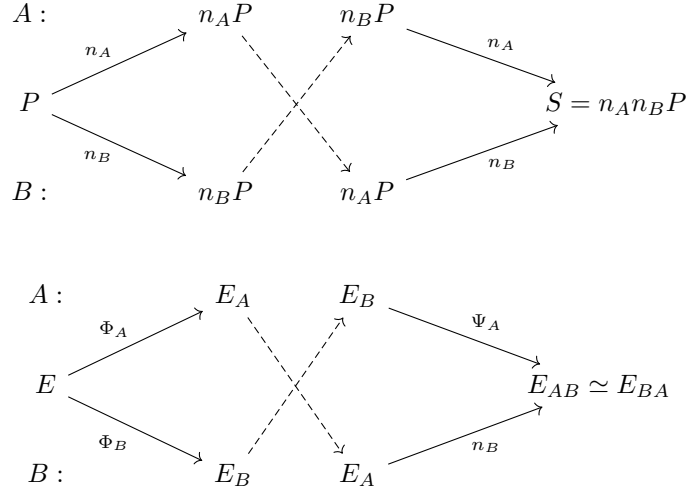
θέλουμε να συμβαίνει το αντίστοιχο:

$$\Phi(P + Q) = \Phi(P) + \Phi(Q)$$

δηλαδή η Φ να είναι ομομορφισμός ομάδων. Έτσι καταλήγουμε στον ορισμό:

Ορισμός (Isogeny). Ένα *isogeny* μεταξύ δύο ελλειπτικών καμπυλών E_1, E_2 είναι μια ρητή απεικόνιση $\Phi : E_1 \rightarrow E_2$ που είναι ομομορφισμός ομάδων.

Η ιδέα τώρα είναι πιο ξεκάθαρη, αν δούμε σε αντιστοιχία το ECDH με το SIDH αντίστοιχα:



Δηλαδή θέλουμε να αντικατασταθούν τα ιδιωτικά κλειδιά n_A, n_B από τα isogenies Φ_A, Φ_B . Τα Ψ_A, Ψ_B είναι στην ουσία το ίδιο με τα Φ_A, Φ_B αντίστοιχα, απλά ξεκινούν από διαφορετική καμπύλη. Όπως σε αντιστοιχία ο δεύτερος πολλαπλασιασμός με n_A δεν ξεκινάει από το P αλλά από το $n_B P$. Ουσιαστικά, το ιδιωτικό κλειδί της Alice θα είναι το Φ_A με το οποίο θα μπορεί να βρει το Ψ_A και όμοια για τον Bob.

Στο ECDH τα δημόσια κλειδιά είναι η εικόνα ενός σημείου μέσα από την απεικόνιση πολλαπλασιασμού. Εδώ στο SIDH η διαφορά είναι ότι πλέον το δημόσιο κλειδί δεν θα είναι ένα σημείο αλλά όπως φαίνεται στο παραπάνω διάγραμμα ολόκληρη η καμπύλη εικόνα. Αυτό βασίζεται στην πρόταση:

Πρόταση. Αν έχουμε isogeny $\Phi : E \rightarrow E'$ τότε η εικόνα $\Phi(E)$ είναι ελλειπτική καμπύλη. Οπότε πλέον θεωρούμε τα isogenies να είναι επιμορφισμοί ομάδων.

Οπότε το δημόσιο κλειδί της Alice θα είναι το $E_A = \Phi_A(E)$. Για τεχνικούς λόγους, θα περιλαμβάνονται στο δημόσιο κλειδί και δύο σημεία.

Για το κοινό μυστικό, στο ECDH υπολογίζεται ακριβώς το ίδιο σημείο $n_A n_B P$. Δυστυχώς, στο SIDH οι αντίστοιχες εικόνες E_{AB} και E_{BA} δεν είναι απαραίτητα η ίδια καμπύλη. Ωστόσο, έχουν την ίδια δομή με βάση τα ακόλουθα:

Ορισμός (Ισομορφισμός). Θα λέμε ισομορφισμό ελλειπτικών καμπυλών E_1, E_2 αν υπάρχει isogeny μεταξύ τους που είναι 1-1, δηλαδή μονομορφισμός ομάδων. Δύο ελλειπτικές καμπύλες θα είναι ισόμορφες αν υπάρχει τέτοιο isogeny μεταξύ τους.

Ένα χρήσιμο κριτήριο για τον ισομορφισμό είναι με βάση ένα αναλλοίωτο μέγεθος που έχουν οι ελλειπτικές καμπύλες.

Ορισμός (j -invariant). Έστω ελλειπτική καμπύλη $E : y^2 = x^3 + ax + b$, ορίζουμε ως j -invariant της E το μέγεθος:

$$j(E) = \frac{6912a^3}{4a^3 + 27b^2}$$

Πρόταση. Δύο ελλειπτικές καμπύλες είναι ισόμορφες (πάνω από την αλγεβρική θήκη του ίδιου σώματος) αν και μόνο αν έχουν το ίδιο j -invariant.

Οπότε για την περίπτωση του SIDH, έχουμε ότι οι καμπύλες E_{AB}, E_{BA} δεν είναι απαραίτητα ίδιες, αλλά ισχύουν τα:

$$E_{AB} \simeq E_{BA}$$

$$j(E_{AB}) = j(E_{BA})$$

Άρα πρέπει η Alice και ο Bob να έχουν έναν τρόπο να διαλέγουν τυχαία isogenies. Αυτό γίνεται εύχρηστα στην κρυπτογραφία βασισμένη σε isogenies με τον ορισμό του πυρήνα:

Ορισμός (Πυρήνας Isogeny). Έστω $\Phi : E_1 \longrightarrow E_2$ isogeny, ο πυρήνας που συμβολίζεται $\ker \Phi$ είναι το σύνολο των στοιχείων:

$$\ker \Phi = \{P \in E_1 \mid \Phi(P) = \mathcal{O}\}$$

Με την ακόλουθη πρόταση παίρνουμε έναν τρόπο να ορίζουμε ολόκληρα isogenies μόνο ορίζοντας τον πυρήνα τους:

Πρόταση. Αν δύο isogenies έχουν τον ίδιο πυρήνα, τότε οι ελλειπτικές καμπύλες εικόνες τους είναι ισόμορφες.

Και μπορούμε να ορίσουμε πυρήνες εύκολα ως την γραμμική θήκη στοιχείων, δηλαδή:

$$\langle P_1, P_2, \dots, P_n \rangle = \{m_1 P_1 + m_2 P_2 + \dots + m_n P_n \in E \mid m_i \in \mathbb{Z}\}$$

και ένα isogeny που ορίζεται από πυρήνα $\langle P \rangle$, δηλαδή από ένα στοιχείο θα λέγεται κυκλικό isogeny.

Οπότε, για να δημιουργήσει το ζεύγος κλειδιών της η Alice πρέπει να επιλέξει τυχαία ένα σημείο R_A της ελλειπτικής καμπύλης συγκεκριμένης τάξης και με αυτό να έχει τον πυρήνα $\langle R_A \rangle$ και το isogeny Φ_A που ορίζει αυτό ως ιδιωτικό κλειδί. Στην συνέχεια, παίρνει το δημόσιο κλειδί της $E_A = \Phi_A(E)$. Δηλαδή, το πρόβλημα μεταφέρεται στο να μπορεί να επιλέξει τυχαία το σημείο R_A στην καμπύλη.

Συνήθως, στην κρυπτογραφία βασισμένη σε isogenies γίνονται οι πράξεις σε μια ελλειπτική καμπύλη E πάνω από το \mathbb{F}_{p^2} για κάποιον πρώτο της μορφής $p = 2^a 3^b - 1$ με $a, b \in \mathbb{N}$. Χρησιμοποιείται επιπλέον ότι οι υποομάδες στρέψης $E[2^a]$ και $E[3^b]$ μπορούν να παραχθούν από δύο στοιχεία το καθένα P_A, Q_A και P_B, Q_B αντίστοιχα. Έτσι, αυτά τα δύο σημεία (για τα οποία υπάρχουν πολλές επιλογές σαν βάση) θα είναι η επιπλέον πληροφορία που θα περιλαμβάνει το δημόσιο κλειδί. Έτσι, η Alice μπορεί πλέον να επιλέξει τυχαία έναν φυσικό αριθμό r_A και να της δώσει αντίστοιχο σημείο R_A :

$$0 \leq r_A \leq 2^a$$

$$R_A = P_A + r_A Q_A$$

και η υποομάδα $\langle R_A \rangle$ βρίσκεται μέσα στην υποομάδα στρέψης $E[2^a]$. Αντίστοιχα, ο Bob θα έχει ένα isogeny με πυρήνα $\langle R_B \rangle$ που βρίσκεται μέσα στην $E[3^b]$. Το ότι οι τάξεις των πυρήνων των isogenies είναι δυνάμεις μικρών πρώτων αριθμών κάνει πιο γρήγορο τον υπολογισμό τους, καθώς και είναι βοηθητικό οι τάξεις να είναι σχετικά πρώτες. Τα μεγέθη a, b συσχετίζονται με την ασφάλεια που έχει το πρωτόκολλο.

Θέλουμε να καταλήξουμε σε ίδια j -invariants και αυτό πετυχαίνεται με το να ορίσουμε Ψ_A το isogeny που προκύπτει από τον πυρήνα $\langle \Phi_B(R_A) \rangle$ και με αυτά αποδεικνύεται ότι:

$$\Psi_A \circ \Phi_B = \langle R_A, R_B \rangle$$

και όμοια:

$$\Psi_B \circ \Phi_A = \langle R_A, R_B \rangle$$

έτσι θα έχουμε:

$$\ker(\Psi_A \circ \Phi_B) = \ker(\Psi_B \circ \Phi_A)$$

και άρα αποδεικνύεται ότι:

$$E_{AB} = \Psi_A \circ \Phi_B(E) \simeq \Psi_B \circ \Phi_A(E) = E_{BA}$$

και άρα παίρνει και η Alice και ο Bob ως κοινό μυστικό την τιμή:

$$j(E_{AB}) = j(E_{BA})$$

Όπως φαίνεται παραπάνω, για να κατασκευάσει η Alice το Ψ_A χρειάζεται να ξέρει το $\Phi_B(R_A)$. Για αυτόν τον λόγο, ο Bob στο δημόσιο κλειδί του περιλαμβάνει τις τιμές

$$\Phi_B(P_A), \Phi_B(Q_A)$$

δηλαδή τις εικόνες της προκαθορισμένης βάσης του $E[2^a]$ μέσω της Φ_B που είναι το ιδιωτικό κλειδί του Bob. Έτσι η Alice υπολογίζει:

$$\Phi_B(R_A) = \Phi_B(P_A) + r_A \Phi_B(Q_A)$$

Συνδυάζοντας τα παραπάνω, μπορούμε να κάνουμε το διάγραμμα του πρωτοκόλλου SIDH. Χρειαζόμαστε μια καμπύλη E που αντέχει μια κβαντική επίθεση όπως αυτή του Shor που σπάει το ECDH. Συγκεκριμένα, μπορούμε να διαλέξουμε $E : y^2 = x^3 + ax + b$ πάνω από το \mathbb{F}_{p^2} για πρώτο $p = 2^a 3^b - 1$, αλλά και οποιαδήποτε supersingular καμπύλη αρκεί. Supersingular είναι οι καμπύλες πάνω από το σώμα K για τις οποίες ισχύει το παρακάτω για τις υποομάδες στρέψης:

$$E[p^r](K) = \begin{cases} \mathbb{Z}/p^r\mathbb{Z} & \text{ή} \\ 0 & \end{cases}$$

Οπότε έχοντας τις δημόσιες παραμέτρους:

$$E(\mathbb{F}_{p^2}) : y^2 = x^3 + ax + b$$

$$p = 2^a 3^b - 1, \quad a, b \in \mathbb{N}$$

$$P_A, Q_A \text{ μια βάση του } E[2^a]$$

$$P_B, Q_B \text{ μια βάση του } E[3^b]$$

Έχουμε το διάγραμμα:

SIDH

Alice	Bob
Διάλεξε $0 \leq r_A < 2^a$ r_A ιδιωτικό κλειδί	Διάλεξε $0 \leq r_B < 3^b$ r_B ιδιωτικό κλειδί
Δημόσιο κλειδί: $E_A = \Phi_A(E), \Phi_A(P_B), \Phi_A(Q_B)$ όπου Φ_A isogeny με $\ker \Phi_A = \langle P_A + r_A Q_A \rangle$	Δημόσιο κλειδί: $E_B = \Phi_B(E), \Phi_B(P_A), \Phi_B(Q_A)$ όπου Φ_B isogeny με $\ker \Phi_B = \langle P_B + r_B Q_B \rangle$
$\xrightarrow{\text{στέλνει η Alice το } E_A}$ $\xleftarrow{\text{στέλνει ο Bob το } E_B}$	
Κοινό μυστικό: $j(\Psi_A(E_B))$ όπου Ψ_A isogeny με $\ker \Psi_A = \langle \Phi_B(P_A) + r_A \Phi_B(Q_A) \rangle$	Κοινό μυστικό: $j(\Psi_B(E_A))$ όπου Ψ_B isogeny με $\ker \Psi_B = \langle \Phi_A(P_B) + r_B \Phi_A(Q_B) \rangle$
Κλειδί:	$j(\Psi_A(E_B)) = j(\Psi_B(E_A))$

Εν κατακλείδι, η ασφάλεια του SIDH οφείλεται στην δυσκολία του ακόλουθου προβλήματος:

Ορισμός. Υποθέτουμε ότι υπάρχει *isogeny* Φ μεταξύ δύο ελλειπτικών καμπύλων E_1, E_2 , όπου ο πυρήνας έχει τάξη ℓ^e . Το ℓ^e -*isogeny* πρόβλημα είναι το πρόβλημα του να υπολογίσει κανείς τον πυρήνα $\ker \Phi$ δεδομένου μόνο των E_1, E_2 .

Δηλαδή, πρέπει τα αντίστοιχα $2^a, 3^b$ -isogeny προβλήματα να είναι δύσκολα, για να μην μπορεί κανείς να βρει το ιδιωτικό κλειδί της Alice ή του Bob αντίστοιχα. Αυτό το πρόβλημα έχει μελετηθεί τα τελευταία 20 χρόνια από μαθηματικούς που ειδικεύονται στην θεωρία αριθμών και θεωρείται εξαιρετικά δύσκολο για μεγάλες δυνάμεις του ℓ , για οποιονδήποτε φυσικό αριθμό ℓ . Για αυτό, θεωρείται ότι είναι ικανό να αντέξει επίθεση από κβαντικό υπολογιστή μεγάλης κλίμακας και για αυτό είναι πλέον υποψήφιο πρωτόκολλο από το NIST για διάφορες κρυπτογραφικές διαδικασίες στην μετα-κβαντική κρυπτογραφία [15].

7 Βιβλιογραφία

- [1] Kontogeorgis, A., & Antoniadis, I. (2015). Finite Fields and Cryptography [Undergraduate textbook]. Kallipos, Open Academic Editions. <http://hdl.handle.net/11419/155>
- [2] W. Stein, 2008. Elementary Number Theory: Primes, Congruences, and Secrets: A Computational Approach. Undergraduate Texts in Mathematics. Springer New York. <https://books.google.gr/books?id=5hYd0yX4mrMC>.
- [3] SageMath, the Sage Mathematics Software System (Version 9.0.0), The Sage Developers, 2022, <https://www.sagemath.org>.
- [4] Python Software Foundation. Python Language Reference, version 3.10. Available at <http://www.python.org>.
- [5] S. Nakov, 2018, Practical Cryptography for Developers, Software University, <https://cryptobook.nakov.com/>.
- [6] S. Pohlig, M. Hellman, An improved algorithm for computing logarithms over $GF(p)$ and its cryptographic significance, 1977. <http://www.ee.stanford.edu/hellman/publications/28.pdf>
- [7] J. H. Silverman, The Arithmetic of Elliptic Curves, Springer-Verlag, 1986.
- [8] L. C. Washington, Elliptic Curves: Number Theory and Cryptography, 2008.
- [9] P. Novotney, Weak Curves In Elliptic Curve Cryptography, 2010. <https://wstein.org/edu/2010/414/projects/novotney.pdf>
- [10] Standards for Efficient Cryptography Group. SEC 1: Elliptic Curve Cryptography, Mar. 2009. Version 2.0. <http://www.secg.org/sec1-v2.pdf>.
- [11] Standards for Efficient Cryptography Group. SEC 2: Recommended Elliptic Curve Domain Parameters, Jan. 2010. Version 2.0. <http://www.secg.org/sec2-v2.pdf>.
- [12] P. W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. SIAM J. Computing, 26(5):1484–1509, 1997.
- [13] X. Πηλιχός, Κρυπτοσυστήματα και Αποδείξεις Μηδενικής Γνώσης στη Μεταθετική και Μη-Μεταθετική Κρυπτογραφία, 2018, Διπλωματική Εργασία.
- [14] National Institute of Standards and Technology. Recommended Elliptic Curves for Federal Government Use, Jul. 1999. csrc.nist.gov/encryption.
- [15] David Jao et al. Supersingular isogeny key encapsulation. NIST Round 1 Submissions for Post-Quantum Cryptography Standardization, November 30, 2017.
- [16] ISARA Corporation, Isogeny-Based Cryptography Tutorial, August 1, 2019, https://www.isara.com/downloads/crypto_tutorials/Intro-to-Iso-v3.pdf.
- [17] A. Semaev, Evaluation of Discrete Logarithms in a Group of p -Torsion Points of an Elliptic Curve in Characteristic p , Mathematics of Computation, Volume 67, Number 221, January 1998, Pages 353–356, S 0025-5718(98)00887-4.
- [18] A. J. Menezes, T. Okamoto, S. A. Vanstone, Reducing elliptic curve logarithms to a finite field, IEEE Trans., Info Theory, 39 (1993), pp.1639–1646
- [19] D. J. Bernstein and T. Lange. SafeCurves: choosing safe curves for elliptic-curve cryptography. <https://safecurves.cr.yp.to>, accessed 1 August 2022.
- [20] A. Miele, A. K. Lenstra, Efficient ephemeral elliptic curve cryptographic keys, 2015.