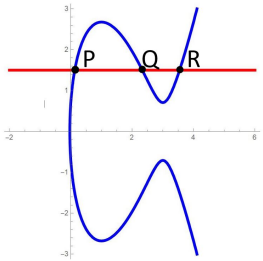
Crypto 11

Note: this material is not intended to replace the live lecture for students.

Contents



11.1 Computing with Elliptic curves

Elliptic curves have been studied for many years and there is an enormous amount of literature on the subject. In 1985, Neal Koblitz and V. S. Miller independently proposed using them for public-key cryptosystems [867,1095]. They did not invent a new cryptographic algorithm with elliptic curves over finite fields, but they implemented existing public-key algorithms, like Diffie-Hellman, using elliptic curves.

Elliptic curves are interesting because they provide a way of constructing "elements" and "rules of combining" that produce groups. These groups have enough familiar properties to build cryptographic algorithms, but they don't have certain properties that may facilitate cryptanalysis. For example, there is no good notion of "smooth" with elliptic curves. That is, there is no set of small elements in terms of which a random element has a good chance of being expressed by a simple algorithm. Hence, index calculus discrete logarithm algorithms do not work. See [1095] for more details.

Elliptic curves over the finite field GF(2^n) are particularly interesting. The arithmetic processors for the underlying field are easy to construct and are relatively simple to implement for n in the range of 130 to 200. They have the potential to provide faster public-key cryptosystems with smaller key sizes. Many public-key algorithms, like Diffie-Hellman, ElGamal, and Schnorr, can be implemented in elliptic curves over finite fields.

[Schneier15, page 480]

The introduction of *elliptic curve cryptography (ECC)* in 1985 revolutionized the way we do public-key cryptography. ECC is more powerful and efficient than alternatives like RSA and classical Diffie–Hellman (ECC with a 256-bit key is stronger than RSA with a 4096-bit key), but it's also more complex.

[Aumasson18, Chapter 12]

The reason we use elliptic curves is that the ECDLP has very good one-way characteristics.

[Paar10, page 251, Section 9.4]

Roughly speaking for n bits of security level ECC needs keys of length $2n$.

11.1.1 Plane curves

Given a field $\mathbb{K}$ the set of all pairs (x, y) with $x, y \in \mathbb{K}$ is the so called Cartesian plane denote by $\mathbb{K}^2$. A pair $P = (x_P, y_P)$ is called a point. For example if $\mathbb{K} = \mathbb{R}$ we are familiarized with points of the plane $\mathbb{R}^2$.

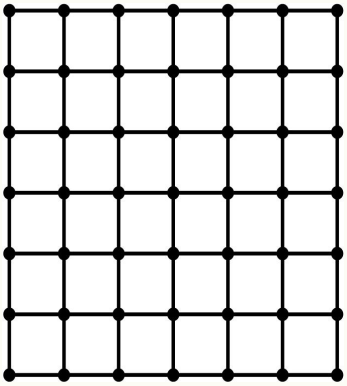
Exercise 11.1.1

Consider $\mathbb{K} = \mathbb{F}_q$. How many points there are in $\mathbb{K}^2$?

As in $\mathbb{R}^2$ an equation $ax + by + c = 0$ with $a, b, c \in \mathbb{K}$ (not all zero) defines a subset $L \subset \mathbb{K}^2$ of points. Namely, L is a so called **line** or straight line.

Exercise 11.1.2

Consider $\mathbb{K} = \mathbb{Z}_7$. Let $L : 2x + 5y + 1 = 0$ be a line of $\mathbb{Z}_7^2$. How many points contains L ? Circle the points of L:



Exercise 11.1.3

Let $P, Q, R \in \mathbb{K}^2$ be three points. Check that P, Q, R belongs to a line L if and only if the following holds:

$$\det \begin{bmatrix} x_P & x_Q & x_R \\ y_P & y_Q & y_R \\ 1 & 1 & 1 \end{bmatrix} = 0 \ .$$

Exercise 11.1.4

Let $P = (1, 6), Q = (4, 4) \in \mathbb{Z}_7^2$ be two points of the plane $\mathbb{Z}_7^2$. Find a equation $ax+by+c = 0$ of a line L through P and Q i.e. find $a, b, c \in \mathbb{Z}_7$ not all zero such that P, Q satisfies $ax + by + c = 0$.

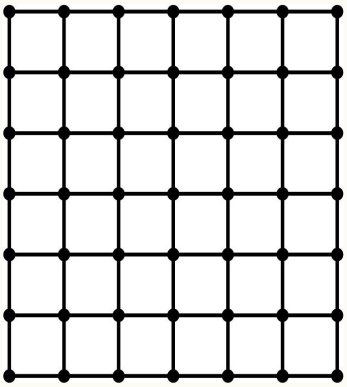
It is useful to remember the parametric line L:

$$P + tV$$

namely, P is a point of the plane $\mathbb{K}^2$, V is a vector of $\mathbb{K}^2$ and t is the parameter.

Exercise 11.1.5

Let $P = (1, 6) \in \mathbb{Z}_7^2$ be a point and $V = (3, 5)$ a vector of the plane $\mathbb{Z}_7^2$. Circle the points of L for $t = 0, 1, 2, 3, 4, 5, 6$:



11.1.2 Elliptic curves

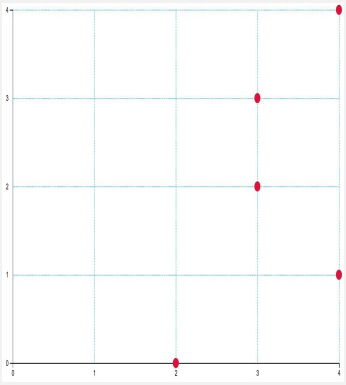
An elliptic curve E is a set consisting of pairs (x, y) that satisfies a special equation of the form:

$$f(x, y) = 0$$

where x, y belongs to a finite field GF(q).

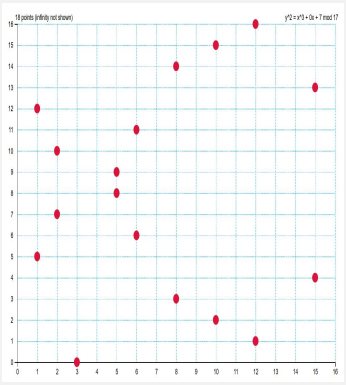
11.1.6 Example

Let E be defined by $f(x, y) = y^2 - x^3 - 7 = 0$ over GF(5). In this case E has



11.1.7 Example

Let E be defined by $f(x, y) = y^2 - x^3 - 7 = 0$ over GF(17). In this case E has



NOTE 11.1.8

Notice that in the above examples the expression $f(x, y) = y^2 - x^3 - 7$ is the same but the elliptic curves are different. To avoid such ambiguities the notation E(GF(q)) or E($\mathbb{F}_q$) is often used in the literature. That is to say, the goal of this notation is to remember that the coordinates x, y belong to the finite Galois field GF(q).

For q a prime number I will write:

$$f(x, y) = 0 \pmod q$$

which indicates that we are computing modulo q.

Notice that the number of points in E is less than $2 \cdot |\mathrm{GF}(q)|$. The number of points on E is usually denoted as |E| or #E.

The **important fact** about elliptic curves is that two points (x_1, y_1), (x_2, y_2) of E can be "added" to get a third point (x_3, y_3) of E:

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

The set E endowed with the operation + is commutative group. Moreover:

1) such "addition" of points on E is computationally feasible.

2) as consequence of 1) any cryptographic algorithm based on the discrete logarithm can be adapted to work on a elliptic curve.

3) usually the tradeoff key dimension vs. security level of adaptations as 2) are highly better than the originals.

11.1.3 Explicit formulae: Weierstrass' form

Most used elliptic curves are given in such form. Namely E is given as:

$$f(x,y) = y^2 - (x^3 + ax + b) = 0 \pmod q \tag{1}$$

Exercise 11.1.9

Check that $Q = (7,3)$ is a point of $\mathbf{E} : y^2 = x^3 + 7 \pmod{11}$.

11.1.10 Check that (x, y) belongs to E

```
''' checkEll(P,a,b,q) = boolean True or False .
        according P=(x,y) belongs to
       the elliptic curve y^2 = x^3 + ax + b  (mod q)
'''
def checkEll(P,a,b,q):
    if P=="0":return True
    else:
        (x,y)=P
        z = (y**2 - (x**3 + a*x + b))%q
        if z==0: return True
        else: return False
```

To compute the addition $(x_3, y_3) = (x_1, y_1)+(x_2, y_2)$ the formulae is:

$$(x_3, y_3) = \left(\lambda^2 - x_1 - x_2, -(\lambda x_3 + \nu)\right)$$

where $\lambda = \frac{y_2 - y_1}{x_2 - x_1} = \frac{x_1^2 + x_1 x_2 + x_2^2 + a}{y_1 + y_2}$ and $y_1 - \lambda x_1 = \nu$.

NOTE 11.1.11

If $y_1 = -y_2$ and $x_1 = x_2$ then (x_1, y_1)+(x_2, y_2) is **not given by the above formulae**. Instead, in this case the addition is by convention the neutral element **O** (the zero element) of the elliptic curve. Thus, to be precise an elliptic curve is the set of pairs E augmented with the identity element **O**. The **O** is also called neutral or zero element.

Summing up here is a common definition:

11.1.12 Elliptic curve

The elliptic curve E over GF(q) is the set of pairs (x, y), $x, y \in \mathrm{GF}(q)$ which fulfill

$$y^2 = x^3 + a \cdot x + b \pmod q$$

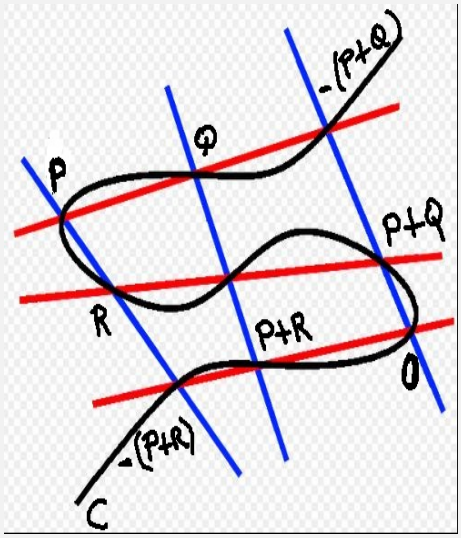
together with an extra point **O**, where

$$a, b \in \mathrm{GF}(q)$$

and the condition $4 \cdot a^3 + 27 \cdot b^2 \neq 0 \pmod q$.

The condition $4 \cdot a^3 + 27 \cdot b^2 \neq 0 \pmod q$ implies that + behaves as the usual addition i.e. $(E, +)$ is a commutative group.

To show that + satisfy **the associative property is non trivial !**



Exercise 11.1.13

Take $a = b = 0$ and $p = 5$. Check that

$$(0,0)\mathbf{+}(1,1) = (0,0)\,.$$

Conclusion: $\mathbf{+}$ does not behaves well. Indeed, otherwise $(1,1) = \mathbf{O}$ which is a contradiction since $\mathbf{O}$ is not a pair.

11.1.14 The Bitcoin curve Secp256k1

E' la curva di Koblitz E(GF(p)):

$$y^2 = x^3 + 7$$

dove $p = 2^{256} - 2^{32} - 977$.

Exercise 11.1.15

Check that $G = (x_G, y_G)$ where

x_G = 79BE667EF9DCBBAC55A06295CE870B07029BFCDB2DCE28D959F2815B16F81798

y_G = 483ADA7726A3C4655DA4FBFC0E1108A8FD17B448A68554199C47D08FFB10D4B8

is a point of Secp256k1. Such point G called Generator (or Primitive element) comes form page 9 of http://www.secg.org/sec2-v2.pdf.

Exercise 11.1.16

Compute the number of points on the elliptic curve E : $y^2 = x^3 + 2x + 2 \pmod{17}$. That is to say, compute the cardinal |E| of E.

NOTE 11.1.17

A point $P = (x, y)$ of an elliptic curve can be given in **compressed** or **uncompressed** form. Compressed means that just x is given. In such a case there are two values for y. So something else it is necessary to recover P. One possibility is to take $\min(y, -y)$. For example, take E : $y^2 = x^3 + 7 \pmod{11}$ and the compressed form of a point P equals to $x = 7$. Then for y there are two possibilities $y = 3$ and $y = 8$. Then $P = (7, 3)$.

The compress form starts with 03 or 02 whilst the uncompressed with 04:

G = 03 DB4FF10E C057E9AE 26B07D02 80B7F434 1DA5D1B1 EAE06C7D

G = 04 DB4FF10E C057E9AE 26B07D02 80B7F434 1DA5D1B1 EAE06C7D
9B2F2F6D 9C5628A7 844163D0 15BE8634 4082AA88 D95E2F9D

11.1.18 addition of $(x1, y2)$+$(x2, y2)$ on E

```
''' addpairs(P,Q,a,b,q) = (x3,y3) or "O"
        where (x3,y3) is the addition of (x1,y1) and (x2,y2) on
       the elliptic curve y^2 = x^3 + ax + b  (mod q).
'''
def addpairs(P,Q,a,b,q):
    (x1,y1)=P
    (x2,y2)=Q
    from inv import inv
    from checkEll import checkEll
    if checkEll(P,a,b,q)==True and checkEll(Q,a,b,q)==True:
        if x1%q==x2%q and y1%q==-y2%q:
          return "O"
        else:
            if x1%q != x2%q:
                lagreek = ((y2-y1)*inv(x2-x1,q))%q
                nugreek = (y1 - lagreek*x1)%q
                x3 = (lagreek**2 - x1 - x2)%q
                y3 = (-(lagreek*x3+nugreek))%q
                R = (x3,y3)
                if checkEll(R,a,b,q)==True: return R
                else: return "error computing addition x1!=x2"
            else:
                lagreek = ((x1**2 + x1*x2 + x2**2 + a)*inv(y1+y2,q))%q
                nugreek = (y1 - lagreek*x1)%q
                x3 = (lagreek**2 - x1 - x2)%q
                y3 = (-(lagreek*x3+nugreek))%q
                R = (x3,y3)
                if checkEll(R,a,b,q)==True: return R
                else: return "error computing addition x1==x2"
    else: return "error addition input"
```

Exercise 11.1.19

Compute $2 \cdot Q$ where $Q = (7, 3)$ and $\mathbf{E} : y^2 = x^3 + 7 \pmod{11}$.

http://ricerca.mat.uniroma3.it/users/codogni/CR510
https://cryptography.fandom.com/wiki/Elliptic_curve_cryptography
http://www.graui.de/code/elliptic2/

11.2 Encryption with Elliptic curves

Domain parameters

Following SEC 1 [SEC 1], elliptic curve domain parameters over $\mathbb{F}_p$ are a sextuple:

$$T = (p, a, b, G, n, h)$$

consisting of an integer p specifying the finite field $\mathbb{F}_p$, two elements $a, b \in \mathbb{F}_p$ specifying an elliptic curve $E(\mathbb{F}_p)$ defined by the equation:

$$E: \ y^2 \equiv x^3 + a.x + b \pmod{p},$$

a base point $G = (x_G, y_G)$ on $E(\mathbb{F}_p)$, a prime n which is the order of G, and an integer h which is the cofactor $h = \#E(\mathbb{F}_p)/n$.

Links to https://www.secg.org/
SEC 1
SEC 2

NOTE 11.2.1

Elliptic curves must be carefully chosen since not all of them are secure:
https://safecurves.cr.yp.to/index.html

Moreover in page 87 FIPS 186-4 there is an important Appendix with recommendations about curves to be used.

11.3 About the generator G

A point G of an elliptic curve E can be used to construct more points just by addition of G to itself

$$G{+}G$$

three times

$$G{+}G{+}G$$

or n-times

$$n \cdot G = G{+}G{+}\cdots{+}G\,.$$

So the notation $n \cdot G$ means that G is added to itself n-times.

11.3.1 Example

Let $G = (5, 1)$ be a point of $\mathrm{E} : y^2 = x^3 + 2x + 2 \pmod{17}$ (check it!).
Here you see all its multiples:

$$\begin{aligned}
1 \cdot G &= (5,\ 1) \\
2 \cdot G &= (6,\ 3) \\
3 \cdot G &= (10,\ 6) \\
4 \cdot G &= (3,\ 1) \\
5 \cdot G &= (9,\ 16) \\
6 \cdot G &= (16,\ 13) \\
7 \cdot G &= (0,\ 6) \\
8 \cdot G &= (13,\ 7) \\
9 \cdot G &= (7,\ 6) \\
10 \cdot G &= (7,\ 11) \\
11 \cdot G &= (13,\ 10) \\
12 \cdot G &= (0,\ 11) \\
13 \cdot G &= (16,\ 4) \\
14 \cdot G &= (9,\ 1) \\
15 \cdot G &= (3,\ 16) \\
16 \cdot G &= (10,\ 11) \\
17 \cdot G &= (6,\ 14) \\
18 \cdot G &= (5,\ 16) \\
19 \cdot G &= \mathbf{O}
\end{aligned}$$

Exercise 11.3.2

Let E be the elliptic curve of the example above. Find:

- $P \in \mathrm{E}$ such that $18 \cdot P = (0, 6)$.
- $Q \in \mathrm{E}$ such that $3 \cdot Q = (3, 16)$.

11.4 Some famous Elliptic curves

11.4.1 Koblitz curves

Are the following curves $\mathrm{E}_a(\mathrm{GF}(2^m))$, $a \in \{0, 1\}$:

$$\mathrm{E}_0 : y^2 + xy = x^3 + 1$$

$$\mathrm{E}_1 : y^2 + xy = x^3 + x^2 + 1$$

Notice that the above curves are defined on a Galois Field $\mathrm{GF}(2^m)$ where $q = 2^m$ is not a prime number.

11.4.2 The elliptic curve P-160

Is the $\mathrm{E}(\mathrm{GF}(p))$ curve:

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

where $p = 2^{160} - 229233$ is a prime number of 160 bits.

11.4.3 Curve25519

Is the curve $\mathrm{E}(\mathrm{GF}(p))$:

$$y^2 = x^3 + 486662x^2 + x$$

dove $p = 2^{255} - 19$ is prime.

Curve25519: new Diffie-Hellman speed records

NOTE 11.4.4

Some times are used elliptic curves in which the equation $f(x, y) = 0$ is not a cubic e.g. Edwards curve:

$$x^2 + y^2 = 1 + dx^2y^2$$

See also https://en.wikipedia.org/wiki/EdDSA#Ed25519 o The first 10 years of Curve25519

Monero use the Edwards curve: https://web.archive.org/web/20190501100100/https://lab.getmonero.org/pubs/MRL-0003.pdf

11.5 Bibliography

Books I used to prepare this note:

[Aumasson18] Jean-Philippe Aumasson, *Serious Cryptography: A Practical Introduction to Modern Encryption*, No Starch Press, 2018.

[KatLin15] Jonathan Katz; Yehuda Lindell, *Introduction to Modern Cryptography* Second Edition, Chapman & Hall/CRC, Taylor & Francis Group, 2015.

[Paar10] Paar, Christof, Pelzl, Jan, *Understanding Cryptography, A Textbook for Students and Practitioners*, Springer-Verlag, 2010.

[Schneier15] Bruce Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C* ,Wiley; 20th Anniversary edition,2015.

Here a list of papers:

and some interesting links:

http://www.secg.org/sec2-v2.pdf

https://bitcoin.stackexchange.com/questions/21907/what-does-the-curve-used-in-bitcoin-secp256k1-look-like

http://ricerca.mat.uniroma3.it/users/codogni/CR510

https://cryptography.fandom.com/wiki/Elliptic_curve_cryptography

http://www.graui.de/code/elliptic2/

https://tools.ietf.org/id/draft-jivsov-ecc-compact-05.html

https://pypi.org/project/fastecdsa/

https://medium.com/@schaetzcornelius/learn-how-to-code-elliptic-curve-cryp

http://www.nicolascourtois.com/papers/ga18/Lecture%20-%20DLOG%20and%20Factoring%20Algorithms.pdf