

IA174: Elliptic curve cryptography (ECC)

Special thanks to Vlád'a Sedláček for the initial development of the ECC lecture and these slides.

History of elliptic curves

Elliptic curves have been with us for quite a while:

- First appeared by Diophantus's *Arithmetica* in the third century A.D.; later studied by many mathematicians thanks to the rich algebraic and geometric structure; even used in the proof of Fermat's Last Theorem.
- The name comes from their appearance in certain integrals measuring the length of the arc of an ellipse.

History of elliptic curves

Elliptic curves have been with us for quite a while:

- First appeared by Diophantus's *Arithmetica* in the third century A.D.; later studied by many mathematicians thanks to the rich algebraic and geometric structure; even used in the proof of Fermat's Last Theorem.
- The name comes from their appearance in certain integrals measuring the length of the arc of an ellipse.
- First practical application in 1984 - Lenstra's factoring method based on elliptic curves (still useful today).
- In 1985, Koblitz and Miller propose ECC: very efficient compared to other asymmetric cryptography primitives - small keys and fast computations; great for small devices.

Fields

$$5.3 \quad -5.3 \quad (\mathbb{R}, +, \cdot) \quad 1 \cdot x = \frac{1}{x}$$

Definition 1: Field

A *field* is a tuple $(F, +, \cdot)$, such that:

- $(F, +)$ is an abelian group (called *additive*) with neutral element 0,
- $(F \setminus \{0\}, \cdot)$ is an abelian group (called *multiplicative*). *with neutral element 1.*
- multiplication **distributes** over addition: for all $a, b, c \in F$, we have
 $(a + b) \cdot c = a \cdot c + b \cdot c.$ *(D I S T)*

$$(\mathbb{R}, +, \cdot)$$

Definition 1: Field

A *field* is a tuple $(F, +, \cdot)$, such that:

- $(F, +)$ is an abelian group (called *additive*) with neutral element 0,
- $(F \setminus \{0\}, \cdot)$ is an abelian group (called *multiplicative*).
- multiplication **distributes** over addition: for all $a, b, c \in F$, we have $(a + b) \cdot c = a \cdot c + b \cdot c$.

Common infinite fields include $\mathbb{Q}, \mathbb{R}, \mathbb{C}$.

How about finite fields?

Field $\rightarrow (\mathbb{Z}_p, +, \cdot)$ $(\mathbb{Z}_p, +)$
 $(\mathbb{Z}_p^\times, \cdot)$

Finite fields

In the previous lectures, we encountered **Galois fields**, which are finite. E.g. the field of **bytes** $GF(2^8)$ with xor as addition and the multiplication defined via polynomials.

Finite fields

In the previous lectures, we encountered **Galois fields**, which are finite. E.g. the field of **bytes** $GF(2^8)$ with xor as addition and the multiplication defined via polynomials.

The polynomial-based construction can be used to define a field $GF(q)$ for any **prime power** $q = p^k$. All finite fields are of this form.

Finite fields

In the previous lectures, we encountered **Galois fields**, which are finite. E.g. the field of **bytes** $GF(2^8)$ with xor as addition and the multiplication defined via polynomials.

The polynomial-based construction can be used to define a field $GF(q)$ for any **prime power** $q = p^k$. All finite fields are of this form.

Luckily, when dealing with elliptic curves, we will be dealing only with fields $GF(p)$ for a p prime. These are **isomorphic** to $\mathbb{F}_p = (\mathbb{Z}_p, +, \cdot)$.

What is an elliptic curve?

Definition 2: Elliptic curve, basic version

Let F be any field and $a, b \in F$ such that $4a^3 + 27b^2 \neq 0$. An elliptic curve over F is the set of points $(x, y) \in F^2$ satisfying the Weierstrass equation

$$y^2 = x^3 + a \cdot x + b, \quad y = \pm \sqrt{\dots}$$

together with a special "point at infinity" \mathcal{O} .

In cryptography, we usually take $F = \mathbb{F}_p$ with $p \geq 5$ a prime number.

$$\mathbb{F}_p$$

What is an elliptic curve?

Definition 2: Elliptic curve, basic version

Let F be **any** field and $a, b \in F$ such that $4a^3 + 27b^2 \neq 0$. An *elliptic curve* over F is the set of points $(x, y) \in F^2$ satisfying the Weierstrass equation

$$y^2 = x^3 + a \cdot x + \underline{b},$$

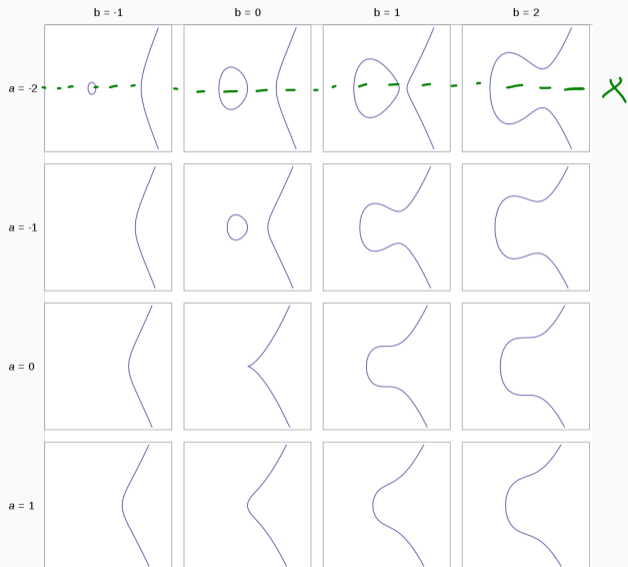
together with a special “point at infinity” \mathcal{O} .

In cryptography, we usually take $F = \mathbb{F}_p$ with $p \geq 5$ a prime number.

Innocent pedagogic lies

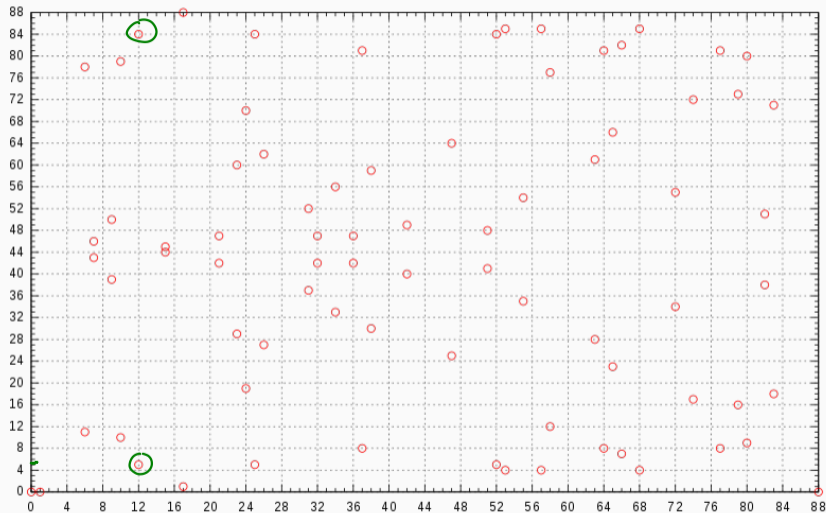
For the purposes of this lecture, **green text** might not always be true, but it is “morally true”.

Elliptic curves over \mathbb{R}



Elliptic curves over \mathbb{F}_p

$$89 - 5 = 84$$



An elliptic curve over \mathbb{F}_{89} given by $y^2 = x^3 - x$.

$$\mathbb{F}_p = (\mathbb{Z}_p, +, \cdot)$$

Groups from elliptic curves

$$O, \underbrace{P + \dots + P}_k = k \cdot P$$

Each elliptic curve E defines a **group** $(E, +)$ (typically written in additive notation) where the group operation $+$ is defined as follows: if $E : y^2 = x^3 + ax + b$, then $(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$ is defined as follows:

- If $x_1 \neq x_2$, then we compute

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

$$k \cdot m + c$$

and put

$$x_3 = m^2 - x_1 - x_2, \quad y_3 = -m(x_3 - x_1) - y_1.$$

- If $(x_1, y_1) = (x_2, y_2)$ and $y_1 \neq 0$ then we compute

$$m = \frac{3x_1^2 + a}{2y_1}$$

and put

$$x_3 = m^2 - 2x_1, \quad y_3 = -m(x_3 - x_1) - y_1.$$

Groups from elliptic curves II

- If $x_1 = x_2$ and either $y_1 \neq y_2$ or $y_1 = y_2 = 0$, then

$$(x_1, y_1) + (x_2, y_2) = \mathcal{O}$$

- \mathcal{O} is the neutral element of the group, i.e. $P + \mathcal{O} = \mathcal{O} + P = P$ for any point P of the curve.

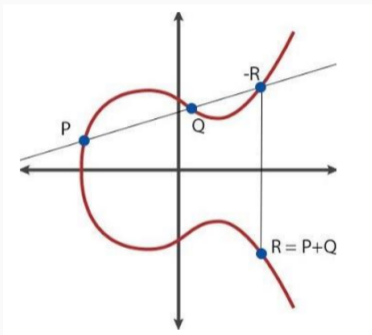
These addition formulas work over **any** field. In practice, we often use other formulas, where we do not have to distinguish the cases

The EC group is always cyclic or bicyclic (a product of two cyclic groups).

The group law explained in geometrical terms

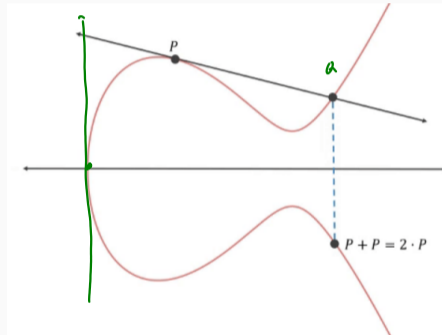
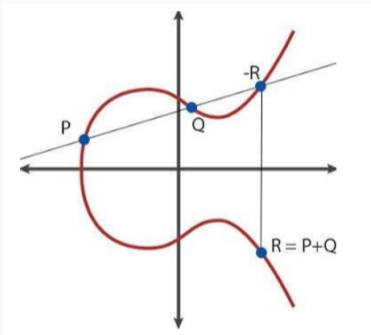
Fact: Every line that intersects an elliptic curve in at least two points intersects it in exactly three points (if counted in the right way).

$P + Q$



The group law explained in geometrical terms

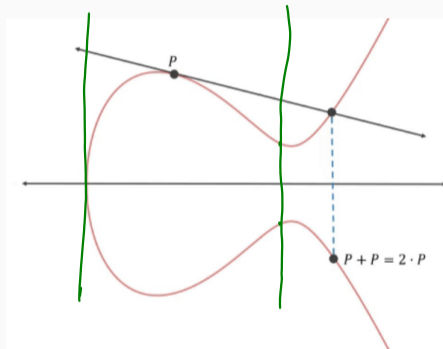
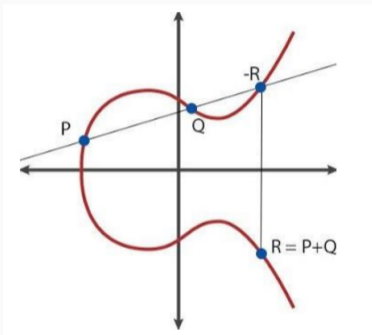
Fact: Every line that intersects an elliptic curve in at least two points intersects it in exactly three points (if counted in the right way).



If $P = Q$, we have a tangent instead of a secant.

The group law explained in geometrical terms

Fact: Every line that intersects an elliptic curve in at least two points intersects it in exactly three points (if counted in the right way).



If $P = Q$, we have a tangent instead of a secant.

If the line through P and Q is vertical, then $P + Q = \mathcal{O}$ (\mathcal{O} is the “vertical direction”).

Discrete logarithm over elliptic curves

We customarily use **additive** notation when working with EC groups. That is, a k -fold application of the group operation $+$ is denoted as

$$k \cdot P = \underbrace{P + \dots + P}_{k \text{ times}}$$

The **discrete logarithm** problem over elliptic groups can then be re-formulated as follows: Given a group $(E, +)$ defined by an elliptic curve, some point G on the curve (generator of a cyclic sub-group $\langle G \rangle$), and a point $P = k \cdot G$, find k .

$$\underbrace{G + \dots + G}_{k \text{-times}}$$

How to choose the right elliptic curve?

... is a question we will be dealing with for much of the remainder of this lesson. All elliptic curves are not created equal.

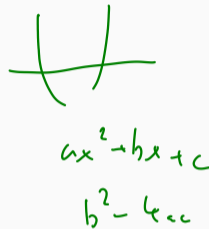
The discriminant

...allows us to recognize some bad-behaved curves.

Let $E : y^2 = f(x)$, where $f(x) := x^3 + ax + b$. Then

$$\Delta(f) := -(4a^3 + 27b^2) \neq 0$$

is the *discriminant* of f .



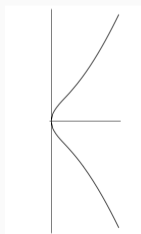
The discriminant

... allows us to recognize some bad-behaved curves.

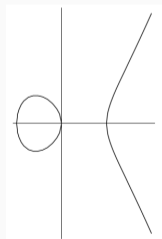
Let $E : y^2 = f(x)$, where $f(x) := x^3 + ax + b$. Then

$$\Delta(f) := -(4a^3 + 27b^2)$$

is the *discriminant* of f . Over \mathbb{R} , the number of components of E depends on the sign of $\Delta(f)$.



$$y^2 = x^3 - x, \Delta(f) > 0$$

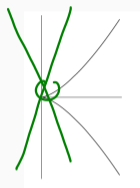


$$y^2 = x^3 + x, \Delta(f) < 0$$

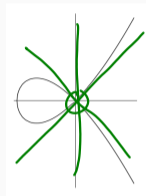
We have $\Delta(f) = 0$ iff f has a multiple root. Then E is not an elliptic curve, but a singular one. ^{13/22}

Singular curves

The group law fails for singular curves, as the tangents cannot always be defined.



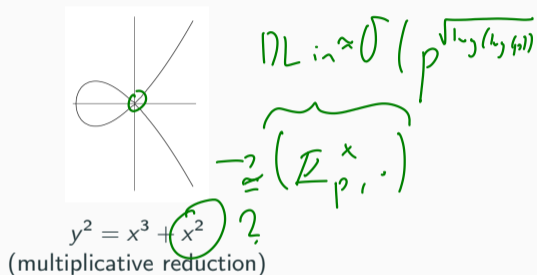
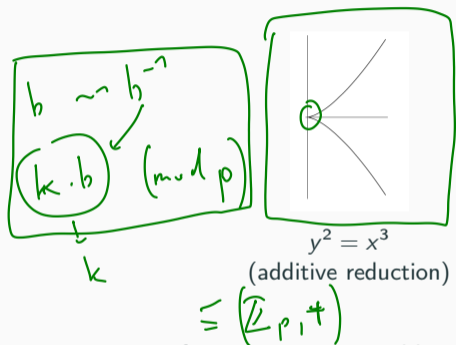
$y^2 = x^3$
(additive reduction)



$y^2 = x^3 + x^2$
(multiplicative reduction)

Singular curves

The group law fails for singular curves, as the tangents cannot always be defined.



However, if we remove the problematic point, we get a group isomorphic to either the additive or the **multiplicative group** of the underlying field. In the first case, we already saw that the DLP in $(\mathbb{F}_p, +)$ is easy. The second case is the classical finite field crypto in (\mathbb{F}_p^*, \cdot) . Since these are just degenerate cases, we can view ECC as a direct generalization.

Group size

The curve E and the point G should be such that $|\langle G \rangle|$ obeys the standard requirements on the hardness of the discrete logarithm problem:

- $|\langle G \rangle|$ should be sufficiently large, to prevent DL computation by bruteforcing or, e.g., the **baby-step giant-step** algorithm.
- $|\langle G \rangle|$ should not be smooth, this is to defend against DL algorithms whose runtime is dominated by the term exponential in the bitsize of the largest prime factor of $|G|$ (**Pohlig-Hellman** algorithm, **Pollard's ρ** algorithm for discrete logarithm)

Group size

The curve E and the point G should be such that $|\langle G \rangle|$ obeys the standard requirements on the hardness of the discrete logarithm problem:

- $|\langle G \rangle|$ should be sufficiently large, to prevent DL computation by bruteforcing or, e.g., the **baby-step giant-step** algorithm.
- $|\langle G \rangle|$ should not be smooth, this is to defend against DL algorithms whose runtime is dominated by the term exponential in the bitsize of the largest prime factor of $|G|$ (**Pohlig-Hellman** algorithm, **Pollard's ρ** algorithm for discrete logarithm)

Theorem 1: The Hasse-Weil bound

For any elliptic curve E over \mathbb{F}_p , we have

$$|E| \approx \mathcal{O}(p)$$

$$p + 1 - 2\sqrt{p} \leq |E| \leq p + 1 + 2\sqrt{p}.$$

In practice, we use polynomial-time Schoof's algorithm to compute the group order.

Known attacks on the ECDLP:

- Anomalous curves ($|E| = p$, underlying field \mathbb{F}_p) admit Smart's attack - an additive transfer of the ECDLP into the DL in $(\mathbb{Z}_p, +)$.

Fast DL algorithms for elliptic curves

Known attacks on the ECDLP:

- Anomalous curves ($|E| = p$, underlying field \mathbb{F}_p) admit Smart's attack - an additive transfer of the ECDLP into the DL in $(\mathbb{Z}_p, +)$.
- Supersingular curves ($|E| = p + 1$, underlying field \mathbb{F}_p) and more generally curves with a low *embedding degree* r (**embedding degree** = the multiplicative order of p **modulo** $|E|$) admit the MOV attack - a multiplicative transfer of the ECDLP into the DLP in the multiplicative group $(\mathbb{Z}_{p^r}^\times, \cdot)$.

$$p \equiv -1 \pmod{p+1}$$

$$p^2 \equiv 1 \pmod{p+1}$$

$$p^r \equiv 1 \pmod{|E|}$$

$$p^r \equiv 1 \pmod{p+1}$$

Known attacks on the ECDLP:

- Anomalous curves ($|E| = p$, underlying field \mathbb{F}_p) admit Smart's attack - an additive transfer of the ECDLP into the DL in $(\mathbb{Z}_p, +)$.
- Supersingular curves ($|E| = p + 1$, underlying field \mathbb{F}_p) and more generally curves with a low *embedding degree* r (**embedding degree** = the multiplicative order of p **modulo** $|E|$) admit the MOV attack - a multiplicative transfer of the ECDLP into the DLP in the multiplicative group $(\mathbb{Z}_{p^r}^*, \cdot)$.
- That's it, no other **known** subexponential ECDLP-specific attacks!

Known attacks on the ECDLP:

- Anomalous curves ($|E| = p$, underlying field \mathbb{F}_p) admit Smart's attack - an additive transfer of the ECDLP into the DL in $(\mathbb{Z}_p, +)$.
- Supersingular curves ($|E| = p + 1$, underlying field \mathbb{F}_p) and more generally curves with a low *embedding degree* r (**embedding degree** = the multiplicative order of p **modulo** $|E|$) admit the MOV attack - a multiplicative transfer of the ECDLP into the DLP in the multiplicative group $(\mathbb{Z}_{p^r}^*, \cdot)$.
- That's it, no other **known** subexponential ECDLP-specific attacks!

But remember: in practice, ECC security is not the same thing as ECDLP security...

Implementation choices

In real-life scenarios, an implementation has many choices to make:

- curve choice
- scalar multiplication algorithm choice,
- addition formula choice,
- point representation choice,
- finite field arithmetic implementation choice,
- random number generator choice (recall the Sony Playstation ECDSA incident),
- the choice of running certain security validations,...

Getting these choices right is very hard in practice; many of them might open you up to unexpected attacks. Do not try this at home!

Implementation-specific attacks

Examples of attacks against missing security validations:

- the small subgroup attack - applies when the curve has a composite order; mitigated by cofactor validation; can be combined with the next two attacks,

Implementation-specific attacks

Examples of attacks against missing security validations:

- the small subgroup attack - applies when the curve has a composite order; mitigated by cofactor validation; can be combined with the next two attacks,

$$|G| = \underbrace{(n_1)}_{\substack{3, 5, 7, \dots \\ \text{huge}}} \cdot n_2$$

$$|\langle b \rangle| = (n_1)$$

$$\begin{array}{c} \rightarrow P \\ (k \cdot P) \\ \leftarrow \end{array}$$

$\frac{1}{3}$

$$k \pmod{3}$$

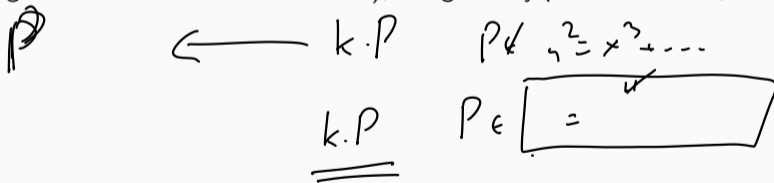
$$k \pmod{5}$$

$$k \pmod{2}$$

Implementation-specific attacks

Examples of attacks against missing security validations:

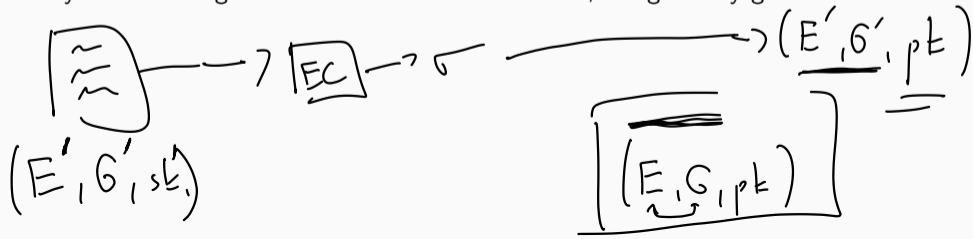
- the small subgroup attack - applies when the curve has a composite order; mitigated by cofactor validation; can be combined with the next two attacks,
- the invalid-curve attack - applies when the attacker is allowed to compute on another curve (typically only differing in the Weierstrass b coefficient); mitigated by point validation,



Implementation-specific attacks

Examples of attacks against missing security validations:

- the small subgroup attack - applies when the curve has a composite order; mitigated by cofactor validation; can be combined with the next two attacks,
- the invalid-curve attack - applies when the attacker is allowed to compute on another curve (typically only differing in the Weierstrass b coefficient); mitigated by point validation,
- the recent Curveball attack (also known as ChainOfFools) - applies if the attacker can actually solve a self-generated instance of the ECDLP; mitigated by generator validation...



Implementation-specific attacks

Examples of attacks against missing security validations:

- the small subgroup attack - applies when the curve has a composite order; mitigated by cofactor validation; can be combined with the next two attacks,
- the invalid-curve attack - applies when the attacker is allowed to compute on another curve (typically only differing in the Weierstrass b coefficient); mitigated by point validation,
- the recent Curveball attack (also known as ChainOfFools) - applies if the attacker can actually solve a self-generated instance of the ECDLP; mitigated by generator validation...

And even if you manage to avoid all of these, there is a huge number of side channel attacks (using time, electricity, EM radiation, caches, exceptional arithmetic cases,...).

Curve standardization and trust

In practice, you want to use some **standardized** curves. But this elicits further questions:

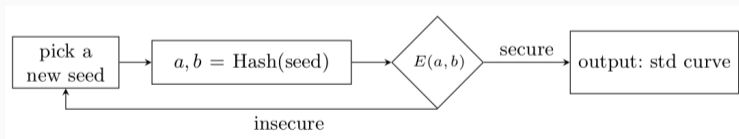
- Who gets to choose them and how?
- Is the process transparent enough?
- How do we know there is no hidden weakness or even a backdoor?

Curve standardization and trust

In practice, you want to use some **standardized** curves. But this elicits further questions:

- Who gets to choose them and how?
- Is the process transparent enough?
- How do we know there is no hidden weakness or even a backdoor?

Currently, most of ECC on the Internet still uses the P-256 curve designed by the NSA and standardized by NIST. It follows the ANSI X9.62 standard, but the choice of the seed was never explained.



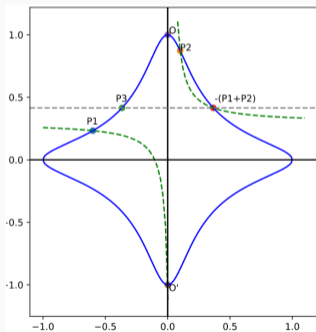
A simplified template for generating verifiably pseudorandom curves over fixed \mathbb{F}_p .

This causes some trust issues.

Modern curves

Two curves that are gaining popularity lately are Curve25519 and its sibling Ed25519, both designed by Daniel Bernstein. They differ from most of the standard curves in several ways:

- they offer better performance and misuse-resistance, leading to higher practical security;
- the generation process is more transparent;



The key messages of this lecture:

- ECC is a powerful and efficient way to convert deep algebra and number theory into strong cryptography.
- ECC offers better performance than RSA or discrete logarithm over plain \mathbb{Z}_p^\times (shorter keys required to offer the same security level).
- One must be very careful with ECC implementations, there are many pitfalls.
- Curve standardization introduces trust issues.

AES - 128 3072 bits
 2 256 bits

95% of people cannot solve this!

$$\frac{\text{🍎}}{\text{🍌} + \text{🍍}} + \frac{\text{🍌}}{\text{🍎} + \text{🍍}} + \frac{\text{🍍}}{\text{🍎} + \text{🍌}} = 4$$

Can you find positive whole values

for 🍎, 🍌, and 🍍?

95% of people cannot solve this!

$$\frac{\text{🍎}}{\text{🍌} + \text{🍍}} + \frac{\text{🍌}}{\text{🍎} + \text{🍍}} + \frac{\text{🍍}}{\text{🍎} + \text{🍌}} = 4$$




Can you find positive whole values

for 🍎, 🍌, and 🍍?

Just an elliptic curve in disguise!

95% of people cannot solve this!

$$\frac{\text{apple}}{\text{banana} + \text{pineapple}} + \frac{\text{banana}}{\text{apple} + \text{pineapple}} + \frac{\text{pineapple}}{\text{apple} + \text{banana}} = 4$$

**Can you find positive whole values
for , , and ?**

Just an elliptic curve in disguise! The smallest solution:

apple = 154476802108746166441951315019919837485664325669565431700026634898253202035277999,
banana = 36875131794129999827197811565225474825492979968971970996283137471637224634055579,
pineapple = 4373612677928697257861252602371390152816537558161613618621437993378423467772036.