

# Lesson 1

## Course Introduction

## Course Plan

### Week 1 Introduction

Overview / Maths & Cryptography introduction

ZKP Theory / Zokrates

Use Cases of ZKPs / L2

Introduction to Starknet / Cairo

### Week 2 - Starknet and Cairo / Noir

Cairo

DeFi / Aztec

Noir

MINA

### Week 3 Mina / DeFi / Circom

MINA / zkApps

zkEVM Solutions

Risc Zero / Circom

zkSNARKS Theory

### Week 4 - Theory and Cryptography

zkML / PLONK

STARK Theory

Cryptographic alternatives

Research and review

# Practical Details

The format will usually be 45 mins of theory followed by 45 mins practical

## Homeworks

There will be a homework for each lesson, but you do not need to submit the homeworks.

## About Extropy

**ABOUT US**  
Extropy.io was founded 2015 by Laurence Kirk in Oxford to provide consultancy services in Distributed Ledger Technology. Laurence is also the founder of the Oxford Blockchain Society.

**INNOVATE.  
QUALITY.  
CUTTING EDGE.**

**CONTACT US**  
Oxford Centre for Innovation, New Road, Oxford, OX1 1BY, UK  
[www.extropy.io](http://www.extropy.io)  
+44 (0)1865 261 424

Providing Blockchain solutions  
DApp development and customised blockchains  
Security Audits

**EXTROPY.IO**  
CONSULTANCY IN DISTRIBUTED LEDGER TECHNOLOGY

### Free Developer Workshops

- Basic
- Enterprise
- Advanced EVM
- Zero Knowledge Proofs

### Business Workshops

Website :

<https://extropy.io>

Email :

[info@extropy.io](mailto:info@extropy.io)

Twitter : [@extropy](https://twitter.com/@extropy)

## Social Media :

- X : [@Extropy](https://twitter.com/@Extropy)
- Warpcast: [@Extropy](https://warpcast.net/@Extropy)
- Discord [Invite](https://discord.com/invite/Extropy)

For Q&A use [Sli.do](https://sli.do)

## General Resources

[Zero Knowledge Podcast](#)

[Awesome ZK Repo](#)

[Alex Pinto's Blog](#)

# **Introduction**

## **Context**

"Human dignity demands that personal information, like medical and forensic data, be hidden from the public. But veils of secrecy designed to preserve privacy may also be abused to cover up lies and deceit by institutions entrusted with Data, unjustly harming citizens and eroding trust in central institutions." - Starkware

"ZK gives out similar vibe as ML. More and more people just mention ZK as a magic solution that fixes everything with no context of its current limitation." - 0xMisaka

---

# Introductory Maths

## Numbers

The set of Integers is denoted by  $\mathbb{Z}$  e.g.  $\{\dots, -4, -3, -2, -1, 0, 1, 2, 3, 4, \dots\}$

The set of Rational Numbers is denoted by  $\mathbb{Q}$  e.g.  $\{\dots, 1, \frac{3}{2}, 2, \frac{22}{7}, \dots\}$

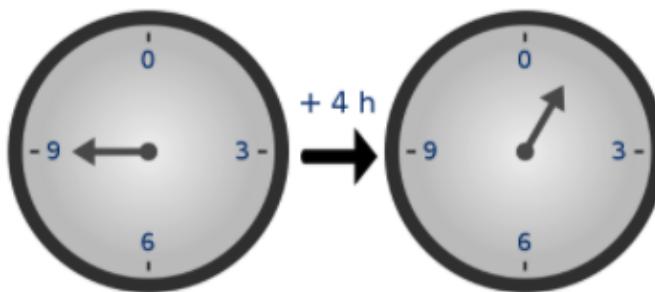
The set of Real Numbers is denoted by  $\mathbb{R}$  e.g.  $\{2, -4, 613, \pi, \sqrt{2}, \dots\}$

Fields are denoted by  $\mathbb{F}$ , if they are a finite field or  $\mathbb{K}$  for a field of real or complex numbers we also use  $\mathbb{Z}_p^*$  to represent a finite field of integers mod prime  $p$  with multiplicative inverses.

We use finite fields for cryptography, because elements have “short”, exact representations and useful properties.

## Modular Arithmetic

See this [introduction](#)



Because of how the numbers "wrap around", modular arithmetic is sometimes called "clock math"

When we write  $n \bmod k$  we mean simply the remainder when  $n$  is divided by  $k$ . Thus

$$25 \bmod 3 = 1$$

$$15 \bmod 4 = 3$$

The remainder should be positive.



# Group Theory

Simply put a group is a set of elements {a,b,c,...} plus a binary operation, here we represent this as

- To be considered a group this combination needs to have certain properties

## 1. Closure

For all a, b in G, the result of the operation,  $a \cdot b$ , is also in G

## 2. Associativity

For all a, b and c in G,  $(a \cdot b) \cdot c = a \cdot (b \cdot c)$

## 3. Identity element

There exists an element e in G such that, for every element a in G, the equation  $e \cdot a = a \cdot e = a$  holds. Such an element is unique and thus one speaks of the identity element.

## 4. Inverse element

For each a in G, there exists an element b in G, commonly denoted  $a^{-1}$  (or  $-a$ , if the operation is denoted "+"), such that  $a \cdot b = b \cdot a = e$ , where e is the identity element.

## Sub groups

If a subset of the elements in a group also satisfies the group properties, then that is a subgroup of the original group.

## Cyclic groups and generators

A finite group can be cyclic. That means it has a generator element. If you start at any point and then apply the group operation with the generator as argument a certain number of times, you go around the whole group and end in the same place,

## Finding an inverse

From Fermat's little theorem

$$a^{-1} \equiv a^{p-2} \pmod{p}$$

Let  $p = 7$  and  $a = 2$ . We can compute the inverse of a as:

$$a^{p-2} = 2^5 = 32 \equiv 4 \pmod{7}.$$

This is easy to verify:  $2 \times 4 \equiv 1 \pmod{7}$ .

## Equivalence classes

Since

$$6 \pmod{7} = 6$$

$$13 \pmod{7} = 6$$

$$20 \pmod{7} = 6$$

...

we can say that 6, 13, 20 ... form an equivalence class  
more formally

modular arithmetic partitions the integers into  $N$  equivalence classes, each of the form  $i + kN \mid k \in \mathbb{Z}$  for some  $i$  between 0 and  $N - 1$ .

Thus if we are trying to solve the equation

$$x \bmod 7 = 6$$

$x$  could be 6, 13, 20 ...

This gives us the basis for a one way function.

---

# Fields

A field is a set of say Integers together with two operations called addition and multiplication. One example of a field is the Real Numbers under addition and multiplication, another is a set of Integers mod a prime number with addition and multiplication. The field operations are required to satisfy the following field axioms. In these axioms, a, b and c are arbitrary elements of the field  $\mathbb{F}$ .

1. Associativity of addition and multiplication:  $a + (b + c) = (a + b) + c$  and  $a \cdot (b \cdot c) = (a \cdot b) \cdot c$ .
2. Commutativity of addition and multiplication:  $a + b = b + a$  and  $a \cdot b = b \cdot a$ .
3. Additive and multiplicative identity: there exist two different elements 0 and 1 in  $\mathbb{F}$  such that  $a + 0 = a$  and  $a \cdot 1 = a$ .
4. Additive inverses: for every  $a$  in  $\mathbb{F}$ , there exists an element in  $\mathbb{F}$ , denoted  $-a$ , called the additive inverse of  $a$ , such that  $a + (-a) = 0$ .
5. Multiplicative inverses: for every  $a \neq 0$  in  $\mathbb{F}$ , there exists an element in  $\mathbb{F}$ , denoted by  $a^{-1}$ , called the multiplicative inverse of  $a$ , such that  $a \cdot a^{-1} = 1$ .
6. Distributivity of multiplication over addition:  $a \cdot (b + c) = (a \cdot b) + (a \cdot c)$ .

## Finite fields and generators

A finite field is a field with a finite set of elements, such as the set of integers mod  $p$  where  $p$  is a prime.

To try out operations on finite fields, see <https://asecuritysite.com/encryption/finite>

The **order** of the field is the number of elements in the field's set.

For a finite field the order must be either

- prime ( a prime field)
- or
- the power of a prime (an extension field)

An element can be represented as an integer greater or equal than 0 and less than the field's order:  $\{0, 1, \dots, p-1\}$  in a simple field.

Every finite field has a generator. A generator is capable of generating all of the elements in the set by exponentiating the generator .

So for generator  $g$  we can take  $g^0, g^1, g^2$  and eventually this will give us all elements in the group

For example. taking the set of integers and prime  $p = 5$ , we get the group  $\mathbb{Z}_5^* = \{0, 1, 2, 3, 4\}$ .

In the group  $\mathbb{Z}_5^*$ , operations are carried out modulo 5; hence, we don't have  $3 \times 4 = 12$  but instead have  $3 \times 4 = 2$ , because  $12 \bmod 5 = 2$ .

$\mathbb{Z}_5^*$  is cyclic and has two generators, 2 and 3, because  $2^1 = 2, 2^2 = 4, 2^3 = 3, 2^4 = 1$ , and  $3^1 = 3, 3^2 = 4, 3^3 = 2, 3^4 = 1$

In a finite field of order  $q$ , the polynomial  $X^q - X$  has all  $q$  elements of the finite field as roots.

# Group Homomorphisms

A homomorphism is a map between two algebraic structures of the same type, that preserves the operations of the structures.

This means a map  $f : A \rightarrow B$  between two groups A, B equipped with the same structure such that,

if  $\cdot$  is an operation of the structure (here a binary operation), then

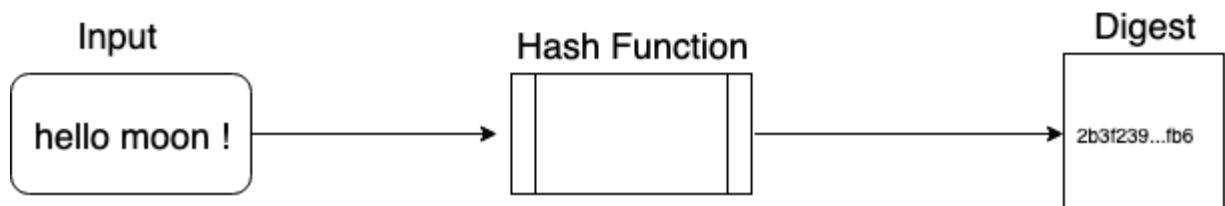
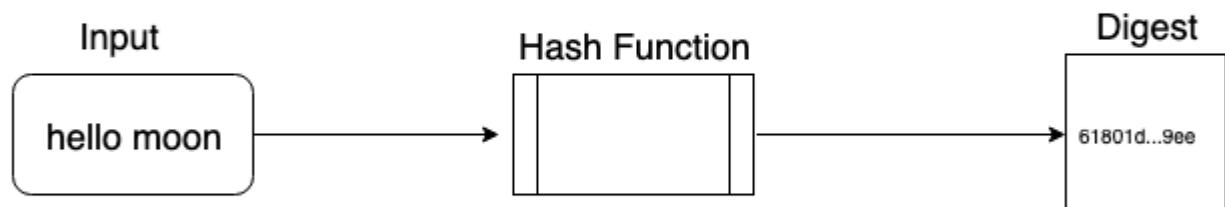
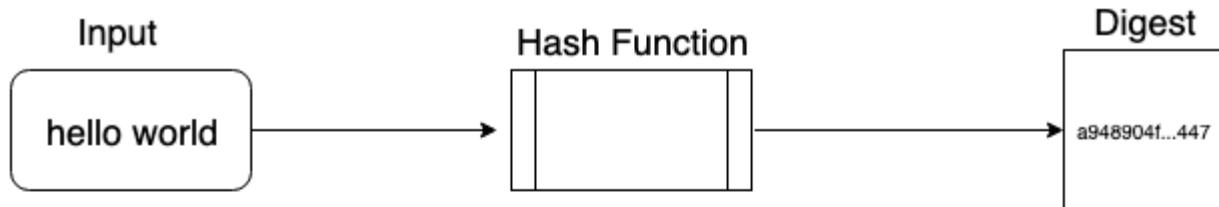
$$f(x \cdot y) = f(x) \cdot f(y)$$

To try out operations on finite fields, see <https://asecuritysite.com/encryption/finite>

For a great introduction see <http://coders-errand.com/zk-snarks-and-their-algebraic-structure/>

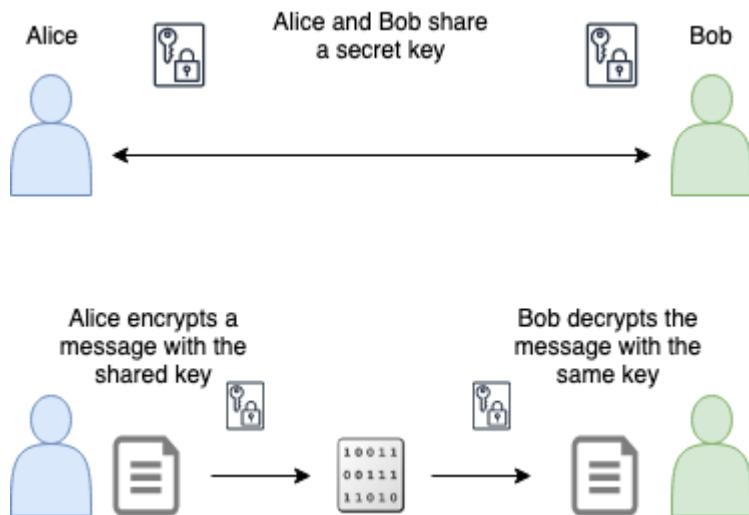
# Cryptography Background

## Hash Functions



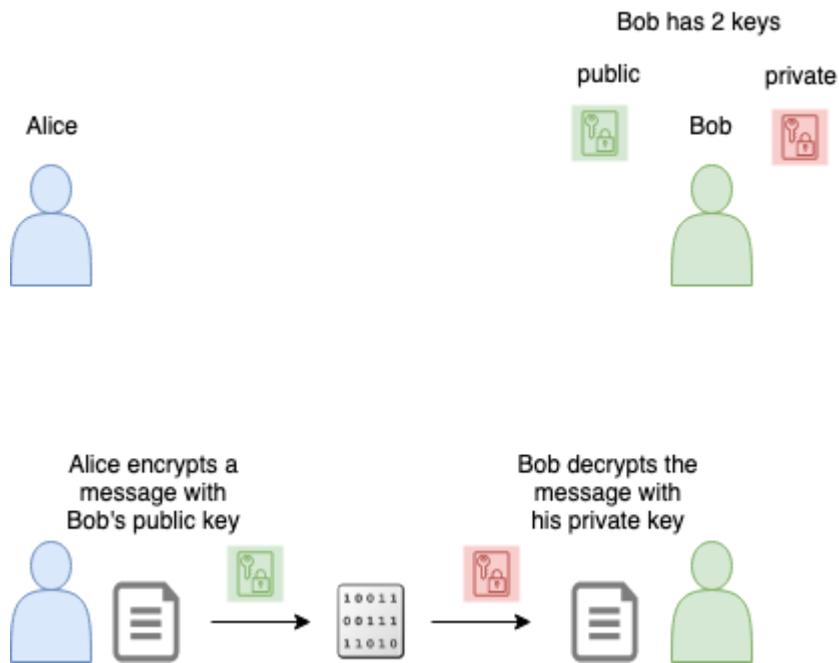
## Encryption

### Symmetric Encryption

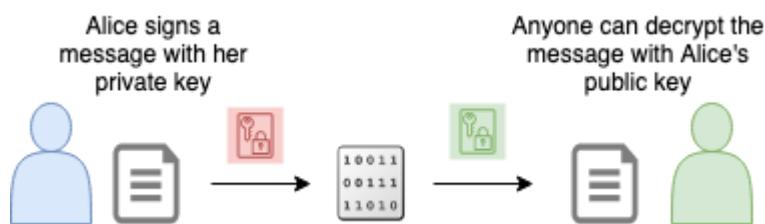


# Asymmetric Encryption

## Sending a secret message



## Proving ownership (knowledge of) of a private key



# Verifiable Random Functions

From [Algorand VRFs](#) :

A Verifiable Random Function (VRF) is a cryptographic primitive that maps inputs to verifiable pseudorandom outputs. VRFs were introduced by Micali, Rabin, and Vadhan in '99.

Given an input value  $x$ , the knowledge of the secret key  $SK$  allows one to compute  $y = F_{SK}(x)$  together with the proof of correctness  $\pi_x$ . This proof convinces every verifier that the value  $y = F_{SK}(x)$  is indeed correct with respect to the public key of the VRF. We can view VRFs as a commitment to a number of random-looking bits

The owner of a secret key can compute the function value as well as an associated proof for any input value. Everyone else, using the proof and the associated public key or verification key can check that this value was indeed calculated correctly, yet this information cannot be used to find the secret key.

Algorand have released it as an extension to the [libsodium](#) library

Such functions are ideal to find block producers in a blockchain in a trustless verifiable way.

---

# Elliptic Curves

An elliptic curve is a set of points  $(x, y)$  that satisfy an equation such as

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

where  $a$  and  $b$  are constants.

In addition to the points on the curve, there is also an "identity element" called the point at infinity.

For certain equations they will satisfy the group axioms

- every two points can be added to give a third point (closure);
- it does not matter in what order the two points are added (commutativity);
- if you have more than two points to add, it does not matter which ones you add first either (associativity);
- there is an identity element.

We find that some curves do indeed form a group under the "point addition" operation.

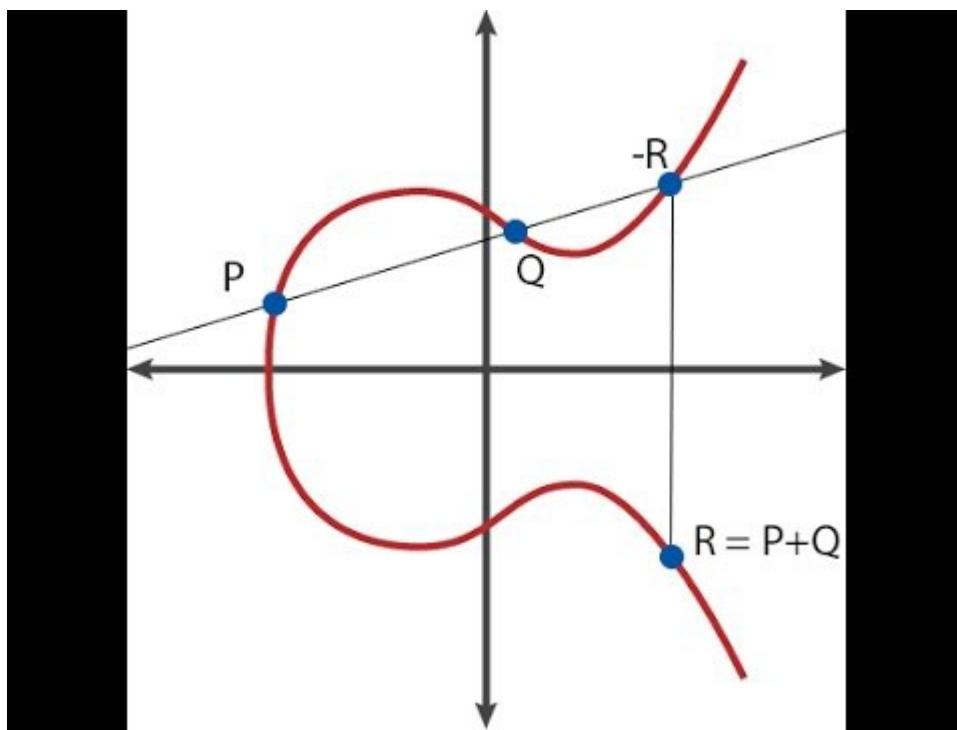
Point addition combines two points on the curve to produce a third point on the curve.

The point addition operation has special rules based on the geometric properties of elliptic curves, which ensure that the result of the addition is also a point on the curve.

Point addition is geometrically motivated, and it allows us to define scalar multiplication, which is the basis for elliptic curve cryptography.

We often use 2 families of curves :

## Montgomery Curves



For example curve 25519 with equation  $y^2 = x^3 + 486662x^2 + x$

Curve 25519 gives 128 bits of security and is used in the Diffie–Hellman (ECDH) key agreement scheme

BN254 / BN\_128 is the curve used in Ethereum for ZKSNARKS

BLS12-381 is the curve used by ZCash

## Edwards Curves

The general equation is  $ax^2 + y^2 = 1 + dx^2y^2$  with  $a = 1$   
for some scalar d which can be 0 or 1.

If  $a \neq 1$  they are called Twisted Edwards Curves

Every twisted Edwards curve is birationally equivalent to a Montgomery curve

## Scalar multiplication

Scalar multiplication is an operation in which a point on an elliptic curve is added to itself a certain number of times.

The result of this operation is another point on the elliptic curve.

Given an elliptic curve  $E$  defined by the equation  $y^2 = x^3 + ax + b$ , where  $a$  and  $b$  are constants, and a point  $P(x, y)$  on the curve  $E$ , scalar multiplication is defined as follows:

$$kP = P + P + \dots + P \text{ (} k \text{ times)}$$

where  $k$  is a scalar and the addition operation is the point addition operation defined on the elliptic curve.

# Roots of unity

Recall, over a finite field  $\mathbb{F}_p$ , where  $p$  is a prime number, an elliptic curve has a finite number of points (including a special "point at infinity" denoted as  $O$ ).

The set of points on an elliptic curve forms a group under point addition

Roots of unity are points on the curve that, when added to themselves a certain number of times using point addition, result in the identity element  $O$ .

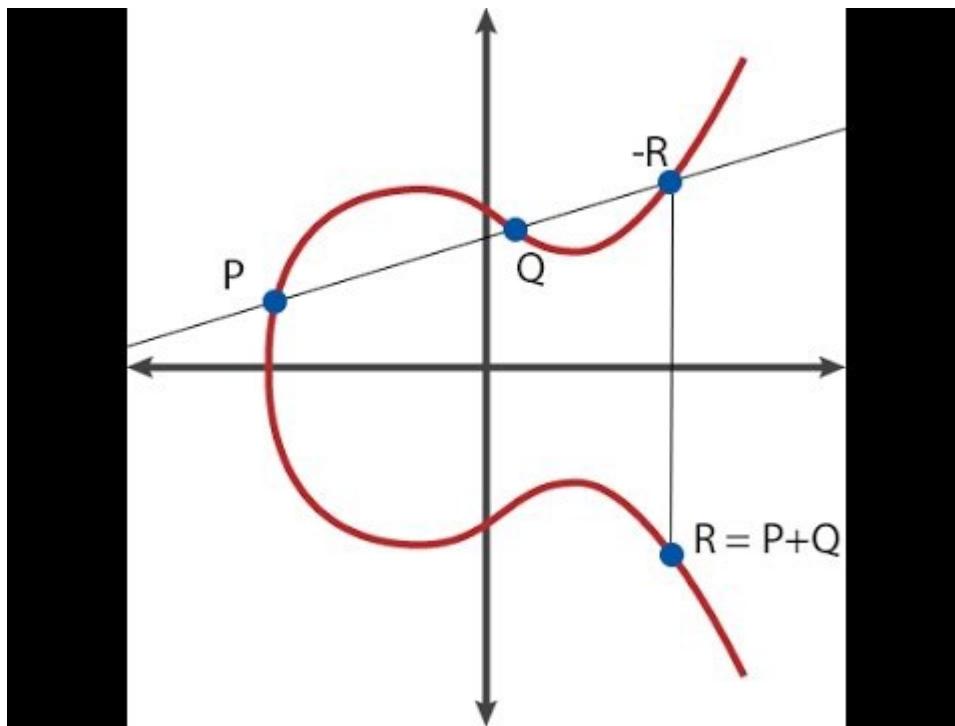
In other words, for a point  $P$  on the elliptic curve, there exists a positive integer  $n$  such that:

$$nP = P + P + \dots + P \text{ (n times)} = O$$

Here,  $n$  is called the order of the point  $P$ .

The smallest positive integer  $n$  for which this condition holds is the order of the point  $P$ .

# Visualisations



From

Serious Cryptography Jean-Philippe Aumasson

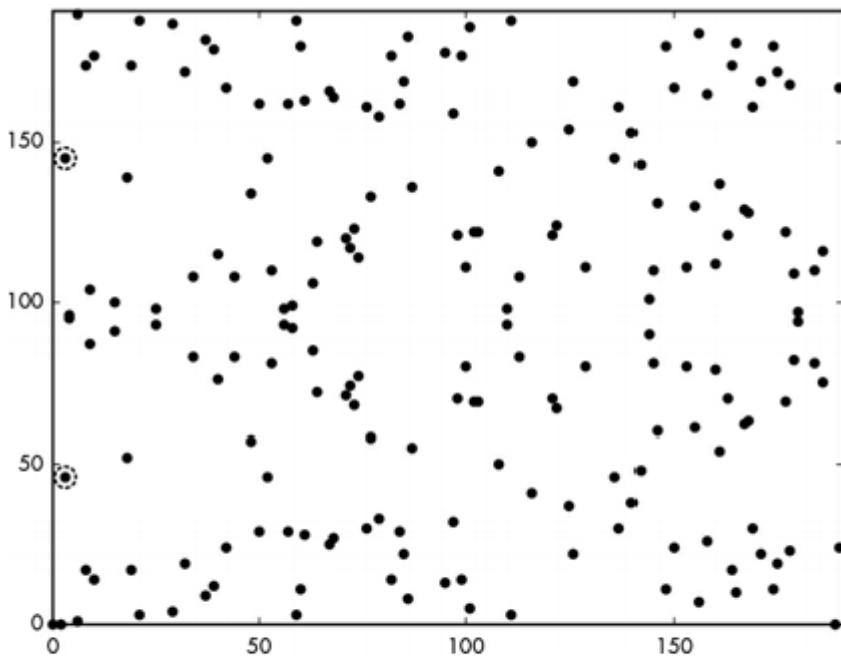


Figure 12-2: The elliptic curve with the equation  $y^2 = x^3 - 4x$  over  $\mathbb{Z}_{191}$ , the set of integers modulo 191

This [video](#) gives further examples of elliptic curve arithmetic.

# Polynomial Introduction

A polynomial is an expression that can be built from constants and variables by means of addition, multiplication and exponentiation to a non-negative integer power.

e.g.  $3x^2 + 4x + 3$

Quote from Vitalik Buterin

"There are many things that are fascinating about polynomials. But here we are going to zoom in on a particular one: **polynomials are a single mathematical object that can contain an unbounded amount of information** (think of them as a list of integers and this is obvious)."  
Furthermore, **a single equation between polynomials can represent an unbounded number of equations between numbers.**

For example, consider the equation  $A(x)+B(x)=C(x)$ . If this equation is true, then it's also true that:

- $A(0)+B(0)=C(0)$
- $A(1)+B(1)=C(1)$
- $A(2)+B(2)=C(2)$
- $A(3)+B(3)=C(3)$

## Adding, multiplying and dividing polynomials

We can add, multiply and divide polynomials, for examples

see [https://en.wikipedia.org/wiki/Polynomial\\_arithmetic](https://en.wikipedia.org/wiki/Polynomial_arithmetic)

## Roots

For a polynomial  $P$  of a single variable  $x$  in a field  $K$  and with coefficients in that field, the root  $r$  of  $P$  is an element of  $K$  such that  $P(r) = 0$

$B$  is said to divide another polynomial  $A$  when the latter can be written as

$$A = BC$$

with  $C$  also a polynomial, the fact that  $B$  divides  $A$  is denoted  $B|A$

If one root  $r$  of a polynomial  $P(x)$  of degree  $n$  is known then polynomial long division can be used to factor  $P(x)$  into the form

$$(x - r)(Q(x))$$

where

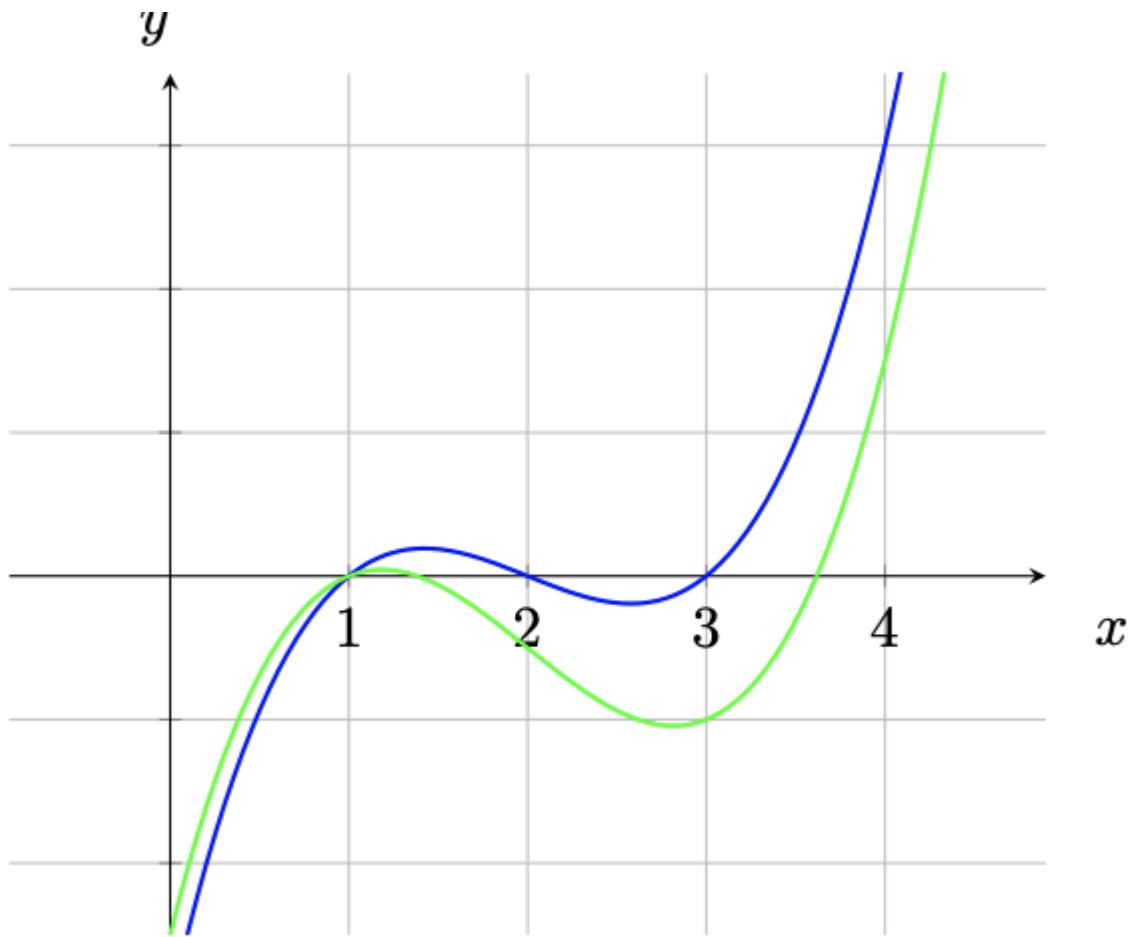
$Q(x)$  is a polynomial of degree  $n - 1$ .

$Q(x)$  is simply the quotient obtained from the division process; since  $r$  is known to be a root of  $P(x)$ , it is known that the remainder must be zero.

## Schwartz-Zippel Lemma

"different polynomials are different at most points".

Polynomials have an advantageous property, namely, if we have two non-equal polynomials of degree at most  $d$ , they can intersect at no more than  $d$  points.



if  $f$  and  $g$  are polynomials and are equal, then

$$f(x) = g(x) \text{ for all } x$$

if  $f$  and  $g$  are polynomials and are NOT equal, then

$$f(x) \neq g(x) \text{ for all pretty much any } x$$

## What does it mean to say 2 polynomials are equal ?

1. They evaluate to the same value or all points
2. They have the same coefficients

If we are working with real numbers, these 2 points would go together, however that is not the case when we are working with finite fields.

For example all elements of a field of size  $q$  satisfy the identity

$$x^q = x$$

The polynomials  $X^q$  and  $X$  take the same values at all points, but do not have the same coefficients.

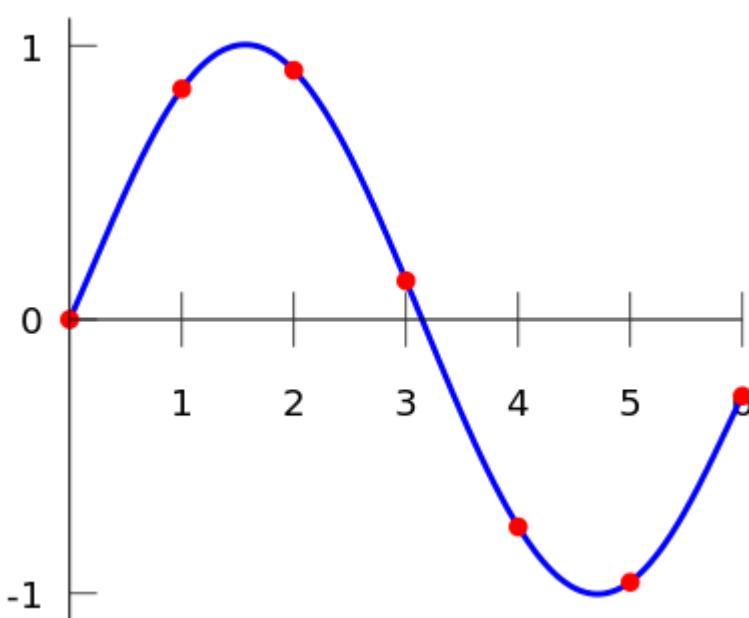
## Lagrange Interpolation

If you have a set of points then doing a Lagrange interpolation on those points gives you a polynomial that passes through all of those points.

If you have two points on a plane, you can define a single straight line that passes through both, for 3 points, a single 2nd-degree curve (e.g.  $5x^2 + 2x + 1$ ) will go through them etc.

For n points, you can create a n-1 degree polynomial that will go through all of the points.

(We can use this in all sorts of interesting schemes as well as zkps)



## Representations

We effectively have 2 ways to represent polynomials

1. Coefficient form

$$f(x) = a_0 + a_1x + a_2x^2 + a_3x^3\dots$$

2. Point value form

$$(x_1, y_1), (x_2, y_2), \dots$$

We can switch between the two forms by evaluation or interpolation.

---

# Zero Knowledge Proof Introduction

## Intuitive grasp of Zero Knowledge Proofs

It is difficult to find zero knowledge resources that avoid the extremes of either over simplifying the subject, or presenting so much mathematical detail that the reader gets bogged down and loses interest.

In this course I hope to find an accessible but informative middle ground.

We start with some examples to show how zero knowledge proofs can proceed, and the situations where they could be used.

## What is a zero knowledge proof

A loose definition

It is a proof that there exists or that we know something, plus a zero knowledge aspect, that is the person verifying the proof only gains one piece of information - that the proof is valid or invalid.

## Actors in a Zero Knowledge Proof System

- Creator - optional, maybe combined with the prover
- Prover - I will call her Peggy
- Verifier - I will call him Victor

## Examples to give an Intuitive grasp of zero-knowledge proofs

### 1. Colour blind verifier

This is an interactive proof showing that the prover can distinguish between a red and a green billiard ball, whereas the verifier cannot distinguish them.

- The prover wants to show the verifier that they have different colours but does not want him to learn which is red and which is green.
  - Step 1: The verifier takes the balls, each one in each hand, holds them in front of the prover and then hides them behind his back. Then, with probability  $1/2$  either swaps them (at most once) or keeps them as they are. Finally, he brings them out in front.
  - Step 2: The prover has to say the verifier switched them or not.
  - Step 3: Since they have different colours, the prover can always say whether they were switched or not.
- But, if they were identical (the verifier is inclined to believe that), the prover would be wrong with probability  $1/2$ .
- Finally, to convince the verifier with very high probability, the prover could repeat Step 1 to Step 3  $k$  times to reduce the probability of the prover being successful by chance to a extremely small

amount.

---

## 2. Wheres Wally

Based on the pictures of crowds where Wally is distinctivly dressed, the aim being to find him within a sea of similar people.

The proof procedes as follows :

Imagine the Peggy has found Wally in the picture and wants to prove this fact to Victor, however if she just shows him, Victor is liable to cheat and claim he also found Wally.

In order to prove to Victor that she has indeed found Wally, without giving away his location in the picture

3. Peggy cuts a hole in a (very) large sheet of paper, the hole should be the exact shape of Wally in the underlying picture.
4. Peggy places the paper sheet over the original picture, so that the location of the picture beneath the paper is obscured.
5. Victor can then see throught he hole that Wally has indeed been found, but since the alignment with the underlying picture cannot be seen, he doesn't gain any information about the location of Wally.



### 3. Sudoku

An interactive proof can be created to prove the knowledge of a solution to a sudoku puzzle by placing cards in the sudoku grid. The process is described here [Sudoku Proof](#)

Quote from Vitalik Buterin

"You can make a proof for the statement "I know a secret number such that if you take the word 'cow', add the number to the end, and SHA256 hash it 100 million times, the output starts with

0x57d00485aa ". The verifier can verify the proof far more quickly than it would take for them to run 100 million hashes themselves, and the proof would also not reveal what the secret number is."

---

# Zero Knowledge Proof Timeline

Changes have occurred because of

- Improvements to the cryptographic primitives (improved curves or hash functions for example)
- A fundamental change to the approach to zero knowledge  
See the excellent blog post from Starkware :  
[The Cambrian Explosion](#)

1984 : Goldwasser, Micali and Rackoff - Probabilistic Encryption.

1989 : Goldwasser, Micali and Rackoff - The Knowledge Complexity of Interactive Proof Systems

1991 O Goldreich, S. Micali and A. Wigderson. Proofs that Yield Nothing but their Validity or All Languages in NP Have Zero-Knowledge Proof Systems. Preliminary version in 1986. (Graph colouring problem)

....

2006 Groth, Ostrovsky and Sahai introduced pairing-based NIZK proofs, yielding the first linear size proofs based on standard assumptions.

2010 Groth combined these techniques with ideas from interactive zero-knowledge arguments to give the first constant size NIZK arguments.

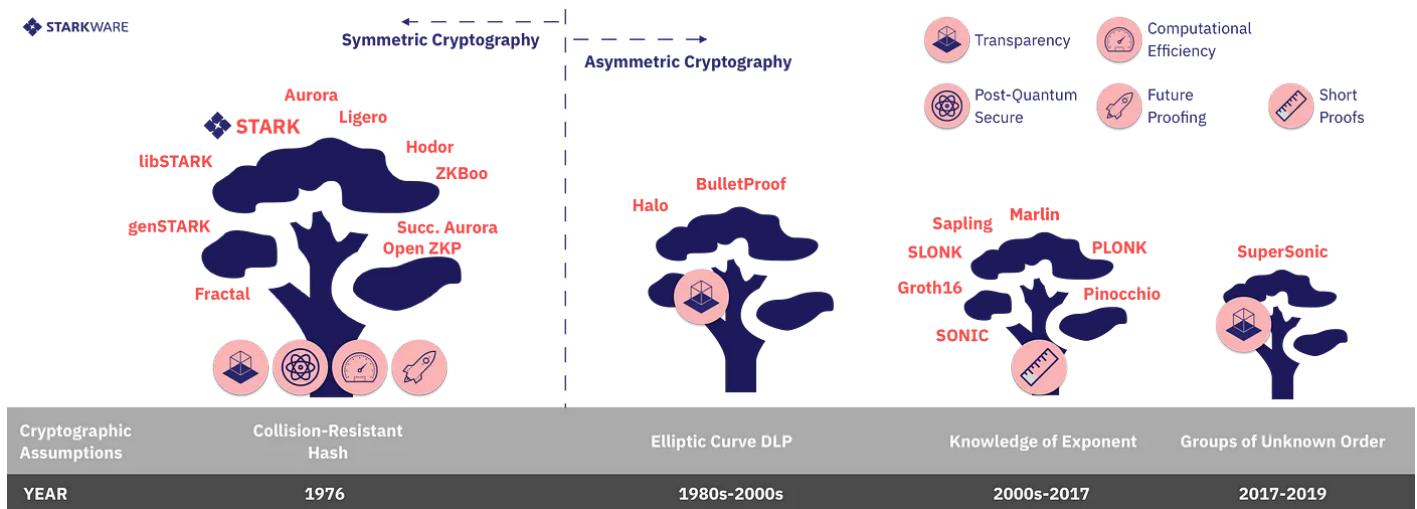
2016 : Jens Groth - On the Size of Pairing-based Non-interactive Arguments

From [Matthew Green](#)

Prior to Goldwasser et al., most work in this area focused on the soundness of the proof system. That is, it considered the case where a malicious Prover attempts to 'trick' a Verifier into believing a false statement. What Goldwasser, Micali and Rackoff did was to turn this problem on its head. Instead of worrying only about the Prover, they asked: what happens if you don't trust the Verifier?

---

# ZKP Ecosystem



from

[The Cambrian Explosion](#)

---

# Proving Systems

A statement is a proposition we want to prove. It depends on:

- Instance variables, which are public.
- Witness variables, which are private.

Given the instance variables, we can find a short proof that we know witness variables that make the statement true (possibly without revealing any other information).

What do we require of a proof ?

- Completeness: there exists an honest prover P that can convince the honest verifier V of any correct statement with very high probability.
- Soundness: even a dishonest prover P running in super-polynomial time cannot convince an honest verifier V of an incorrect statement. Note: P does not necessarily have to run in polynomial time, but V does.

To make our proof zero knowledge we also need 'zero knowledginess'

To oversimplify: represented on a computer, a ZKP is nothing more than a sequence of numbers, carefully computed by Peggy, together with a bunch of boolean checks that Victor can run in order to verify the proof of correctness for the computation.

A zero-knowledge protocol is thus the mechanism used for deriving these numbers and defining the verification checks.

## Interactive v Non Interactive Proofs

Non-interactive is only useful if we want to allow multiple independent verifiers to verify a given proof without each one having to individually query the prover.

In contrast, in non-interactive zero knowledge protocols there is no repeated communication between the prover and the verifier. Instead, there is only a single "round", which can be carried out asynchronously.

Using publicly available data, Peggy generates a proof, which she publishes in a place accessible to Victor (e.g. on a distributed ledger).

Following this, Victor can verify the proof at any point in time to complete the "round". Note that even though Peggy produces only a single proof, as opposed to multiple ones in the interactive version, the verifier can still be certain that except for negligible probability, she does indeed know the secret she is claiming.

## Succint v Non Succint

Succinctness is necessary only if the medium used for storing the proofs is very expensive and/or if we need very short verification times.

## Proof v Proof of Knowledge

A proof of knowledge is stronger and more useful than just proving the statement is true. For instance, it allows me to prove that I know a secret key, rather than just that it exists.

## Argument v Proof

In a proof, the soundness holds against a computationally unbounded prover and in an argument, the soundness only holds against a polynomially bounded prover.

Arguments are thus often called "computationally sound proofs".

The Prover and the Verifier have to agree on what they're proving. This means that both know the statement that is to be proven and what the inputs to this statement represent.

---