

# Bitcoin and Cryptocurrency Technologies

## Lecture 3: Cryptography Basics 2/2

Yuri Zhykin

Apr 19, 2022

# Public-key Cryptography Recap

- **Public-key cryptography** or **asymmetric cryptography** is a cryptographic system that uses **pairs** of keys:
  - **private keys**, which must be kept secret and only known to the owner,
  - **public keys**, which may be known to others.
- The core idea behind public-key cryptography is that anyone who knows the public key, can “lock” some information with that key in a way that only the owner of the private key can “unlock” it.
- Introduced by Ralph Merkle, Whitfield Diffie, Martin Hellman and others in 1970s.
- Arguably the only reason why it’s possible to do anything useful over the Internet.

# Fields 1/2

- A **field** is a set with defined **addition**, **subtraction**, **multiplication** and **division** operations that follow the **field axioms**:

$\forall a, b, c : (a \star b) \star c = a \star (b \star c)$  - associativity for  $+$  and  $*$

$\forall a, b : a \star b = b \star a$  - commutativity for  $+$  and  $*$

$\exists e_+ = 0 : \forall a : e + a = 0 + a = a$  - additive identity

$\exists e_* = 1 : \forall a : e * a = 1 * a = a$  - multiplicative identity

$\forall a : \exists (-a) : a + (-a) = e_+ = 0$  - additive inverse

$\forall a \neq 0 : \exists (a^{-1}) : a * (a^{-1}) = e_* = 1$  - multiplicative inverse

$\forall a, b, c : a * (b + c) = (a * b) + (a * c)$  -  $*$  over  $+$  distributivity

- Set of **rational numbers**  $\mathbb{R}$  is a field over regular addition and multiplication.
- In cryptography, we usually consider **finite (Galois) fields** with **prime order** over modular arithmetic operations:

$$F_n = \mathbb{Z}/n\mathbb{Z} = 0, 1, \dots, n-1$$

where  $n$  is a prime number; note that **this construction is a field iff  $n$  is prime**.

# Groups 1/2

- A **group** is a set equipped with a binary operation that combines any two elements to form a third element in such a way that three conditions called **group axioms** are satisfied, namely

$\forall a, b, c : (a \star b) \star c = a \star (b \star c)$  - associativity

$\exists e : \forall a : e \star a = a$  - identity

$\forall a : \exists b : a \star b = e$  - invertibility

- A **generating set of a group** is a subset of the group set such that every element of the group can be expressed as a combination of finitely many of the subsets elements and their inverses.
- A group generated by a single element (usually denoted as  $G$ ) is called a **cyclic group**.

## Groups 2/2

- Let  $\mathbb{G}$  be any group. Let  $a, b \in \mathbb{G}$ . Denote the group operation by multiplication and its identity element by 1. Let

$$b^k = a$$

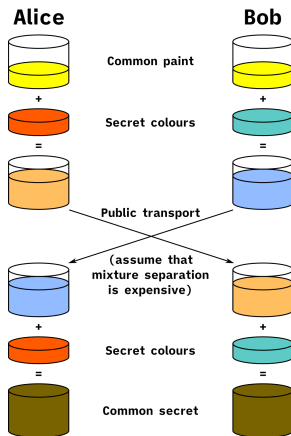
- $k$  that satisfies the above equation is called the **discrete logarithm** of  $a$  to the base  $b$ .
- If we denote group operation by addition and its identity by 0, the notation for **discrete logarithm** becomes

$$kb = a$$

- The **discrete logarithm problem** or simply **DLOG** is **believed** to be **very hard** when defined over certain *groups*.

# Diffie-Hellman Key Exchange Protocol 1/3

- Diffie-Hellman-Merkle key exchange algorithm can be intuitively explained with the following example:



## Diffie-Hellman Key Exchange Protocol 2/3

- DH protocol implementations are based on the following observation written additively

$$A = aG (= G + G + \dots + G)$$

$$B = bG$$

$$bA = b(aG) = (ba)G = (ab)G = a(bG) = aB$$

or multiplicatively

$$A = G^a (= G * G * \dots * G)$$

$$B = G^b$$

$$A^b = (G^a)^b = G^{(ab)} = G^{(ba)} = (G^b)^a = B^a$$



# Diffie-Hellman Key Exchange Protocol 3/3

- Simplest implementation of the DH protocol (as described in the paper) uses the **multiplicative group of integers modulo  $p$** , where  $p$  is **prime**, and  $g$  is a **primitive root modulo  $p$** .
- Example DH with small numbers:
  - Alice and Bob agree to use numbers modulo  $p = 23$  and base  $G = 5$ .
  - Alice chooses a secret integer  $a = 4$ , then sends Bob

$$A = G^a \pmod{p} = 5^4 \pmod{23} = 4$$

- Bob chooses a secret integer  $b = 3$ , then sends Alice

$$B = G^b \pmod{p} = 5^3 \pmod{23} = 10$$

- Alice computes

$$s = B^a \pmod{p} = 10^4 \pmod{23} = 18$$

- Bob computes

$$s = A^b \pmod{p} = 4^3 \pmod{23} = 18$$

# Cryptographic Signatures

- **Cryptographic signature scheme** is a system for verifying the authenticity of messages.
- A cryptographic signature scheme consists of the following three algorithms:
  - a key generation algorithm *Gen* that selects a private key uniformly at random from a set of possible private keys,
  - a signing algorithm *Sign* that, given a message and a private key, produces a signature,
  - a signature verification algorithm *Verify* that, given the message, public key and signature, either accepts or rejects the signature.
- Successful signature verification provides a **very strong** reason to believe that the particular message was authenticated by the owner of the corresponding private key.

# DSA 1/2

- Alice and Bob must agree on the group parameters ( $GROUP, G, n$ ), where  $GROUP$  is a group of prime order  $n$  with generator  $G$ .
- Alice creates a key pair, consisting of a private key integer  $a$ , randomly selected in the interval  $[1, n - 1]$  and a public key group element  $A = aG$ .
- To sign a message  $m$ , Alice
  - calculates  $e = HASH(m)$ ,
  - selects a **cryptographically secure random integer**  $k \in [1, n - 1]$ ,
  - calculates the group element  $(x_1, y_1) = kG$ ,
  - calculates  $r = x_1 \pmod{n}$ ,
  - calculates  $s = k^{-1}(e + ra) \pmod{n}$ .
- Signature is a pair  $(r, s)$ ;  $(r, -s \pmod{n})$  is **also valid**.

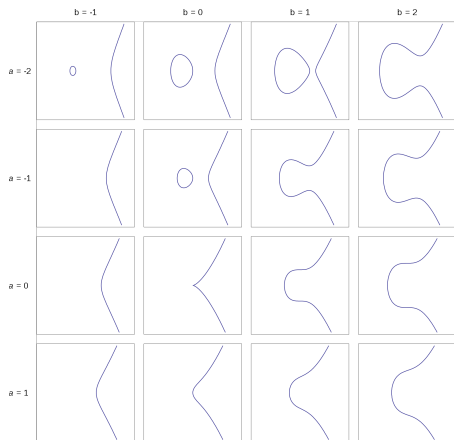
- To verify the signature  $(r, s)$ , Bob
  - verifies that  $r$  and  $s$  are integers in  $[1, n - 1]$ , otherwise the signature is invalid,
  - calculates  $e = \text{HASH}(m)$ ,
  - calculates  $u_1 = es^{-1} \pmod{n}$  and  $u_2 = rs^{-1} \pmod{n}$ ,
  - calculates the group element  $(x_1, y_1) = u_1G + u_2A$ .
- The signature is valid if  $r \equiv x_1 \pmod{n}$  and invalid otherwise:

$$\begin{aligned}
 u_1G + u_2A &= u_1G + u_2aG = (u_1 + u_2a)G \\
 &= (es^{-1} + rs^{-1}a)G = (e + ra)s^{-1}G \\
 &= (e + ra)(a + ra)^{-1}kG = kG \\
 &= r
 \end{aligned}$$

# Elliptic Curves 1/3

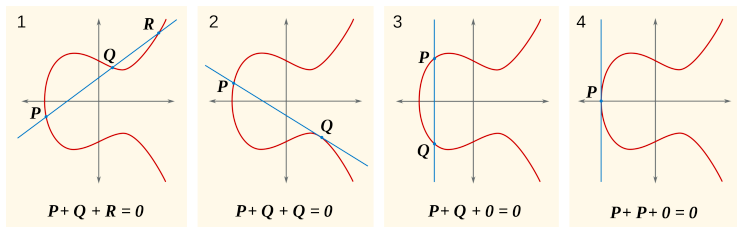
- **Elliptic curves** are algebraic structures described by equations of the form:

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



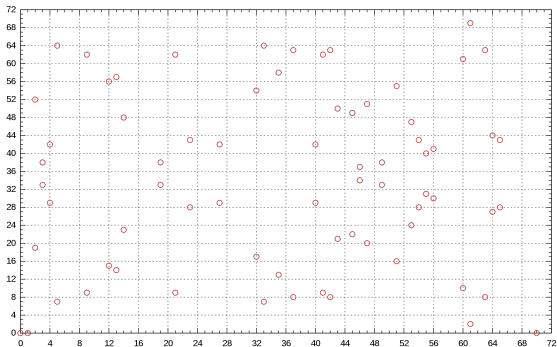
# Elliptic Curves 2/3

- Elliptic curves are defined over a **field**  $K$  and describes points in  $K \times K$ .
- **Group law:**
  - If  $P$  and  $Q$  are two points on the curve, then we can uniquely describe a third point,  $P + Q$ , in the following way. First, draw the line that intersects  $P$  and  $Q$ . This will generally intersect the cubic at a third point,  $R$ . We then take  $P + Q$  to be  $-R$ , the point opposite  $R$ .



## Elliptic Curves 3/3

- Elliptic curves defined over finite fields of prime order form groups that are well suited for cryptographic purposes because of significantly **more complex group structure**, which allows for much **smaller keys**.
- An example of an elliptic curve defined over finite field ( $y^2 = x^3 - x$  over  $F_{71}$ ):



- **Elliptic Curve Digital Signature Algorithm (ECDSA)** is a variant of the **Digital Signature Algorithm (DSA)** which uses **elliptic curve cryptography**.
- With **ECC**, the bit size of the private key believed to be needed for ECDSA is about **twice the size** of the security level, i.e. for 128 bits of security, the key size of 256 bits is sufficient.
- With DSA, the bit size of a key that offers comparable security is 2048 or even 3072 bits.



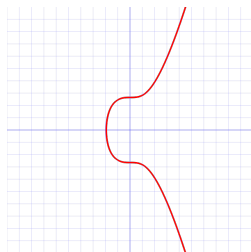
# Elliptic Curve SECP256K1 1/2

- Elliptic curve used in Bitcoin for transaction signing is called **secp256k1**.
- This elliptic curve is defined over the finite field  $F_p$  where

$$p = 2^{256} - 2^{32} - 2^9 - 2^8 - 2^7 - 2^6 - 2^4 - 1$$

and is described by the equation

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



# Elliptic Curve SECP256K1 2/2

- **secp256k1** curve is constructed in a special non-random way which allows for especially efficient computation.
- Unlike other (NIST) elliptic curves used in cryptography (with the notable exception of **curve25519**), **secp256k1**'s constants were selected in a predictable way, which significantly reduces the possibility of an existing *backdoor*.
- **libsecp256k1** is a highly-optimized **secp256k1** curve implementation that was extracted from the Bitcoin Core code base into a separate project:

<https://github.com/bitcoin-core/secp256k1>

- Bitcoin uses **secp256k1** for traditional ECDSA signatures, as well as the Schnorr signatures, introduced as part of the **Taproot** upgrade, which was activated on November 14, 2021.

- **A Computational Introduction to Number Theory and Algebra** by Victor Shoup
  - <https://shoup.net/ntb/>

# The End

Thank you!