Bitcoin and Cryptocurrency Technologies
Lecture 3: Cryptography Basics 2/2

Yuri Zhykin

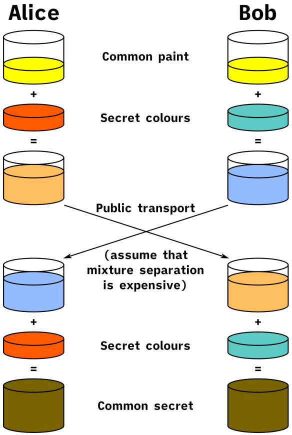
Mar 2, 2023

Public-key Cryptography Recap

- **Public-key cryptography** or **asymmetric cryptography** involves cryptographic systems that use **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.

Diffie-Hellman Key Exchange Protocol 1/2

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



Diffie-Hellman Key Exchange Protocol 2/2

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

$$\begin{aligned}
A &= aG (= G + G + \ldots + G) \\
B &= bG \\
bA &= b(aG) = (ba)G = (ab)G = a(bG) = aB
\end{aligned}$$

or multiplicatively

$$\begin{aligned}
A &= G^a (= G * G * \ldots * G) \\
B &= G^b \\
A^b &= (G^a)^b = G^{(ab)} = G^{(ba)} = (G^b)^a = B^a
\end{aligned}$$

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

Fields 2/2

- 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:

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

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

$$\forall a, b, c : (a \star b) \star c = a \star (b \star c) \text{ - associativity}$$
$$\exists e : \forall a : e \star a = a \text{ - identity}$$
$$\forall a : \exists b : a \star b = e \text{ - invertibility}$$

- 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*.

Simple Diffie-Hellman Key Exchange Implementation

- 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

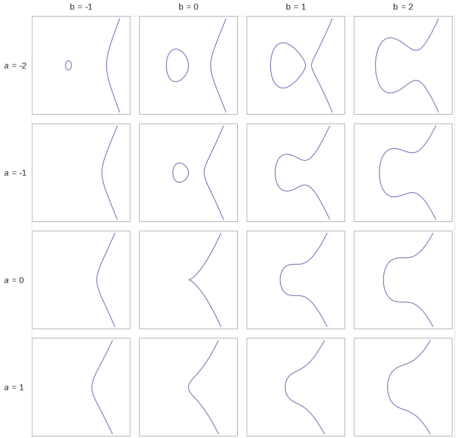
$$s = B^a \pmod{p} = 10^4 \pmod{23} = 18$$

  - Bob computes

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

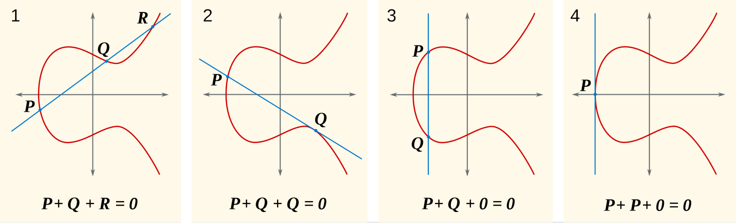
- **Elliptic curves** are algebraic structures described by equations of the form:

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

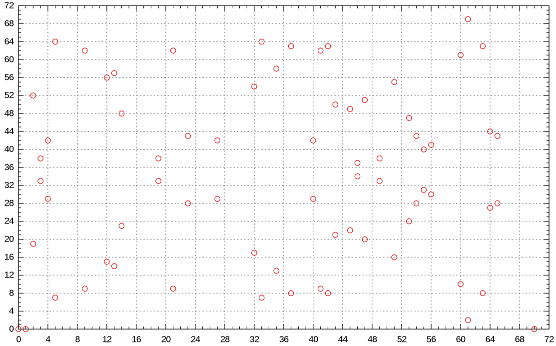


Elliptic Curves 2/3

- 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 curve at a third point, R. We then take $P + Q$ to be $-R$, the point opposite R.



- **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 filed ($y^2 = x^3 - x$ over F_{71}):



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 1/4

- Alice and Bob must agree on the group parameters (p, q, Z_p*, G), where p is prime, q is a smaller prime that divides $p-1$, Z_p* is the multiplicative group of integers modulo p, G is a generator of a subgroup of Z_p* of order q.
- Alice creates a key pair, consisting of a private key integer x, randomly selected from $[1, q-1]$ and a public key group element $y = G^x \pmod p$.
- To sign a message m, Alice
  - calculates $e = HASH(m)$,
  - selects a **cryptographically secure random integer** $k \in [1, q-1]$ such that k and q are relatively prime,
  - calculates $r = G^k \pmod p$,
  - calculates $s = k^{-1}(e + x * r) \pmod q$.
- Signature is a pair (r, s); $(r, -s \pmod n)$ is **also valid**.

DSA 2/4

- To verify the signature (r, s), Bob
  - receives Alice's public key y, the message m and the signature (r, s)
  - verifies that r and s are integers in $[1, q-1]$, otherwise the signature is invalid,
  - calculates $w = s^{-1} \pmod q$,
  - calculates $e = HASH(m)$,
  - calculates $u_1 = ew$ and $u_2 = rw \pmod q$,
  - calculates the group element $v = G^{u_1} * y^{u_2} \pmod p \pmod q$.
- The signature is valid if $r \equiv v$ and invalid otherwise:

$$
\begin{aligned}
G^{u_1} * y^{u_2} &= G^{u_1} * G^{u_2 x} = G^{u_1 + u_2 x} \\
&= G^{es^{-1} + rs^{-1}x} = G^{(e+rx)s^{-1}} \\
&= G^{(e+rx)(e+rx)^{-1}k} = G^k \\
&= r
\end{aligned}
$$

DSA 3/4

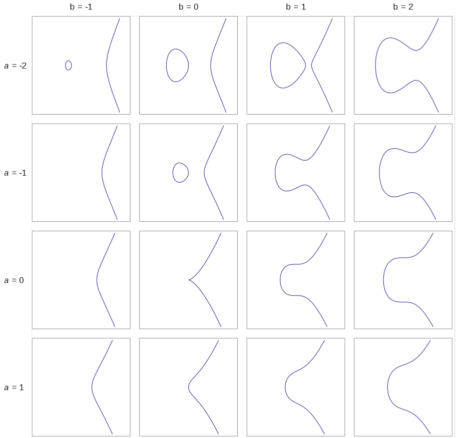
- To illustrate the process using small numbers:
- Global public key elements:
  - Choose a prime $p = 23$ and a prime $q = 11$ that divides $p - 1$.
  - Choose an integer $h = 4 (1 < h < p - 1)$, then calculate $G = h^{(p-1)/q} \pmod{p} = 4^{(23-1)/11} \pmod{23} = 8$.
- Alice's key pair:
  - Private key x is a randomly selected integer from $[1, q - 1]$, let's say $x = 6$.
  - Public key y is $G^x \pmod{p} = 8^6 \pmod{23} = 2$.
- Message signing:
  - Select a random integer k from $[1, q - 1]$, let's say $k = 9$.
  - Computes $r = g^k \pmod{p} \pmod{q} = 8^9 \pmod{23} \pmod{11} = 6 \pmod{11} = 6$.
  - Say $e = HASH(m) = 8$. Compute $s = k^{-1} * (e + x * r) \pmod{q} = 9^{-1} * (8 + 6 * 6) \pmod{11} = 5 * (8 + 36) \pmod{11} = 8$.
  - The signature is $(r, s) = (6, 8)$.

DSA 4/4

- Verification (Bob's step):
  - Bob receives Alice's public key $y = 2$, the message $m = "Hello"$, and the signature $(r, s) = (6, 8)$.
  - Bob calculates:
    - $w = s^{-1} \pmod{q} = 8^{-1} \pmod{11} = 7$ (because $8 * 7 \pmod{11} = 1$).
    - $u_1 = (HASH(m) * w) \pmod{q} = (8 * 7) \pmod{11} = 1$.
    - $u_2 = (r * w) \pmod{q} = (6 * 7) \pmod{11} = 9$
  - Finally, Bob checks if $r = (G^{u_1} * y^{u_2} \pmod{p}) \pmod{q} = (8^1 * 2^9 \pmod{23}) \pmod{11} = 6$.
- Since r matches, Bob can verify that the signature is from Alice.

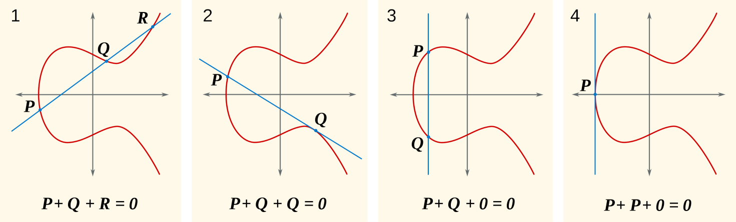
- **Elliptic curves** are algebraic structures described by equations of the form:

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

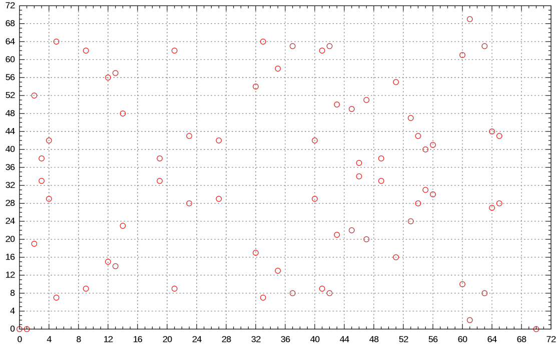


Elliptic Curves 2/3

- 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 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}):



ECDSA

- **Elliptic Curve Digital Signature Algorithm** (**ECDSA**) is a variant of the **Digital Signature Algorithm** (**DSA**) which uses **elliptic curve cryptography**.
- The ECDSA is a variant of the DSA that operates in the group of points on an elliptic curve over a finite field, rather than the multiplicative group of integers modulo a prime.
- 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.

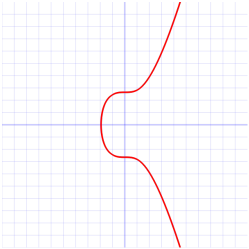
Elliptic Curve SECP256K1 1/2

- 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.

Useful Resources

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

The End

Thank you!