MIT Digital Currency Initiative and the University of Brasilia presents

Cryptocurrency Design and Engineering

Assignment 2: Background
Presented by: Matthew Waleyko
10/22/2025
MAS.S62

Required Materials

- Computer with Python3 installed
  - https://www.python.org/downloads/
- Python's ecdsa library
  - python –m pip install ecdsa
  - pip install ecdsa
- Instructions and Starter Code
  - https://github.com/mit-dci-cde-2025/mit-dci-cde-2025-classroom-students-assignment-2-cde2025.2-digital-signatures-makerere

Goals

- Have a basic working understanding of ECDSA’s usage in bitcoin
- A completed assignment 2
  - Create a signature
  - Sign a message
  - Forge 2 messages
  - Alter a valid signature

Hashing

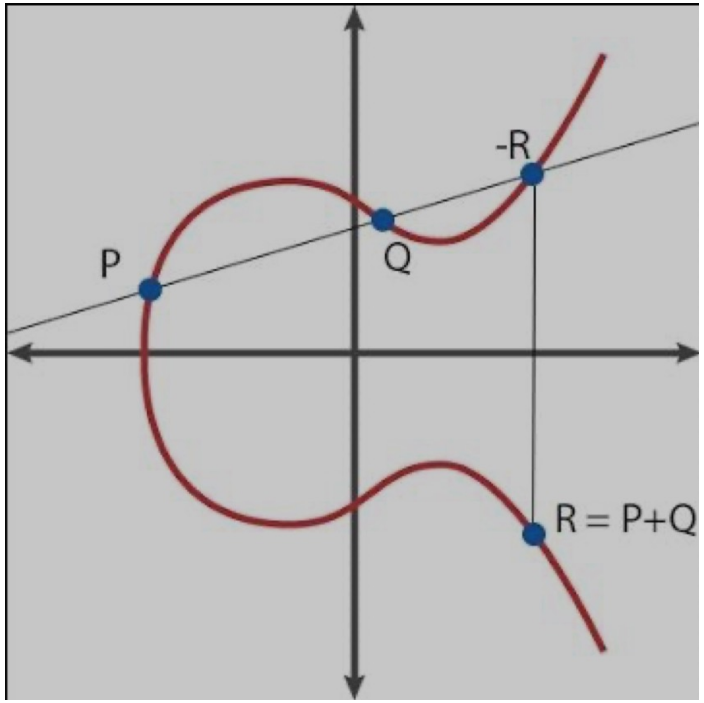
- Takes an arbitrary length input
- Outputs fixed length output
- The output appears random but is deterministic
- Allows for data integrity
- Allows for the commitment of information
  - It will rain tomorrow
    - aa6937cfa2c147206b40c2fe45e8182174bad1d705eae2d91efb8e773e22d36f
  - It will not rain tomorrow
    - 725763ae3279cc027377cf04cce9a5fa14f9850852be6435040b9c16083e7d13

Digital Signatures

- Method by which a data can be verified
- Consists of two key components
  - Private Key
  - Public Key
- Used to verify information came from a specific source
- Relies on complex mathematics
  - RSA
  - ECDSA

Elliptic Curve Digital Signature Algorithm

- Smaller keys
- Faster computation
- Patent free
- Capitol letters will represent a Point on the curve
- Lower Case letters represent a scaler
- P+Q+R = 0
- P+Q = -R



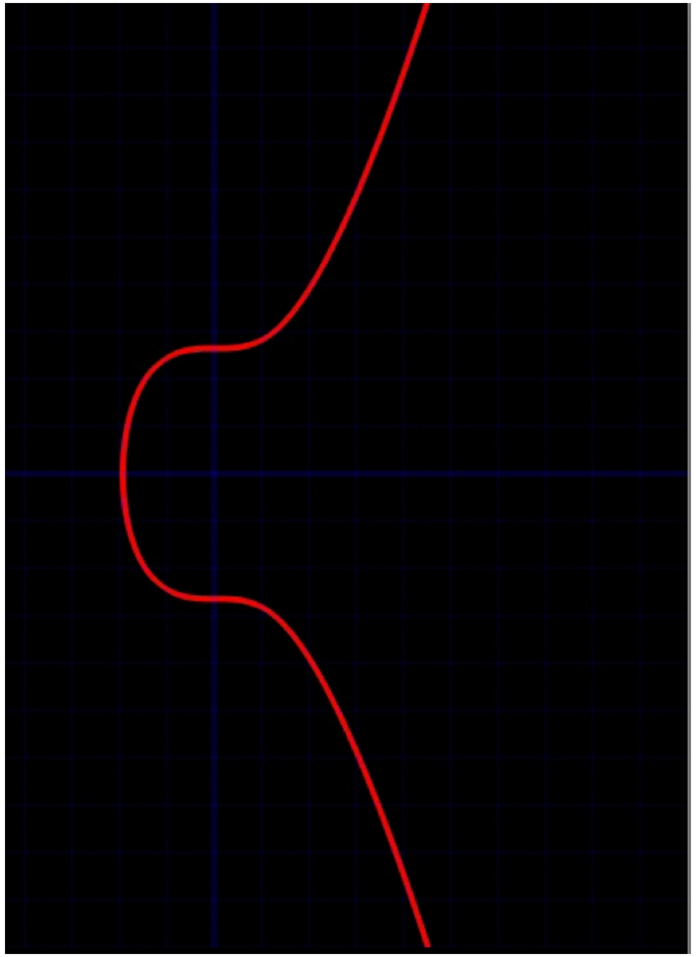
Tadge Dryja. Cryptocurrency Engineering and Design. 2018. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA.

Elliptic Curve Operations

- A+B
- A-B
- A*b
- A/b

Elliptic Curve Digital Signature Algorithm

- SECP256k1
- $y^2 = x^3 + 7$
- G and n are constants
- Private key is random integer e
- Public key is derived from the private key
  - E = e*G
- R = k*G
- r = x mod n
- s = k^(-1) * (hash(m) + r*e) mod n



Tadge Dryja. Cryptocurrency Engineering and Design. 2018. Massachusetts Institute of Technology: MIT OpenCouseWare, https://ocw.mit.edu/. License: Creative Commons BY-NC-SA.

Assignment 2

- Directories
    - data
    - graders
    - implementation
    - Solutions
- https://ecdsa.readthedocs.io/en/latest/modules.html

Exercise 1

- Create a Public/Private key pair
- Sign a message
  - hashlib.sha256(b"").digest().hex()
- Output your public key and signed message
  - All hex characters
  - Compressed format public key

Exercise 2

- Given a known k forge a message
- s = k^(-1) * (hash(m) + r*e) mod n
    - -> e = (s * k - h) * r_inv % n
- r_inv = pow(r, -1, n)

Exercise 3

- Given k reuse forge a message
- Sony PlayStation 3 Hack
- $k = (hash(m1) - hash(m2))/(s1 - s2) \bmod n$

Exercise 4

- Signature malleability
- $(r, s) \equiv (r, n-s)$
- Vulnerability was patched

Exercise 5

- Modify a block
- (r, s) ≡ (r, n-s)
- r = tx_data[47:79]
- s = tx_data[81:113]