PV181 Laboratory of security and applied cryptography

Public key crypto
Common math operations

Marek Sýs
syso@mail.muni.cz, A405

CRoCS
Centre for Research on Cryptography and Security

You will learn

- How to generate public/private key pair.
- Formats of cryptographic data
  - Base64,
  - ASN1 (PEM, DER)
- Mathematical operations used in public-key cryptography
  - Modular operations
  - Operations with points on Elliptic curve

Public key operations

- Modular operations:
  - Addition
  - Multiplication
  - Exponentiation
  - Inversion (uses Extended Euclid alg.)
- Elliptic curve operations:
  - Point addition
  - Point multiplication (e.g. point doubling 2*P = P+P)

Modular operations

- The result always in [0, m-1] for modulus m
- Addition, multiplication: $a + b \; mod \; m$, $a * b \; mod \; m$
  - In python ***a+b % m, a*b % m***
- Exponentiation: $a^e \; mod \; m$
  - In python **pow(a, e, m)**
  - Computed iteratively with modulo applied to decrease size of intermediate products
  - $a^{11} \; mod \; m \Rightarrow 11 = 1 + 2 + 8 \Rightarrow a^{11} = \; a^1 * a^2 * a^8$
    - $a^1, a^2 = a * a \; mod \; m, a^4 = a^2 * a^2 \; mod \; m, \; a^8, a^{16}$
    - $a^{11} \; mod \; m = \left((a^1 * a^2) mod \; m * a^8\right) \; mod \; m$

Modular inversion

- Inverse b^{-1} of b modulo m
  - b^{-1} such that $b^{-1} * b = b * b^{-1} = 1 \; mod \; m$
  - b and m must be coprime!
- Modular “division” using inverse:

$$\frac{a}{b} mod \; m = a * b^{-1} \; mod \; m$$

  - In python **(a * pow(a, -1, m)) %**
- Examples:
  - $3^{-1} \; mod \; 7 = 5$ since $3 * 5 \; mod \; 7 = 1$
  - $4^{-1} \; mod \; 10$ does not exist ($\gcd(4, 10) = 2$)

RSA

- Public N, e
  - Encryption – modular exponentiation
    - $E(m) = m^e \ mod \ N$
- Private e, d, p, q, N
  - Decryption – modular exponentiation
    - $D(c) = c^d \ mod \ N$
- Private
  - Decryption – modular exponentiation using CRT

Elliptic curve cryptography (ECC)

- Defined by parameters
  - $a,\ b,\ p,\ G,\ q$
- Groups of points (x, y)
  - $y^2 = x^3 + ax + b \ mod\ p$
- Operations with points:
  - Addition - complex wiki $(x_1, y_1) + (x_2, y_2) \neq (x_3, y_3)$
  - Multiplication vs addition – classical relationship
    - E.g. $3 * (x_1, y_1) = (x_1, y_1) + (x_1, y_1) + (x_1, y_1)$
    - Maximal multiplier q i.e
      - $k * (x_1, y_1) = (k\ mod\ q) * (x_1, y_1)$

Tasks

1. Tasks_bash.txt
2. encoding_tasks.py or encoding_tasks.ipynb
3. ASN1_basic.py
4. ASN1_dump.py

Cheatsheet for encodings – see docs

Solutions to tasks are provided in solutions folder