Crypto

Week 2

Thanks!

- Over 95% of these slides provided by Kuruwa

ToC

- Asymmetric cryptography
  - ……
  - Discrete Log
  - Elliptic Curve
- Digital Signature
- Lattices

Asymmetric cryptography (cont.)

How to build Public-Key Algorithms

- Asymmetric schemes are based on a "**one-way function**" f:
  - Computing $y = f(x)$ is computationally easy
  - Computing $x = f^{-1}(y)$ is computationally infeasible
- One-way functions are based on mathematically hard problems. Three main families:
  - **Factoring Integers** (RSA): Given a composite integer n, find its prime factors (Multiply two primes: easy)
  - **Discrete Logarithm** (Diffie-Hellman, Elgamal, DSA): Given a, y and m, find x such that $a^x = y \bmod m$ (Exponentiation a^x : easy)
  - **Elliptic Curves** (ECDH, ECDSA): Generalization of discrete logarithm

Discrete Logarithm

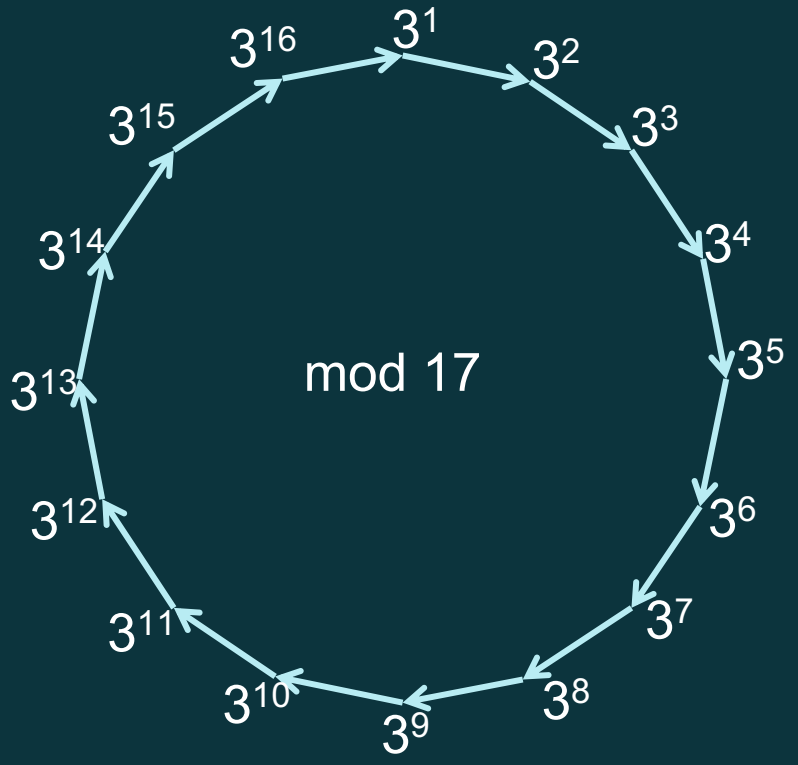
- Introduction
- Diffie-Hellman Key Exchange
- ElGamal Encryption
- Generalized DLP
- Attacks against DLP

The Discrete Logarithm Problem

- Given a finite cyclic group $\mathbb{Z}_p^*$ of order $p - 1$ and a primitive element $\alpha \in \mathbb{Z}_p^*$ and another element $\beta \in \mathbb{Z}_p^*$
- The DLP is the problem of determining the integer $1 \leq x \leq p - 1$ such that

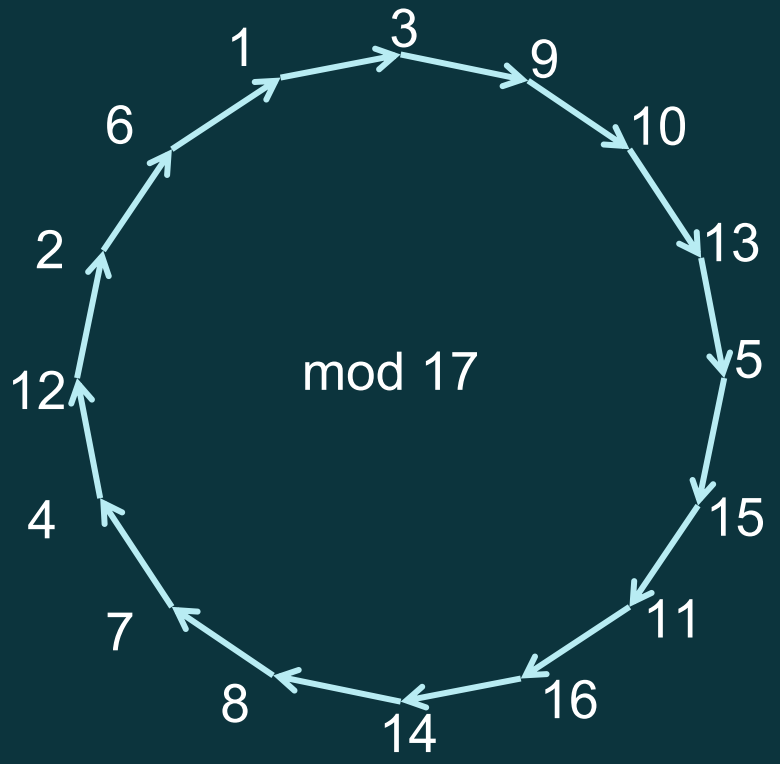
$$\alpha^x = \beta \pmod{p}$$

The Discrete Logarithm Problem



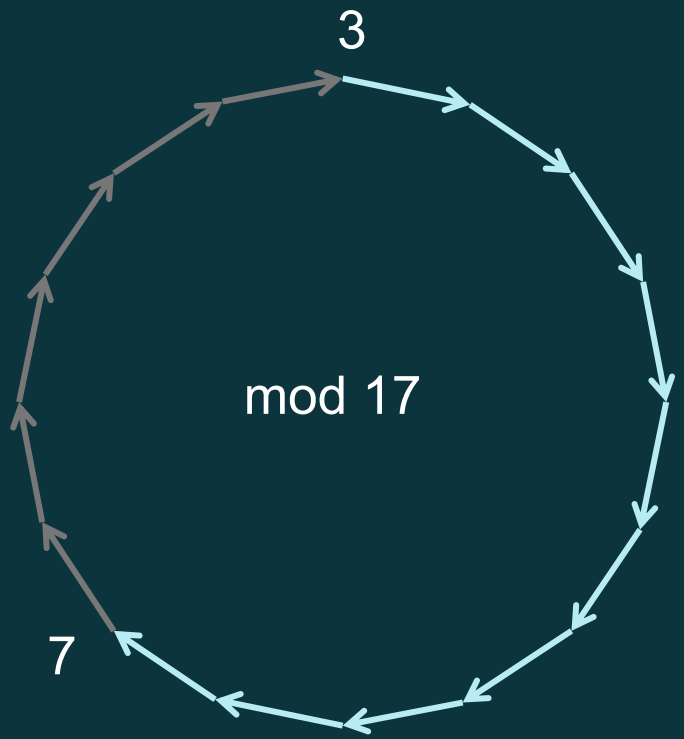
Generator : 3
Order: 16

The Discrete Logarithm Problem



Generator : 3
Order: 16

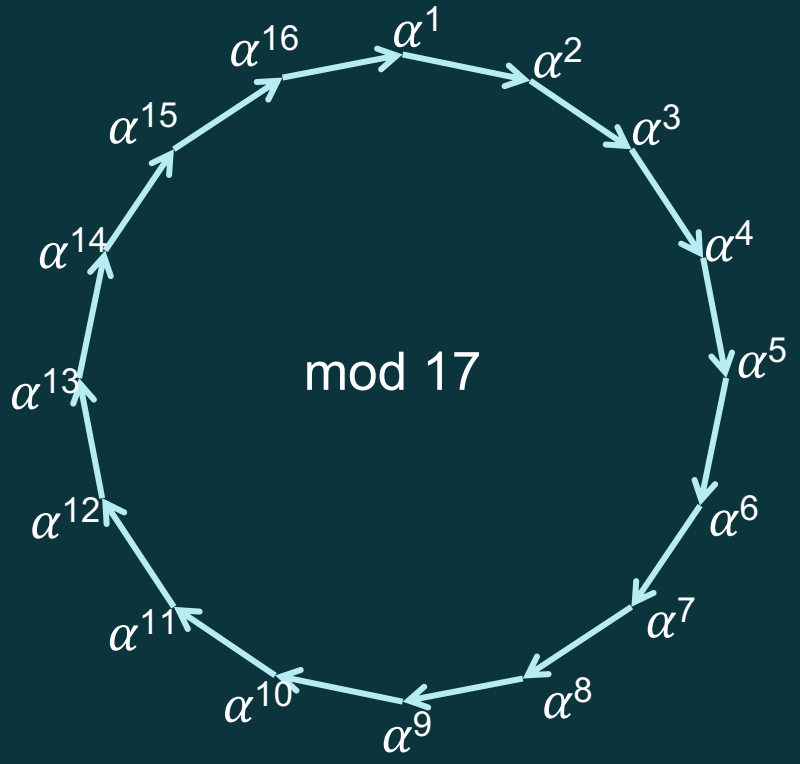
The Discrete Logarithm Problem



$3^X = 7 \bmod 17$

$X = ?$

The Discrete Logarithm Problem



Generator : α

Order: 16

Diffie-Hellman Key Exchange

- Set-up
  - Choose a large prime p
  - Choose an integer $\alpha \in \{2, 3, \ldots, p - 2\}$
  - Publish p and α

Diffie-Hellman Key Exchange

Choose random private key
$K_{prA} = a \in \{1, 2, \dots, p-1\}$

Choose random private key
$K_{prB} = b \in \{1, 2, \dots, p-1\}$

Compute
$A = \alpha^a \pmod p$

A →

← B

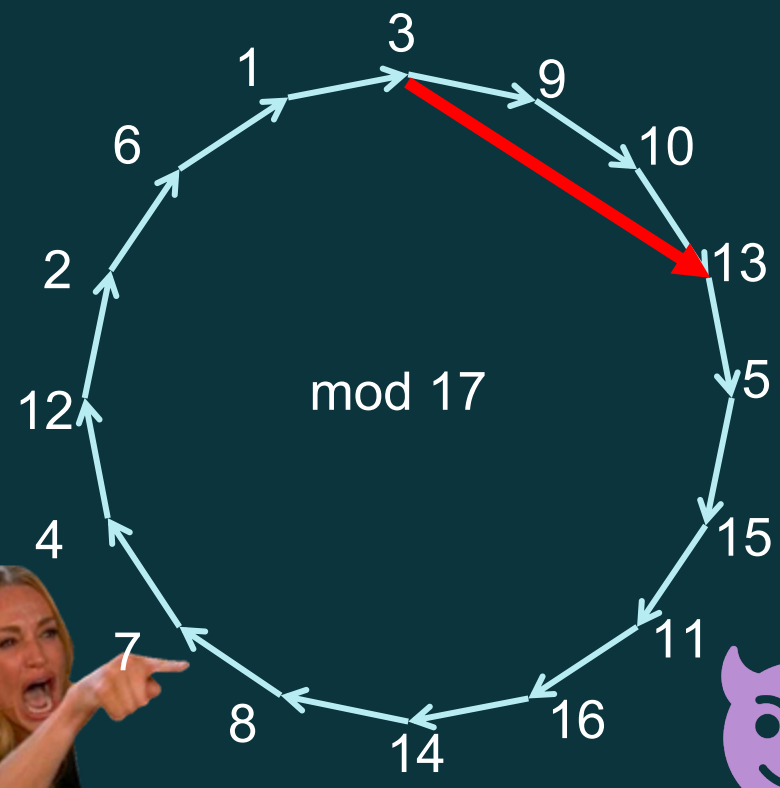
Compute
$B = \alpha^b \pmod p$

Caluculate common secret
$K = B^a = (\alpha^b)^a \pmod p$

Caluculate common secret
$K = A^b = (\alpha^a)^b \pmod p$

$y = AES_K(x)$ —y→ $x = AES^{-1}{}_K(y)$

The Discrete Logarithm Problem



The Discrete Logarithm Problem

???

The ElGamal Encryption Scheme

Choose b = K_{prB} ∈ {2, … , p−2}

Compute B = K_{pubB} = α^b (mod p)

(p, α, B)

Choose a = K_{prA} ∈ {2, … , p−2}

Compute the ephemeral key
A = K_E = K_{pubA} = α^a (mod p)

Compute the masking key
K_M = B^a (mod p)

Encrypt the message x
y = x × K_M (mod p)

(A, y)

Compute the masking key
K_M = K_E^b (mod p)

Decrypt the message
x = y × K_M^{-1} (mod p)

The ElGamal Encryption Scheme

- Encryption: $x * K_M = y \pmod{p}$
- Decryption: $x = y \times K_M^{-1} \pmod{p}$
- K_M : Same as Diffie-Hellman Key Exchange

Security

- Summary of records for computing discrete logarithms

Digits	Bit length	Date
58	193	1991
68	216	1996
85	282	1998
100	332	1999
120	399	2001
135	448	2006
160	532	2007
180	596	2014
232	768	2016
240	795	2019

Generalized DLP

- Generalized DLP
  - Let $(G, \circ)$ be an abelian group
  - Given $g, h \in G$, find x (if it exists) such that $g^x = h$
- The difficulty of this problem depends on the group G
  - Very easy: polynomial time algorithm
    - e.g. $(\mathbb{Z}_N, +)$
  - Rather hard: sub-exponential time algorithm
    - e.g. $(\mathbb{F}_p, \times)$
  - Very hard: exponential time algorithm
    - e.g. Elliptic Curve groups

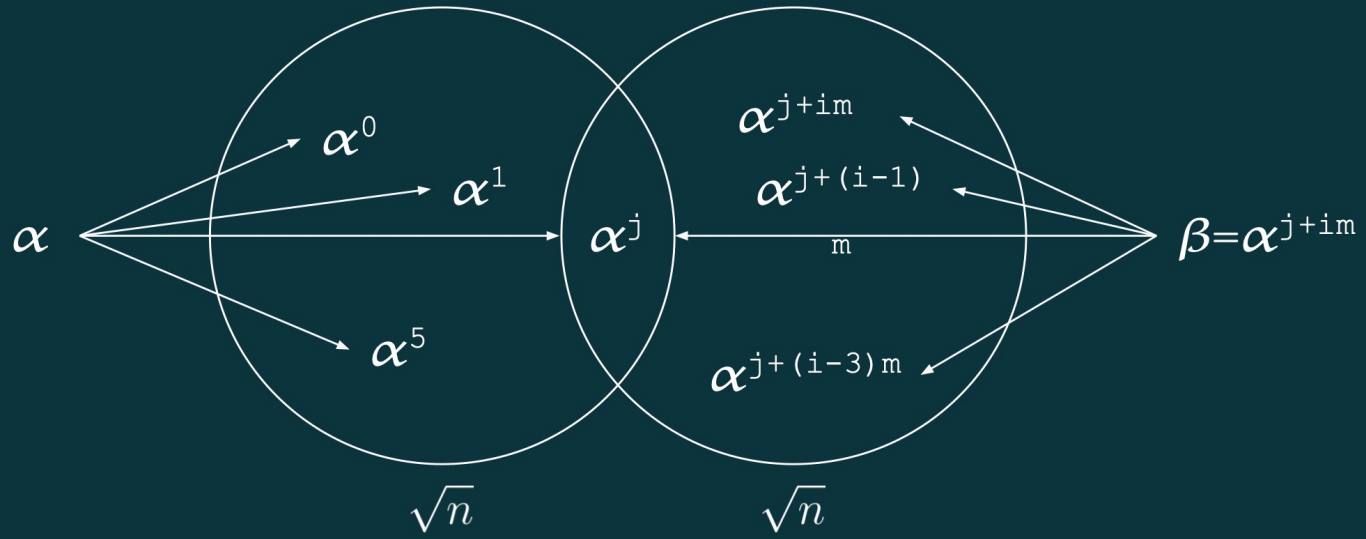
Attacks against the DLP

- Generic algorithms: Work in any cyclic group
  - Brute-Force Search
  - Baby-Step-Giant-Step
  - Pollard's Rho Method
  - Pohlig-Hellman Method

- Non-generic Algorithms: Work only in specific groups, in particular in $\mathbb{Z}_p^*$
  - The Index Calculus Method

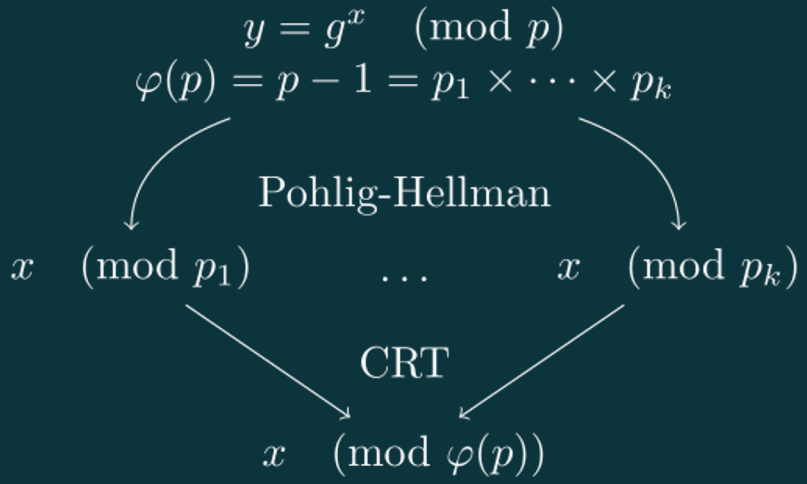
Baby-Step-Giant-Step

- We want to solve $\alpha^x = \beta$
- Rewrite x = im + j, where m = $\lceil \sqrt{n} \rceil$
  - $0 \le i < m$, $0 \le j < m$
  - $\alpha^j = \beta(\alpha^{-m})^i$



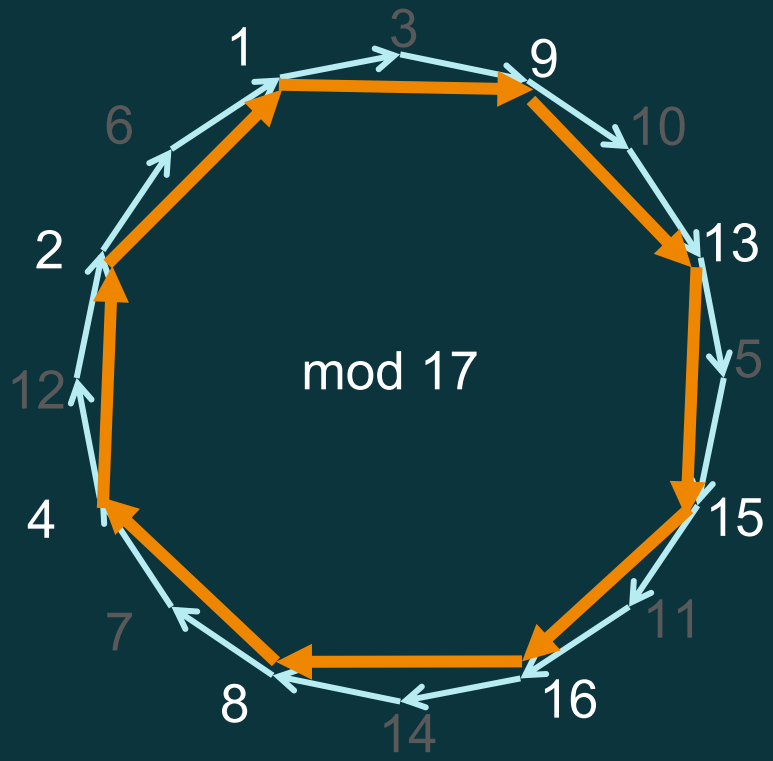
Pohlig-Hellman

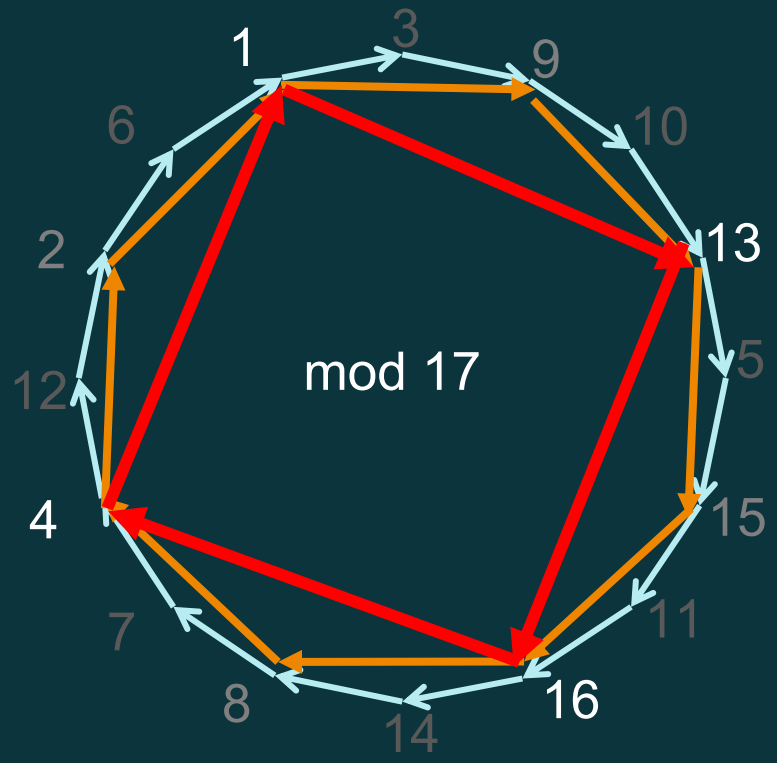
- If $p-1 = p_1 p_2 \ldots p_k$
  - $(g^{(p-1)/p_i})^{p_i} = 1$
  - $g_i = g^{(p-1)/p_i}$ has order p_i
  - $(g_i)^x = (g_i)^{(x \bmod p_i)} = y^{(p-1)/p_i} = h_i$
- Find x_i such that $(g_i)^x = h_i$
  - e.x. BSGS
  - $x_i = x \pmod{p_i}$
- Use CRT to recover x



- Runtime: $O(\sum_i (\log n + \sqrt{p_i}))$

Pohlig-Hellman





Pohlig-Hellman

- $3^x = y \pmod{17}$
- $(3^x)^2 = (3^2)^x = 9^x = y^2 \pmod{17}$

- Order 16 -> Order 8
- However, if x >= 8, using $9^x = y^2$, we can only recover x % 8

Pohlig-Hellman

Input: A cyclic group G of order $n = p_1 \ldots p_r$, having a generator g and an element h.

Output: A value x satisfying $g^x = h$

1. For all i where $1 \leq i \leq r$:
    1. Compute $g_i = g^{n/p_i}$
    2. Compute $h_i = h^{n/p_i}$
    3. Use BSGS to compute x_i such that $g_i^{x_i} = h_i$
2. Solve the CRT

$$x \equiv x_i \pmod{p_i} \quad \forall i \in \{1, \ldots, r\}.$$

1. Return x

Elliptic Curve

- Introduction
- ECDLP
- ECDH
- Attacks against ECDLP

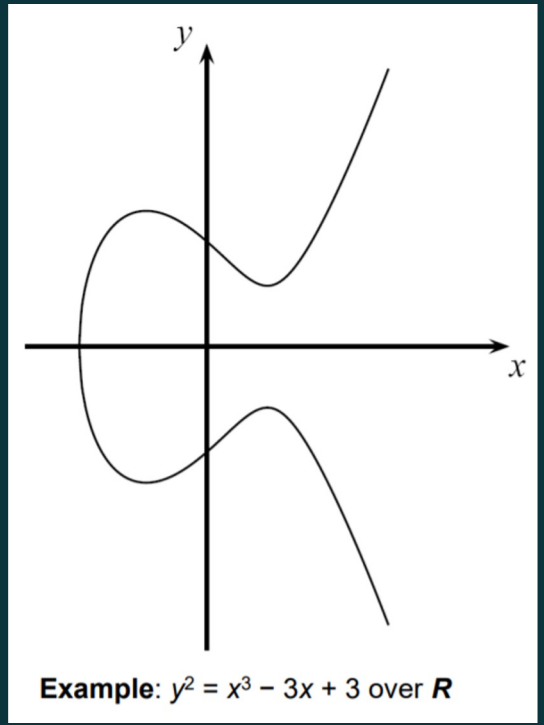
Elliptic Curve

- Elliptic curves are polynomials that define points based on the (simplified) Weierstraß equation:

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

for parameters a, b that specify the exact shape of the curve

- On the real numbers and with parameters $a, b \in \mathbb{R}$, an elliptic curve looks like this



Example: $y^2 = x^3 - 3x + 3$ over ***R***

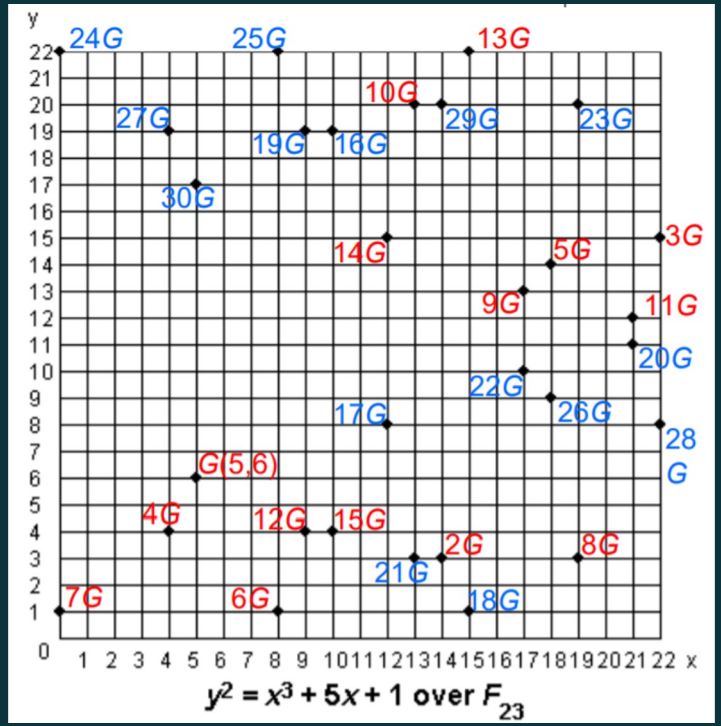
Elliptic Curve

- In cryptography, we are interested in elliptic curves modulo a prime p
- The elliptic curve over $\mathbb{Z}_p$, $p > 3$ is the set of all pairs $(x,y) \in \mathbb{Z}_p$ which fulfill

$$y^2 = x^3 + ax + b \pmod{p}$$

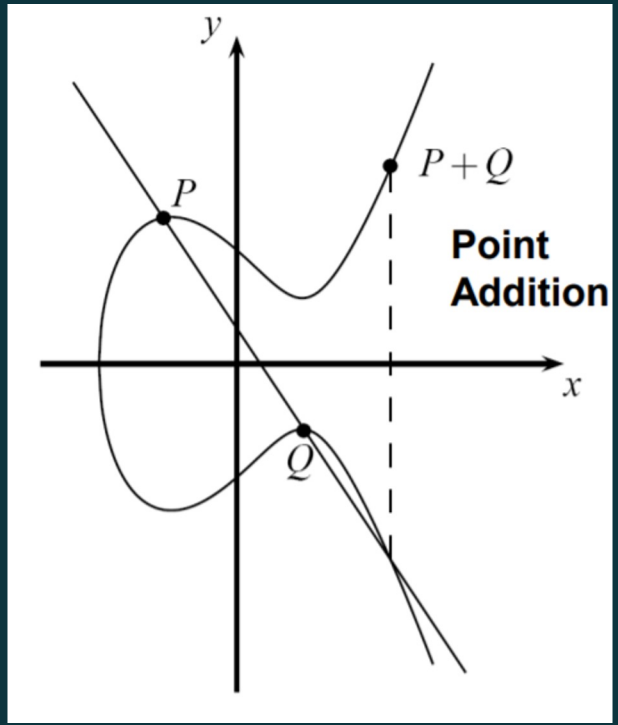
together with an imaginary point at infinity θ, where

$$4a^3 + 27b^2 \neq 0 \pmod{p}$$



Elliptic Curve

- Generating a group of points on elliptic curves based on point addition operation P + Q = R, i.e., $(x_P, y_P) + (x_Q, y_Q) = (x_R, y_R)$
- Geometric Interpretation of point addition operation
  - Draw straight line through P and Q; if P = Q use tangent line instead
  - Mirror third intersection point of drawn line with the elliptic curve along the x-axis



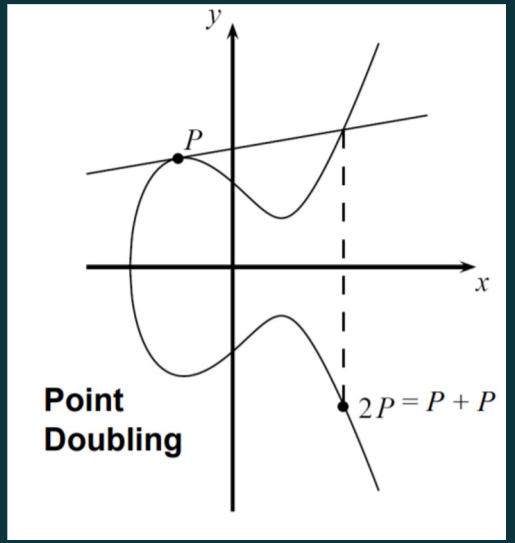
Elliptic Curve

- Elliptic Curve Point Addition and Doubling Formulas

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p \text{ (addition)} \\ \frac{3x_1^2 + a}{2y_1} \mod p \text{ (doubling)} \end{cases}$$

$$x_3 = s^2 - x_1 - x_2$$

$$y_3 = s(x_1 - x_3) - y_1$$



Elliptic Curve

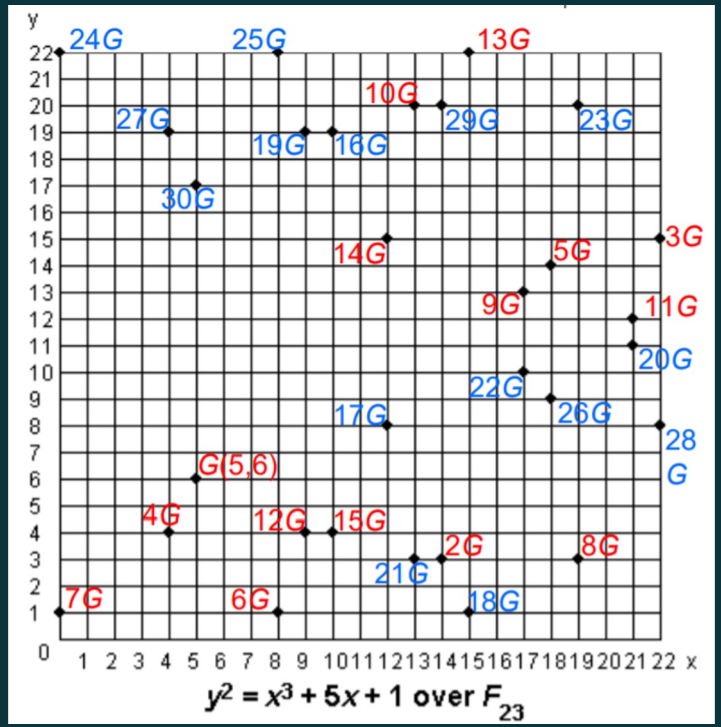
- Example: Compute 2G = G + G = (5, 6) + (5, 6) = (x_3 , y_3)

$$s = \frac{3x_1^2 + a}{2y_1} = (3 \cdot 5^2 + 5)(2 \cdot 6)^{-1} = 1 \cdot 2 = 22 \pmod{23}$$

$$x_3 = s^2 - x_1 - x_2 = 22^2 - 5 - 5 = 14 \pmod{23}$$

$$y_3 = s(x_1 - x_3) - y_1 = 22(5 - 14) - 6 = 3 \pmod{23}$$

- The points on an elliptic curve and the point at infinity Θ form cyclic groups
- This elliptic curve has order #E = |E| = 31



Number of Points on an Elliptic Curve

- Hasse's Theorem:
  - Given an elliptic curve modulo p, the number of points on the curve is denoted by #E and is bounded by

$$p + 1 - 2\sqrt{p} \leq \#E \leq p + 1 + 2\sqrt{p}$$

- The number of points is "close to" the prime p
  - To generate a curve with about 2^{160} points, a prime with a length of about 160 bits is required

ECDLP

- Cryptosystems rely on the hardness of the Elliptic Curve Discrete Logarithm Problem (ECDLP)
  - Given an element P and another element Q on an elliptic curve E. The ECDLP problem is finding the integer d, where $1 \leq d \leq \#E$ such that

$$P + P + \ldots + P = dP = Q$$

- Cryptosystems are based on the idea that d is large and kept secret, and attackers cannot compute it easily
- If d is known, an efficient method to compute the point multiplication dP is required to create a reasonable cryptosystem

Double-and-Add Algorithm

- Example $25P = (11001_2)P$
  - $\theta + \theta = \theta$ #DOUBLE
  - $\theta + P = P$ #ADD
  - $P + P = 2P$
  - $2P + P = 3P$
  - $3P + 3P = 6P$
  - #NO ADD
  - $6P + 6P = 12P$
  - #NO ADD
  - $12P + 12P = 24P$
  - $24P + P = 25P$

```
def Double_and_Add(d, P):
    bits = bin(d)[2:]
    Q = θ
    for bit in bits:
        Q = Q + Q
            if bit == "1":
            Q = Q + P
    return Q
```

Elliptic Curve Diffie-Hellman Key Exchange

- ECDH

Given a prime p, a suitable elliptic curve E and a point $P = (x_P, y_P)$

Choose random private key
$K_{prA} = a \in \{1, 2, \dots, \#E-1\}$

Choose random private key
$K_{prB} = b \in \{1, 2, \dots, \#E-1\}$

A →

Compute
$A = aP = (x_A, y_A)$

Compute
$B = bP = (x_B, y_B)$

← B

Caluculate common secret
$K = aB = a(bP)$

Caluculate common secret
$K = bA = b(aP)$

Parameter Choice

- E has smooth order
  - Pohlig-Hellman
- E has order equal to p (anomalous curve)
  - Transform the DLP to $(\mathbb{F}_p, +)$
  - Smart's Attack
- E is singular
  - Node: Transform the DLP to $(\mathbb{F}_p, \times)$
  - Cusp: Transform the DLP to $(\mathbb{F}_p, +)$

Pohlig-Hellman (on ECC)

Input: Elliptic Curve E of order $n = p_1 ... p_r$, having a generator G and an element P.

Output: A value d satisfying $dG = P$

1. For all i where $1 \leq i \leq r$:
    1. Compute $G_i = (n/p_i)G$
    2. Compute $P_i = (n/p_i)P$
    3. Use BSGS to compute d_i such that $d_i G_i = P_i$
2. Solve the CRT

$$d \equiv d_i \pmod{p_i} \quad \forall i \in \{1, \ldots, r\}.$$

1. Return d

Singular Curve

- A curve is singular if $4a^3 + 27b^2 = 0 \pmod{p}$
  - ECDLP becomes much easier if curve is singular
- There are two types of singular point
  - Node: $y^2 = (x - \alpha)^2(x - \beta)$
  - Cusp: $y^2 = x^3$

Node

- $y^2 = (x - \alpha)^2(x - \beta)$
- Define $\varphi(P(x, y)) = \frac{y + \sqrt{\alpha - \beta}(x - \alpha)}{y - \sqrt{\alpha - \beta}(x - \alpha)}$
- If we have homomorphism $\varphi(P + Q) = \varphi(P) \times \varphi(Q)$
  - $\varphi(dP) = \varphi(P)^d$
  - Reduce to DLP on $(\mathbb{F}_p, \times)$

Cusp

- $y^2 = x^3$
- Define $\varphi(P(x, y)) = x/y$
- If we have homomorphism $\varphi(P + Q) = \varphi(P) + \varphi(Q)$
  - $\varphi(dP) = d\varphi(P)$
  - Reduce to DLP on $(\mathbb{F}_p, +)$
  - $Q = dP \Rightarrow d = \varphi(Q)\varphi(P)^{-1}$

Invalid Curve Attack

- Lack of checking whether the Point is actually on the Curve
- Giving a Point outside the Curve with shorter Order (or maybe “smoother” order) will make the ECDLP problem easier.

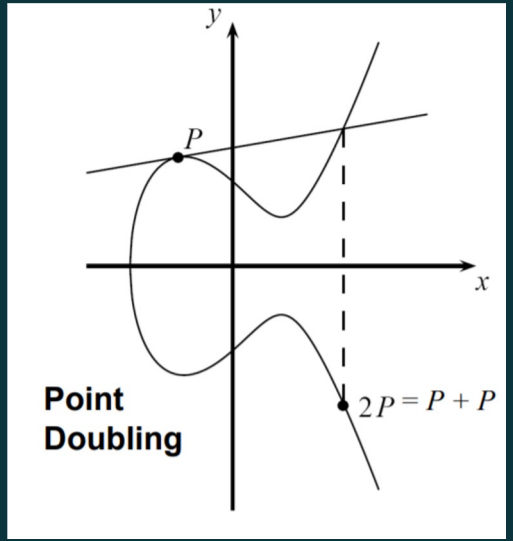
Recall that……

- Elliptic Curve Point Addition and Doubling Formulas

$$s = \begin{cases} \frac{y_2 - y_1}{x_2 - x_1} \mod p \text{ (addition)} \\ \frac{3x_1^2 + a}{2y_1} \mod p \text{ (doubling)} \end{cases}$$

$$x_3 = s^2 - x_1 - x_2$$

$$y_3 = s(x_1 - x_3) - y_1$$



```
Curve 1
p  = 65537
a = 17, b = 7
Order: 2^2 * 3^2 * 911
```

```
Curve 2
p  = 65537
a = 17, b = 13
Order: 2^2 * 5 * 11 * 13 * 23
```

Order of my curve is not smooth.
It is really safe.

Her curve may be safe.
But it is no use if the point is not on her curve.

Some resources

- Standard curve database: https://neuromancer.sk/std/
- https://wstein.org/edu/2010/414/projects/novotney.pdf

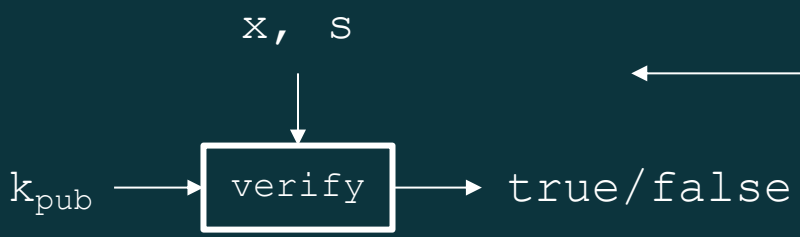
Lab: dlog

Digital Signature

Motivation

- Bob orders an Iphone15 from Alice
- After seeing the Iphone15, Bob states that he has never ordered it
- How can Alice prove towards a judge that Bob has ordered an Iphone15? (And that she did not fabricate the order herself)
  - Symmetric cryptography fails because both Alice and Bob can be malicious
  - Can be achieved with public-key cryptography

Digital Siganture



Main Idea

- For a given message x, a digital signature is appended to the message (just like a conventional signature)
- Only the person with the private key should be able to generate the signature
- The signature must change for every document
  - The signature is realized as a function with the message x and the private key as input
  - The public key and the message x are the inputs to the verification function

Objectives

- Integrity
  - Ensures that a message has not been modified in transit.
- Message Authentication
  - Ensures that the sender of a message is authentic. An alternative term is data origin authentication.
- Non-repudiation
  - Ensures that the sender of a message can not deny the creation of the message. (e.x. order of a GPU)

RSA Signature

- To generate the signature
  - Sign (encrypt) the message x with the private key

$$s = sig_{Kpr}(x) = x^d \bmod n$$

  - Append s to message x
- To verify the signature
  - Verify (decrypt) the signature with the public key

$$x' = ver_{Kpub}(s) = s^e \bmod n$$

  - If x = x', the signature is valid

RSA Signature Protocol

$k_{pub} = (n, e)$
$k_{pr} = d$

k_{pub} ←

(x, s) ←

$s = x^d \bmod n$

$x' = s^e \bmod n$

if $x' = x \longrightarrow$ valid

if $x' \neq x \longrightarrow$ invalid

Existential Forgery

(n, e)

(n, e)

$k_{pub} = (n, e)$
$k_{pr} = d$

Choose signature
$s \in \mathbb{Z}_n$
Compute message
$x = s^e \bmod n$

(x, s)

Verification:
$x' = s^e = x \bmod n$
⟶ Signature is valid

Existential Forgery

- An attacker can generate valid message-signature pairs (x, s)
- But an attack can only choose the signature s and NOT the message x
- Formatting the message x according to a **padding scheme** can be used to make sure that an attacker cannot generate valid (x, s) pairs

Digital Signature Algorithm (DSA)

- Key generation of DSA:
  - Generate a prime p with $2^{1023} < p < 2^{1024}$
  - Find a prime divisor q of p − 1 with $2^{159} < q < 2^{160}$
  - Find an integer α with ord(α) = q
    - $\alpha = g^{(p-1)/q} \neq 1 \bmod p$
  - Choose a random integer d with 0 < d < q
  - Compute $\beta = \alpha^d \bmod p$
- The keys are: $k_{pub} = (p, q, \alpha, \beta)$ and $k_{pr} = (d)$

Digital Signature Algorithm (DSA)

- Signature (message: $H < q$)
  - Choose an integer k_E as a random ephemeral key with $0 < k_E < q$
  - Compute $r = (\alpha^{k_E} \bmod p) \bmod q$
  - Compute $s = k_E^{-1}(H + d \times r) \bmod q$
    - In practice, H is hash of the message
- Verification
  - Compute auxiliary value $u_1 = s^{-1} \times H \bmod q$
  - Compute auxiliary value $u_2 = s^{-1} \times r \bmod q$
  - Compute $v = (\alpha^{u_1} \times \beta^{u_2} \bmod p) \bmod q$
    - if $v = r \longrightarrow$ siganature is valid
    - if $v \neq r \longrightarrow$ siganature is invalid

Correctness

$$s = (H + d \times r)k_E^{-1} \mod q$$

$$\Leftrightarrow k_E = s^{-1} \times H + d(s^{-1} \times r) \mod q$$

$$\Leftrightarrow k_E = u_1 + du_2 \mod q$$

$$\Leftrightarrow \alpha^{k_E} \mod p = \alpha^{u_1+du_2} \mod p$$

$$\Leftrightarrow (\alpha^{k_E} \mod p) \mod q = (\alpha^{u_1} \times \beta^{u_2} \mod p) \mod q$$

$$\Leftrightarrow r = v$$

Security

- DSA can achieve same security level as RSA scheme with less siganture length

p	q	length	security
1024	160	320	80
2048	224	448	112
3072	256	512	128

ECDSA

- Key generation of ECDSA:
  - Find a generator G on an elliptic curve E with prime order n
  - Choose a random integer d with $0 < d < n$
  - Compute $P = dG$
- The keys are: $k_{pub} = (E, G, n, P)$ and $k_{pr} = (d)$
  - Shorter private key and higher speed than DSA

ECDSA

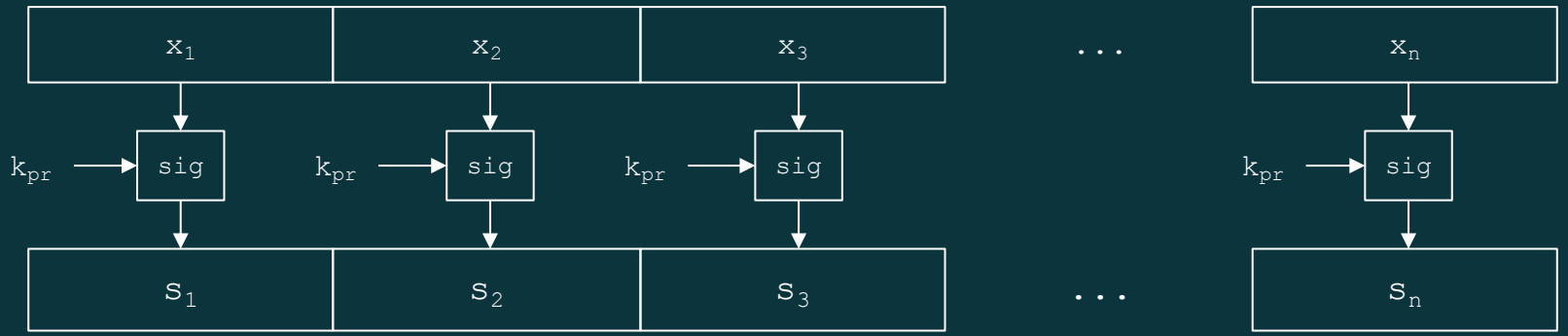
- Signature (message: H < n)
  - Choose an integer k_E as a random ephemeral key with $0 < k_E < n$
  - Calculate the curve point $(x_1, y_1) = k_E \times G$
  - Compute $r = x_1 \bmod n$
  - Compute $s = k_E^{-1}(H + d \times r) \bmod n$
- Verification
  - Compute auxiliary value $u_1 = s^{-1} \times H \bmod n$
  - Compute auxiliary value $u_2 = s^{-1} \times r \bmod n$
  - Compute $(x_1, y_1) = u_1G + u_2P$
    - if $x_1 = r \bmod n \longrightarrow$ siganature is valid
    - if $x_1 \neq r \bmod n \longrightarrow$ siganature is invalid

Sensitivity

- The entropy of the random value k_E are critical
- Example: sign two different messages, $k_1 = k_2$
  - $k_1 = s_1^{-1}H_1 + d(s_1^{-1}r_1) \bmod q$
  - $k_2 = s_2^{-1}H_2 + d(s_2^{-1}r_2) \bmod q$
  - $d = (s_1^{-1}H_1 - s_2^{-1}H_2) / (s_2^{-1}r_2 - s_1^{-1}r_1)$

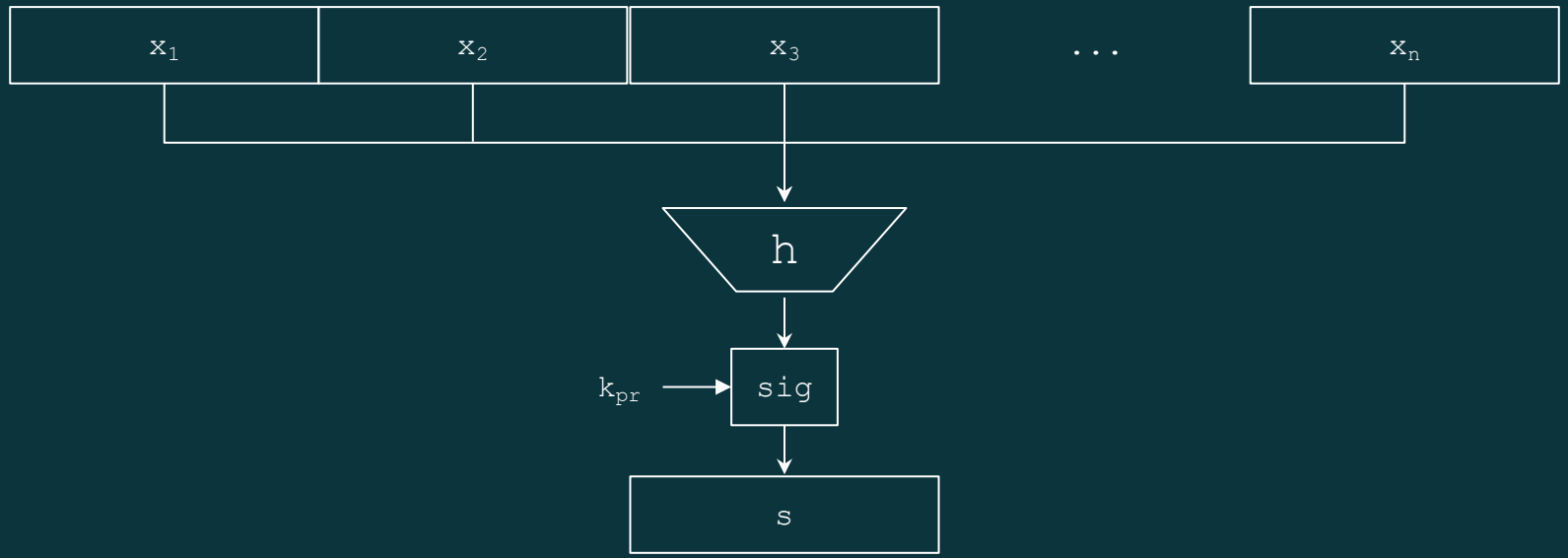
Hash - Motivation

- Naive signing of long messages generates a signature of same length.



- Solution
  - Instead of signing the whole message, sign only a digest (hash)

Digital Signature with Hash Function



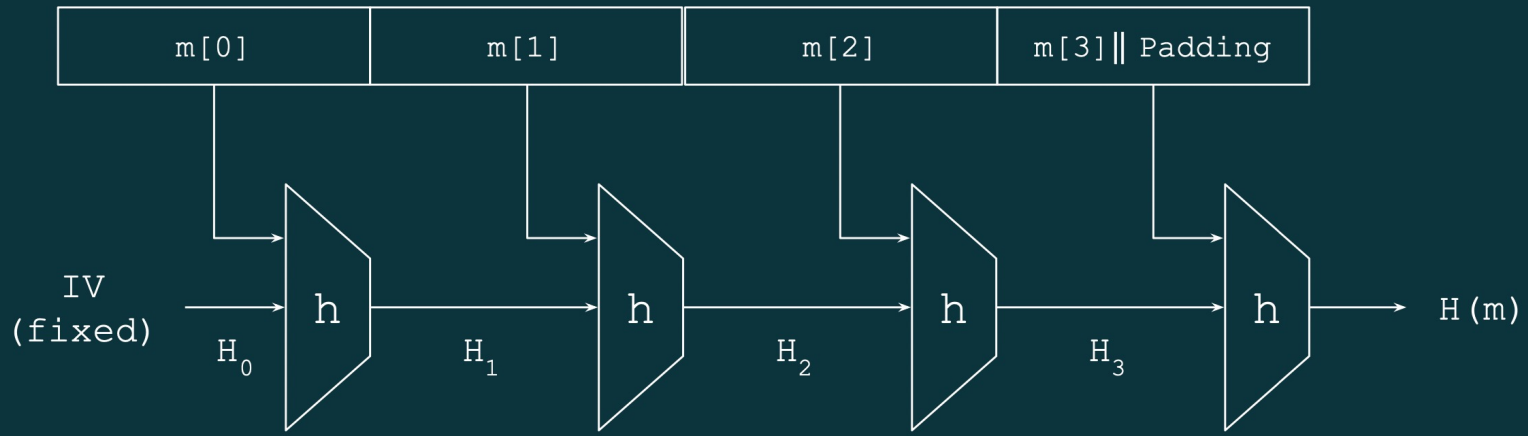
Hash - Properties

MD5, SHA-1, SHA-256, SHA-512 ……

- Collision Resistance
- One-wayness
- Input arbitrary length message, always output same length result

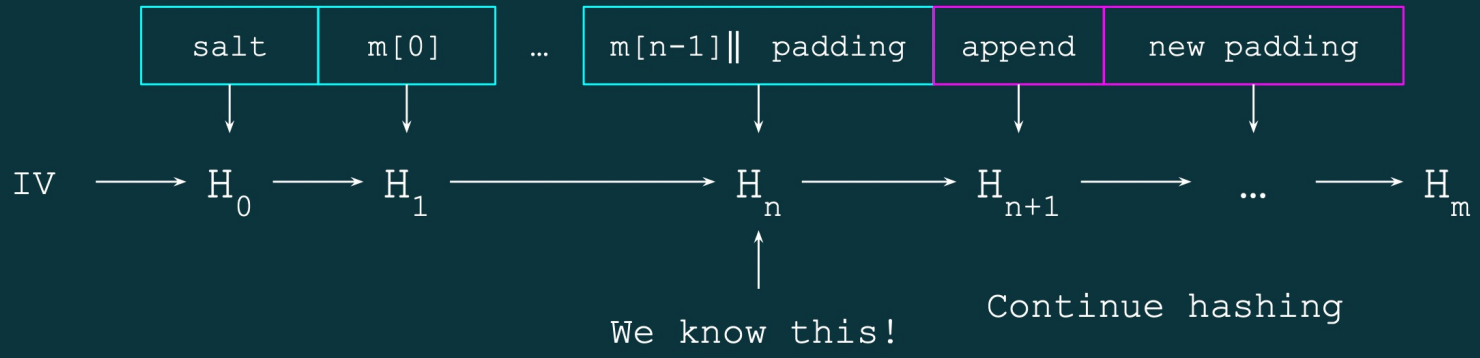
Merkle–Damgård construction

- Used in the design of many popular hash algorithms such as MD5, SHA-1 and SHA-2



Length Extension Attack

- Continue calculating hash after appending extra message
- New plaintext is message‖ padding‖ append



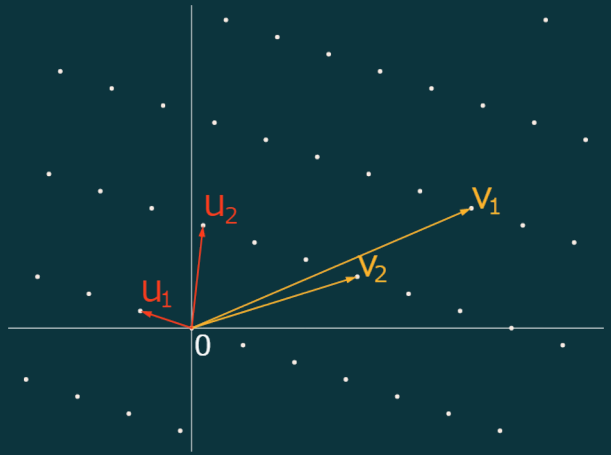
Lab: Signature

Lattices

Lattices

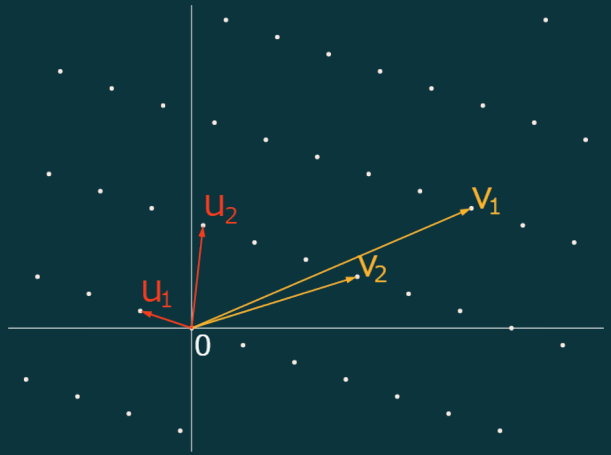
- Let $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n \in \mathbb{R}^m$ be a set of linearly independent vectors
- The **lattice** L generated by $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n$ is the set of linear combinations with coefficients in $\mathbb{Z}$,

$$L = \{a_1\mathbf{v}_1 + a_2\mathbf{v}_2 + \ldots + a_n\mathbf{v}_n \mid a_1, a_2, \ldots, a_n \in \mathbb{Z}\}$$



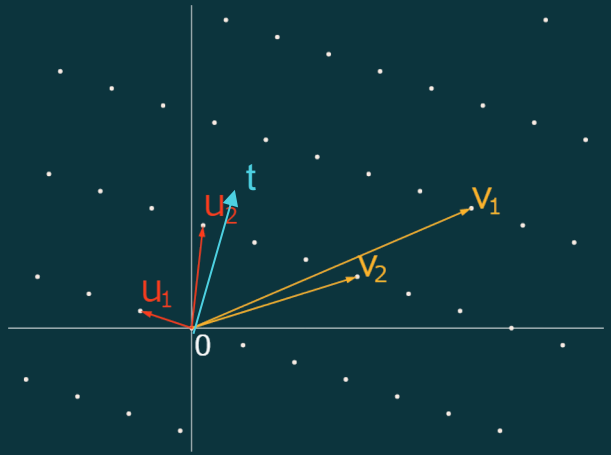
Shortest Vector Problem (SVP)

- The basis of lattice is not unique
- Given a basis of L, find the shortest vector in L
  - SVP is NP-hard



Closest Vector Problem (CVP)

- Given a basis of L, and an arbitrary vector **t**.
- Find the closest vector to **t** in L
  - CVP is also NP-hard

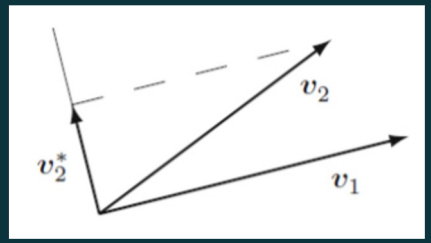


Gaussian Lattice Reduction

- Suppose that $L \subset \mathbb{R}^2$ is a 2-dimensional lattice with basis vectors $\mathbf{v}_1$, $\mathbf{v}_2$
  - May assume $\| \mathbf{v}_1 \| < \| \mathbf{v}_2 \|$
- If allowed to subtract any multiple of $\mathbf{v}_1$, then replace $\mathbf{v}_2$ with the vector

$$v_2^* = v_2 - \frac{v_1 \cdot v_2}{\|v_1\|^2} v_1$$

  - $\mathbf{v}_2^*$ is orthogonal to $\mathbf{v}_1$
  - But $\mathbf{v}_2^*$ is unlikely to be in L



- So the best is to replace $\mathbf{v}_2$ with the vector $\mathbf{v}_2 - m\mathbf{v}_1$ with

$$m = \left\lfloor \frac{v_1 \cdot v_2}{\|v_1\|^2} \right\rceil$$

Gaussian Lattice Reduction (cont.)

- If $\| \mathbf{v}_1 \| < \| \mathbf{v}_2 \|$, then stop
- Otherwise, swap $\mathbf{v}_1$ and $\mathbf{v}_2$ and repeat the process

Loop

If $\|v_2\| < \|v_1\|$, swap v_1 and v_2.

Compute $m = \lceil v_1 \cdot v_2 / \|v_1\|^2 \rfloor$.

If $m = 0$, return the basis vectors v_1 and v_2.

Replace v_2 with $v_2 - mv_1$.

Continue Loop

- When the algorithm terminates
  - The vector $\mathbf{v}_1$ is a shortest nonzero vector in L
  - The algorithm solves SVP

Lenstra–Lenstra–Lovász Algorithm (LLL)

- Given a lattice L, LLL solves approximated SVP in polynomial time
- The shortest vector **v** it found satisfies
  $\| \mathbf{v} \| \leq 2^{(n-1)/4} |\det L|^{1/n}$
- On average, LLL achieves
  $\| \mathbf{v} \| \leq 1.02^{n} |\det L|^{1/n}$

INPUT
a lattice basis $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$
a parameter δ with $\frac{1}{4} < \delta < 1$, most commonly $\delta = \frac{3}{4}$

PROCEDURE
$\mathbf{B}^* \leftarrow \text{GramSchmidt}(\{\mathbf{b}_0, \ldots, \mathbf{b}_n\}) = \{\mathbf{b}_0^*, \ldots, \mathbf{b}_n^*\}$; *and do not normalize*
$\mu_{i,j} \leftarrow \dfrac{\langle \mathbf{b}_i, \mathbf{b}_j^* \rangle}{\langle \mathbf{b}_j^*, \mathbf{b}_j^* \rangle}$; *using the most current values of* $\mathbf{b}_i$ and $\mathbf{b}_j^*$
$k \leftarrow 1$;
while $k \leq n$ **do**
  for j **from** $k-1$ **to** 0 **do**
    if $|\mu_{k,j}| > \frac{1}{2}$ **then**
      $\mathbf{b}_k \leftarrow \mathbf{b}_k - \lfloor \mu_{k,j} \rceil \mathbf{b}_j$;
      Update $\mathbf{B}^*$ *and the related* $\mu_{i,j}$*'s as needed.*
      (The naive method is to recompute $\mathbf{B}^*$ *whenever* $\mathbf{b}_i$ *changes:*
      $\mathbf{B}^* \leftarrow \text{GramSchmidt}(\{\mathbf{b}_0, \ldots, \mathbf{b}_n\}) = \{\mathbf{b}_0^*, \ldots, \mathbf{b}_n^*\}$;)
    end if
  end for
  if $\langle \mathbf{b}_k^*, \mathbf{b}_k^* \rangle \geq \left(\delta - \mu_{k,k-1}^2\right) \langle \mathbf{b}_{k-1}^*, \mathbf{b}_{k-1}^* \rangle$ **then**
    $k \leftarrow k + 1$;
  else
    Swap $\mathbf{b}_k$ and $\mathbf{b}_{k-1}$;
    Update $\mathbf{B}^*$ *and the related* $\mu_{i,j}$*'s as needed.*
    $k \leftarrow \max(k-1, 1)$;
  end if
end while
return $\mathbf{B}$ the LLL reduced basis of $\{b_0, \ldots, b_n\}$

OUTPUT
the reduced basis $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n \in \mathbb{Z}^m$

Babai's nearest plane algorithm

- Find reduced Basis $\mathbf{B} = \{\mathbf{b}_1, \ldots, \mathbf{b}_n\}$
- for k in range(n, 0, -1):
  - Consider the hyperplane generated by the first k − 1 lattice vectors.
  - Make the k^{th} vector as close as possible to the target vector **t**.

(EC)DSA - Known High Bits of k

- Two singature (r_1, s_1), (r_2, s_2), both use small nonces k
  - $s_1 \equiv k_1^{-1}(h_1 + dr_1) \bmod n$
  - $s_2 \equiv k_2^{-1}(h_2 + dr_2) \bmod n$
- Eliminate the variable d
  - $k_1 - s_1^{-1}s_2r_1r_2^{-1}k_2 + s_1^{-1}r_1h_2r_2^{-1} - s_1^{-1}h_1 \equiv 0 \bmod n$
- Let $t = -s_1^{-1}s_2r_1r_2^{-1}$, $u = s_1^{-1}r_1h_2r_2^{-1} - s_1^{-1}h_1$
  - $k_1 + tk_2 + u \equiv 0 \bmod n$
- We wish to solve k_1 and k_2, both small.
  - Let $|k_1|, |k_2| < K$

(EC)DSA - Known High Bits of k

- Construct lattice basis

$$B = \begin{bmatrix} n & 0 & 0 \\ t & 1 & 0 \\ u & 0 & K \end{bmatrix}$$

- The vector **v** = $(-k_1, k_2, K)$ is in this lattice
  - $(-q, k_2, 1)B = (-k_1, k_2, K)$
- Can find **v** when
  - $K < (nK)^{1/3}$
  - $\Rightarrow K < n^{1/2}$

Coppersmith's Method

- **Input:** $f(x) \in \mathbb{Z}[x]$, $N \in \mathbb{Z}$
- **Output:** r s.t. $f(r) \equiv 0 \bmod N$
- **Intermediate output:** $Q(x)$ such that $Q(r) = 0$ over $\mathbb{Z}$
  - $Q(x) = s(x)f(x) + t(x)N$
  - $Q(r) \equiv 0 \bmod N$ by construction
  - If $|r| \leq R$, then we can bound

$$|Q(r)| = |Q_n r^n + \ldots + Q_3 r^3 + Q_2 r^2 + Q_1 r + Q_0|$$

$$\leq |Q_n| R^n + \ldots + |Q_3| R^3 + |Q_2| R^2 + |Q_1| R + |Q_0|$$

  - If $|Q(r)| < N$ and $Q(r) \equiv 0 \bmod N$, then $Q(r) = 0$
- We want a Q in our lattice with short coefficient vector!

Coppersmith's Method

1. Construct a matrix of coefficient vectors of elements of <f(x), N>
2. Run LLL algorithm on this matrix
3. Construct a polynomial Q from the shortest vector output
4. Factor Q to find its roots

Theorem (Coppersmith)

Given a polynomial f of degree d and N, we can efficiently find all roots r satisfying $f(r) \equiv 0 \bmod N$ when $|r| < N^{1/d}$.

RSA - Stereotyped Messages

- Known most of the message, ex: padding
  - $m = a + x_0,\ x_0 \leq R$
  - $c = m^3 = (a + x_0)^3 \bmod n$
- x_0 is a small root of $f(x) = (a + x)^3 - c \pmod n$
- Let the biggest degree of Q be 3
  - $Q(x) = c_3(x^3 + 3ax^2 + 3a^2x + (a^3 - c)) + c_2Nx^2 + c_1Nx + c_0N$
  - $Q(x_0) \leq c_3(R^3 + 3aR^2 + 3a^2R + (a^3 - c)) + c_2NR^2 + c_1NR + c_0N$

RSA - Stereotyped Messages (cont.)

- Construct lattice basis

$$\begin{array}{c} Q_3 \quad Q_2 \quad\quad Q_1 \quad\quad Q_0 \\ \begin{bmatrix} R^3 & 3aR^2 & 3a^2R & a^3 - c \\ & NR^2 & & \\ & & NR & \\ & & & N \end{bmatrix} \end{array} \quad \begin{array}{l} c_3(x^3 + 3ax^2 + 3a^2x + (a^3 - c)) \\ c_2Nx^2 \\ c_1Nx \\ c_0N \end{array}$$

  - dim L = 4, det L = N^3R^6
- Ignoring approximation factor, we can solve when
  - $|Q(x_0)| \leq |\mathbf{v}| \leq |\det L|^{1/4} < N$
  - $\Rightarrow (N^3R^6)^{1/4} < N$
  - $\Rightarrow R < N^{1/6}$

RSA - Stereotyped Messages (cont.)

- Construct lattice basis

$$\begin{bmatrix} R^3 & 3aR^2 & 3a^2R & a^3 - c \\ & NR^2 & & \\ & & NR & \\ & & & N \end{bmatrix}$$

  - dim L = 4, det L = N^3R^6
- Ignoring approximation factor, we can solve when
  - $|Q(x_0)| \leq |\mathbf{v}| \leq |\det L|^{1/4} < N$
  - $\Rightarrow (N^3R^6)^{1/4} < N$
  - $\Rightarrow R < N^{1/6}$

Achieving the Coppersmith Bound

- Generate lattice from subset of $\langle f(x), N \rangle^k$
- Allow higher degree polynomials

$$\begin{bmatrix} R^6 & 6aR^5 & 15a^2R^4 & (20a^3-2c)R^3 & (15a^4-6ac)R^2 & (6a^5-6a^2c)R & a^6-2a^3c+c^2 \\ & R^5N & 3aR^4N & 3a^2R^3N & (a^3-c)R^2N & & \\ & & R^4N & 3aR^3N & 3a^2R^2N & (a^3-c)RN & \\ & & & R^3N & 3aR^2N & 3a^2RN & (a^3-c)N \\ & & & & R^2N^2 & & \\ & & & & & RN^2 & \\ & & & & & & N^2 \end{bmatrix}$$

$(R^{21}N^9)^{1/7} < N^2 \Rightarrow R < N^{5/21}$

RSA - Known High Bits of p

- Known large portion of MSBs of one factor
  - $n = pq$, $p = a + x_0$, known a, $x_0 \leq R$
- x_0 is a small roots of $f(x) = a + x \pmod p$
- Construct $Q(x) = 0 \pmod p$
  - $Q(x) = c_1 x(a + x) + c_2(a + x) + N$
  - $Q(x_0) \leq c_1(R^2 + aR) + c_2(R + a) + N$

Theorem (Howgrave-Graham)

Given degree d polynomial f, integer N, we can find roots r modulo divisors B of N satisfying $f(r) \equiv 0 \bmod B$ for $|B| > N^{\beta}$, when $|r| < N^{\beta 2/d}$

RSA - Known High Bits of p

- Construct lattice basis

$$\begin{bmatrix} R^2 & Ra & \\ & R & a \\ & & N \end{bmatrix}$$

  - dim L = 3, det L = NR^3
- Can find the root when
  - $(NR^3)^{1/3} < p = N^{1/2}$
  - $\Rightarrow R < N^{1/6}$

RSA - Partial Key Recovery

- Can factor given 1/2 bits of p [Coppersmith 96]
- Can factor given 1/4 bits of d [Boneh Durfee Frankel 98]
- Can factor given 1/2 bits of d mod (p-1) [Blömer May 03]

Some Resources

- https://github.com/josephsurin/lattice-based-cryptanalysis/blob/main/tutorial.pdf

Lab: Coppersmith