HỌC VIỆN KỸ THUẬT MẬT MÃ
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

THUẬT TOÁN TRONG AN TOÀN THÔNG TIN

Information Security Algorithms

MỤC TIÊU

Trang bị kiến thức về một số thuật toán để thực hiện các tính toán hiệu quả ứng dụng trong an toàn thông tin, đặc biệt là trong mật mã khóa công khai và trong phát hiện tấn công, mã độc.

GIỚI THIỆU HỌC PHẦN

THỜI LƯỢNG: 2tc

- 18 tiết lý thuyết
- 24 tiết thực hành

ĐÁNH GIÁ KẾT QUẢ HỌC TẬP

- Điểm chuyên cần
  - Đi học đầy đủ, đúng giờ
  - Tham gia xây dựng bài
- Kiểm tra giữa kỳ: thi viết
- Thi kết thúc học phần: Thực hành lập trình trên máy

GIỚI THIỆU HỌC PHẦN

Guide to Elliptic Curve Cryptography, Springer, 2004 (Chapter 2. Finite Field Arithmetic)
Darrel Hankerson, Alfred Menezes and Scott Vanstone,

Prime Numbers - A Computational Perspective (2nd edition), Springer, 2005
Richard Crandall and Carl Pomerance,

Algorithms and Teory of Computation Handbook: General Concepts and Techniques, CRC Press, 2010 (Chapter 13. Pattern Matching in Strings)
Mikhail J. Atallah and Marina Blanton

Và các tài liệu khác
Google ....

HỌC VIỆN KỸ THUẬT MẬT MÃ
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

NỘI DUNG

HỌC VIỆN KỸ THUẬT MẬT MÃ
ACADEMY OF CRYPTOGRAPHY TECHNIQUES

CHƯƠNG 01

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p

BÀI 01 - MỤC TIÊU

- Nắm được, cài đặt được các phép tính toán hiệu quả trên số nguyên lớn

BÀI 01 - MỤC TIÊU

- VD: Thời gian cần thiết để phân tích số nguyên n ra thừa số nguyên tố bằng thuật toán nhanh nhất hiện nay:

Số chữ số thập phân	Số phép tính bít	Thời gian
50	$1,4.\ 10^{10}$	3,9 giờ
75	$9.\ 10^{12}$	104 ngày
100	$2,3.\ 10^{15}$	74 năm
200	$1,2.\ 10^{23}$	$3,8.10^{9}$ năm
300	$1,5.\ 10^{29}$	$4,9.10^{15}$ năm
500	$1,3.\ 10^{39}$	$4,2.10^{25}$năm

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_P

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Giới thiệu về các trường hữu hạn

- Trường là một tập hợp với 2 phép toán (+, .) thỏa mãn các tính chất số học thông thường:
  - (F, +) là nhóm Abel với phép cộng
  - $(F\backslash\{0\}, .)$ là nhóm abel với phép nhân
  - Tính phân phối: $(a+b).c = a.c + b.c\ \forall a, b, c \in F$
- Trường hữu hạn (còn gọi là trường Galois) là những trường có hữu hạn số phần tử, số này gọi là bậc của trường đó.

Giới thiệu về các trường hữu hạn

- ...
- Các phép toán trên trường hữu hạn:
  - Có thể nói là có các phép toán cộng, trừ, nhân, chia số khác 0
  - Phép trừ được coi như là cộng với số đối của phép cộng

$$a - b = a + (-b)$$

  - Phép chia là nhân với số đối của phép nhân

$$a/\ b = a.b^{-1}$$

Giới thiệu về các trường hữu hạn

- Số lượng phần tử của một trường hữu hạn được gọi là cấp hoặc bậc của nó.
- Trường hữu hạn F cấp q nếu và chỉ nếu q là lũy thừa nguyên tố p^m (trong đó p là số nguyên tố, m là số nguyên dương). Nếu m = 1 thì F được gọi là trường nguyên tố, nếu $m \geq 2$ F được gọi là trường mở rộng.

Giới thiệu về các trường hữu hạn

- Trường nguyên tố F_p = {0, 1, …, p – 1} với các phép toán (+, .) thực hiện theo modulo p.
  - Ví dụ:
    - F_{29} = {0, 1, 2, …, 28}
      - Phép toán cộng: 17 + 20 = 8 vì 37 mod 29 = 8
      - Phép trừ: 17 – 20 = 26 vì – 3 mod 29 = 26
      - Phép nhân: 17.20 = 21 vì 340 mod 29 = 21
      - Phép lấy nghịch đảo: 17^{-1} = 12 vì 17.12 mod 29 = 1.

Giới thiệu về các trường hữu hạn

- Trường nhị phân $F_{2^m} = \{a_{m-1}z^{m-1} + a_{m-2}z^{m-2} + \cdots +$

Giới thiệu về các trường hữu hạn

- Ví dụ: F_{2^4} gồm 16 phần tử là các đa thức nhị phân có bậc cao nhất là 3

0	z^2	z^3	$z^3 + z^2$
1	$z^2 + 1$	$z^3 + 1$	$z^3 + z^2 + 1$
z	$z^2 + z$	$z^3 + z$	$z^3 + z^2 + z$
z + 1	$z^2 + z + 1$	$z^3 + z + 1$	$z^3 + z^2 + z + 1$

Giới thiệu về các trường hữu hạn

- Ví dụ một số phép toán trong F_{2^4} với đa thức rút gọn f(z) = $z^4 + z + 1$.
  - Phép cộng: $(z^3 + z^2 + 1) + (z^2 + z + 1) = z^3 + z$
  - Phép trừ: $(z^3 + z^2 + 1) - (z^2 + z + 1) = z^3 + z$. Lưu ý, vì – 1 = 1 trong F_2 (ta có – a = a với mọi a ∈ F_{2^m}).
  - Phép nhân: $(z^3 + z^2 + 1) . (z^2 + z + 1)$ = ?

    vì $(z^3 + z^2 + 1) . (z^2 + z + 1) = z^5 + z + 1$ và $z^5 + z + 1 \bmod z^4 + z + 1 = z^2 + 1$
  - Nghịch đảo: $(z^3 + z^2 + 1)^{-1} = z^2$ vì $(z^3 + z^2 + 1) . z^2 \bmod z^4 + z + 1 = 1$

Giới thiệu về các trường hữu hạn

- VD: Input: $a(z) = z^2 + z + 1$; $f(z) = z^3 + z + 1$

q	r	x	y	a	b	x_2	x_1	y_2	y_2
-	-	-	-	$x^3 + x + 1$	$x^2 + x + 1$	1	0	0	1
x + 1	x	1	x + 1	$x^2 + x + 1$	x	0	1	1	x + 1
x + 1	1	x + 1	x^2	x	1	1	x + 1	x + 1	x^2
x	0	$x^2 + x + 1$	1	1	Ω	x + 1	$x^2 + x + 1$	x^2	1

⇒ Output: $a^{-1}(x) = x^2$

Giới thiệu về các trường hữu hạn

- Nhóm nhân của trường hữu hạn:
  - Các phần tử khác 0 của trường hữu hạn F_q, KH: F_q^* là một nhóm cyclic với phép nhân, do đó tồn tại một phần tử sinh b ∈ F_q^* sao cho: F_q^* = {b^i : 0 ≤ i ≤ q – 2}
  - Cấp của phần tử a ∈ F_q^* là số nguyên dương nhỏ nhất t sao cho $a^t = 1$.

    Vì F_q^* là nhóm cyclic nên t là ước của q – 1.

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Phép tính cộng và trừ

- Các thuật toán cộng, trừ, nhân, chia, ..giới thiệu trong chương này phù hợp với triển khai phần mềm
- Ta giả thiết nền tảng triển khai có kiến trúc W – bit trong đó W là bội số của 8 (phổ biến là 64 – 32 bit), các hệ thống máy có công suất thấp có thể có W nhỏ hơn, VD: hệ thống nhúng W = 16 bit, thẻ thông minh W = 8 bit.
- Các bit của một W-bit là từ U được đánh số từ phải qua trái bắt đầu từ 0 đến W -1.

Phép tính cộng và trừ

- Ta có F_p = {0 … p – 1}.
- Tính m = $\lceil log_2 p \rceil$ là độ dài bit của p và t = $\lceil m/W \rceil$ là độ dài từ của p
- Biểu diễn của phần tử a được lưu trữ trong một mảng A = (A[t – 1], …, A[2], A[1], A[0]) của t các từ W bit, trong đó bit ngoài cùng bên phải của A[0] là bit có trọng số thấp nhất.

A[t – 1]	…	A[2]	A[1]	A[0]

- Biểu diễn a ∈ F_p như một mảng A của các từ W-bit:
  - $a = 2^{(t-1)W} A[t-1] + \ldots + 2^{2W} A[2] + 2^{W} A[1] + A[0]$

Phép tính cộng và trừ

- **Ví dụ:** cho W= 8, xét $F_{2147483647}$, hãy biểu diễn số a = 23456789 dưới dạng mảng
  - Ta có m = $\lceil log_2 p \rceil$ = $\lceil log_2 2147483647 \rceil$ = 31, t = $\lceil 31/8 \rceil$ = 4
  - Biểu diễn a dưới dạng mảng (A[3], A[2], A[1], A[0]):
  - $a = 2^{(t-1)W} A[t-1] + \ldots + 2^{2W} A[2] + 2^{W} A[1] + A[0]$
    $= 2^{(4-1).8} A[4-1] + 2^{2.8} A[2] + 2^{W} A[1] + A[0]$
    $= 2^{24} A[3] + 2^{16} A[2] + 2^{8} A[1] + A[0]$
    $= 2^{24}.1 + 2^{16}.101 + 2^{8}.236 + 21$

Vậy a được biểu diễn qua mảng A: (1, 101, 236, 21)

Phép tính cộng và trừ

- **Bài tập áp dụng:**
  - Cho W= 8, xét $F_{2147483647}$, hãy biểu diễn a dưới dạng mảng
    - a= 765432 (0, 11, 173, 248)
    - a= 123456 (0, 1, 226, 64)

Phép tính cộng và trừ

- Thuật toán cộng và trừ trên trường hữu hạn được đưa ra dưới dạng các thuật toán tương ứng cho các số nguyên w. Phép gán dạng "(ε, z) ← w" được định nghĩa như sau:
  - z ← w mod 2^W và
  - ε ← 0 nếu w ∈ [0, 2^W), ngược lại ε ← 1
  - Nếu w = x + y + ε′ với x, y ∈ [0, 2^W) và ε′ ∈ {0, 1} , thì w = ε 2^W + z và ε được gọi là "bit nhớ" (carry bit) cho phép cộng mỗi một từ đơn (ε = 1 nếu và chỉ nếu z < x + ε′)

Phép tính cộng và trừ

- Thuật toán cộng chính xác bội:

Algorithm 1. Multiprecision addition

Input: số nguyên a, b ∈ [0, 2^{Wt})
Output: (ε, c) với c = a + b mod 2^{Wt} và ε là bit nhớ

1. (ε, C[0]) ← A[0] + B[0]
2. For i from 1 to t – 1 do
   2.1 (ε, C[i]) ← A[i] + B[i] + ε
3. Return (ε, c)

Phép tính cộng và trừ

- Ví dụ minh họa thuật toán cộng chính xác bội:
  - Cho a = (0, 11, 173, 248); b = (0, 1, 226, 64), với w = 8, t = 4.
  - Áp dụng thuật toán 1 tìm c = a + b mod 2^{Wt}

Phép tính cộng và trừ

- (ε, C[0]) ← A[0] + B[0] = 248 + 64 = 312 mod 2^8 = 56 (gán ε = 1)
- i = 1: (ε, C[1]) ← A[1] + B[1] + ε = 173 + 226 + 1 mod 2^8 = 400 mod 2^8 = 144 (gán ε = 1)
- i = 2: (ε, C[2]) ← A[2] + B[2] + ε = 11 + 1 + 1 mod 2^8 = 13 (gán ε = 0)
- i = 3: (ε, C[3]) ← A[3] + B[3] + ε = 0 + 0 + 0 mod 2^8 = 0 (gán ε = 0)
- Return (0, (0, 13, 144, 56))

Phép tính cộng và trừ

- **Bài tập:**
  - Cho W = 8, t = 4. Áp dụng thuật toán 1 tính c = a + b mod 2^{Wt} với a = (57, 169, 36, 27); b = (0, 98, 34, 62)

Phép tính cộng và trừ

- Thuật toán trừ chính xác bội:

Algorithm 2. Multiprecision subtraction

Input: số nguyên a, b ∈ [0, 2^{Wt})
Output: (ε, c) với c = a – b mod 2^{Wt} và ε là bit mượn

1. (ε, C[0]) ← A[0] – B[0]
2. For i from 1 to t – 1 do
   2.1 (ε, C[i]) ← A[i] – B[i] – ε
3. Return (ε, c)

Phép tính cộng và trừ

- Ví dụ minh họa thuật toán trừ chính xác bội:
  - Cho a = (0, 11, 173, 248); b = (0, 1, 226, 64), với w = 8, t = 4.
  - Áp dụng thuật toán 2 tìm c = a - b mod 2^{Wt}

Phép tính cộng và trừ

- (ε, C[0]) ← A[0] - B[0] = 248 - 64 = 184 mod 2^8 (gán ε = 0)
- i = 1: (ε, C[1]) ← A[1] - B[1] − ε = 173 - 226 - 0 mod 2^8 = -53 mod 2^8 = 203 (gán ε = 1)
- i = 2: (ε, C[2]) ← A[2] - B[2] − ε = 11 - 1 - 1 mod 2^8 = 9 (gán ε = 0)
- i = 3: (ε, C[3]) ← A[3] + B[3] − ε = 0 - 0 - 0 mod 2^8 = 0 (gán ε = 0)
- Return (0, (0, 9, 203, 184))

Phép tính cộng và trừ

- **Bài tập:**
  - Cho W = 8, t = 4. Áp dụng thuật toán 2 tính c = a - b mod 2^{Wt} với a = (57, 169, 36, 27); b = (0, 98, 34, 62)

Phép tính cộng và trừ

- Thuật toán cộng trên F_p:

Algorithm 3. Addition in F_p

Input: số modulo p, số nguyên a, b ∈ [0, p − 1]
Output: c = a + b mod p

1. Dùng thuật toán Algorithm 1 để thu được (ε, c) với c = a + b mod 2^{Wt} và ε là bit nhớ.
2. Nếu ε = 1 thì trừ p từ c = (C[t – 1], …, C[2], C[1], C[0]);

   Ngược lại nếu c ≥ p thì c ← c – p
3. Return (c)

Phép tính cộng và trừ

- Ví dụ minh họa thuật toán 3:
  - Cho p = 2.147.483.647, W = 8; ta có m = $\lceil log_2 p \rceil$ = 31; t = $\lceil m/W \rceil = 4$
  - a = (0, 11, 173, 248); b = (0, 1, 226, 64).
  - Áp dụng thuật toán 3 tìm c = a + b mod p
    - Áp dụng thuật toán 1 ta có (ε, c) = (0, (0, 13, 144, 56))
    - Vì c < p => Return (0, 13, 144, 56)

Phép tính cộng và trừ

- **Bài tập:**
  - Cho p = 2.147.483.647, W = 8; ta có m = $\lceil log_2 p \rceil$ = 31; t = $\lceil m/W \rceil = 4$
  - a = (157, 0, 173, 23); b = (169, 1, 0, 64).
  - Áp dụng thuật toán 3 tìm c = a + b mod p

Phép tính cộng và trừ

- a = (157, 0, 173, 23); b = (169, 1, 0, 64)
  - Áp dụng tt 1 tìm (ε, c):
    - (ε, C[0]) ← A[0] + B[0] = 23 + 64 = 87 mod 2^8 = 87 (gán ε = 0)
    - i = 1: (ε, C[1]) ← A[1] + B[1] + ε = 173 + 0 + 0 mod 2^8 = 173 (gán ε = 0)
    - i = 2: (ε, C[2]) ← A[2] + B[2] + ε = 0 + 1 + 0 mod 2^8 = 1 (gán ε = 0)
    - i = 3: (ε, C[3]) ← A[3] + B[3] + ε = 157 + 169 + 0 mod 2^8 = 326 mod 2^8 = 70 (gán ε = 1)
    - Vậy (ε, c) = (1, (70, 1, 173, 87))
  - Ta có ε = 1

Phép tính cộng và trừ

- Ta có p = 2.147.483.647 được biểu diễn dưới dạng mảng (127, 255, 255, 255)
- Áp dụng tt2 tính c – p mod 2^{32} (với c = (70, 1, 173, 87))
  - (ε, C[0]) ← c[0] - p[0] = 87 - 255 = - 168 mod 2^8 = 88 (gán ε = 1)
  - i = 1: (ε, C[1]) ← c[1] - p[1] - ε = 173 - 255 - 1 mod 2^8 = -83 mod 2^8 = 173 (gán ε = 1)
  - i = 2: (ε, C[2]) ← c[2] - p[2] - ε = 1 - 255 - 1 mod 2^8 = -255 mod 2^8 = 1 (gán ε = 1)
  - i = 3: (ε, C[3]) ← c[3] - p[3] - ε = 70 - 127 - 1 mod 2^8 = -58 mod 2^8 = 198 (gán ε = 1)
  - Return (198, 1, 173, 88)

Phép tính cộng và trừ

- Thuật toán trừ trên F_q:

Algorithm 4. Subtractionin F_q

Input: số modulo p, số nguyên a, b ∈ [0, $p - 1$]
Output: c = a – b mod p

1. Dùng thuật toán Algorithm 2 để thu được (ε, c) với c = a – b mod 2^{W_t} và ε là bit mượn.
2. Nếu ε = 1 thì thêm p từ c = (C[t – 1], …, C[2], C[1], C[0]);
3. Return (c)

Phép tính cộng và trừ

- BT áp dụng:
  - Cho p = 2.147.483.647, W = 8; ta có m = $\lceil log_2 p \rceil$ = 31; t = $\lceil m/W \rceil = 4$
  - a = (0, 11, 173, 248); b = (0, 1, 226, 64).
  - Áp dụng thuật toán 4 tìm c = a - b mod p

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Phép tính nhân

- Thuật toán nhân
  - Trong đó UV biểu thị cho 2W bit được nối bởi W bit của từ U với W bit từ V

Algorithm 4. Integer multiprecision (operand scanning form)	
Input: số nguyên a, b ∈ [0, $p - 1$) **Output:** c = a . b	
1. For i from 0 to t – 1 do 1.1 C[i] ← 0	(UV) ← C[i + j] + A[i]. B[j] + U
2. For i from 0 to t – 1 do	C[i + j] ← V
2.1 U ← 0	2.3. C[i + t] ← U
2.2 For j from 0 to t – 1 do	3. Return(c).

Phép tính nhân

- **Ví dụ:**
  - Cho p = 2.147.483.647, W = 8; ta có m = $\lceil log_2 p \rceil$ = 31; t = $\lceil m/W \rceil = 4$
  - a = (0, 11, 173, 248); b = (0, 1, 226, 64).
  - Tính c = a.b

Phép tính nhân

Algorithm 4. Integer multiprecision (product scanning form)

Input: số nguyên a, b ∈ [0, p − 1)
Output: c = a . b

1. $R_0 \leftarrow 0$, $R_1 \leftarrow 0$, $R_2 \leftarrow 0$.
2. For k from 0 to 2t – 2 do
   2.1 For (i, j) ∈ {(i, j) | i + j = k, 0 ≤ i, j ≤ t – 1 } do
      (U, V) ← A[i]. B[j]
      $(\varepsilon, R_0) \leftarrow R_0 + V$
      $(\varepsilon, R_1) \leftarrow R_1 + U + \varepsilon$
      $R_2 \leftarrow R_2 + \varepsilon$
   2.2 $C[k] \leftarrow R_0$, $R_0 \leftarrow R_1$, $R_1 \leftarrow R_2$, $R_2 \leftarrow 0$
3. $C[2t-1] \leftarrow R_0$
4. Return(c).

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_P

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Phép tính bình phương

Algorithm 5. Integer squaring

Input: số nguyên a ∈ [0, p − 1)
Output: c = a^2

1. $R_0 \leftarrow 0$, $R_1 \leftarrow 0$, $R_2 \leftarrow 0$.
2. For k from 0 to 2t – 2 do
   2.1 For (i, j) ∈ {(i, j) | i + j = k, 0 ≤ i, j ≤ t – 1 } do
      (U, V) ← A[i]. A[j]
      If (i < j) then do: $(\varepsilon, UV) \leftarrow R_0 + V$
      $(\varepsilon, R_1) \leftarrow R_1 + U + \varepsilon$
      $R_2 \leftarrow R_2 + \varepsilon$
   2.2 $C[k] \leftarrow R_0$, $R_0 \leftarrow R_1$, $R_1 \leftarrow R_2$, $R_2 \leftarrow 0$
3. $C[2t-1] \leftarrow R_0$
4. Return(c).

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_P

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Phép lấy modulo

Algorithm 6. Barrett reduction

Input: p, b ≥ 3, $k = \lfloor log_b p \rfloor + 1$, 0 ≤ z < b^{2k}, và μ = $\lfloor b^{2k}/p \rfloor$
Output: z mod p

1. $\hat{q} \leftarrow \lfloor z/b^{k-1}.\mu/b^{k+1} \rfloor$
2. $r \leftarrow (z \bmod b^{k-1}) - (\hat{q}.p \bmod b^{k-1})$
3. If r < 0 then $r \leftarrow r + b^{k-1}$
4. While r ≥ p do $r \leftarrow r - p$
5. Return (r)

BÀI 02 - MỤC TIÊU

- Nắm được, cài đặt được các phép tính toán hiệu quả trên số nguyên lớn

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Phép lũy thừa

- Lấy R > p với gcd(R, p) = 1. Tính zR^{-1} mod p với z < pR
- Xét p là số lẻ, $R = 2^{Wt}$. Nếu $p' = -p^{-1}$ mod R thì $c = zR^{-1}$ mod p có thể gồm:
  - $c \leftarrow (z + (zp' mod\ R)p)/R$
  - Nếu c ≥ p thì c □ $c - p$
- Với t(t + 1) phép nhân chính xác đơn
- Cho x ∈[0, p), $\tilde{x} \leftarrow xR\ mod\ p$. Chú ý: $(\tilde{x}\tilde{y})R^{-1} mod\ p = (xy)R mod\ p$
- Ta định nghĩa tích của $\tilde{x}$ và $\tilde{y}$:
  - $\text{Mont}(\tilde{x}, \tilde{y}) = \tilde{x}\tilde{y}R^{-1} mod\ p = xy\ R\ mod\ p$

Phép lũy thừa

Algorithm 7. Montgomery exponentiation (basic)

Input: số nguyên lẻ p, $R = 2^{Wt}, p' = -p^{-1} mod R, x \in [0, p), e = (e_l, e_{l-1} \dots, e_0)_2$
Output: x^e mod p

1. $\tilde{x} \leftarrow xR\ mod\ p$, A ← R mod p
2. For i form l downto 0 do
   - 2.1. A ← Mont(A, A)
   - 2.2. If $e_i = 1$ then A ← Mont(A, $\tilde{x}$)
3. Return (Mont(A, 1))

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Phép tính nghịch đảo

- **Thuật toán Euclide mở rộng:**
  - Nếu gcd(a,b) = d thì phương trình bất định ax + by = d có nghiệm nguyên (x,y) và một nghiệm nguyên (x,y) như vậy có thể được tính bằng thuật toán Euclide mở rộng.
  - Điều cần và đủ để có nghịch đảo là d = 1 và khi đó:
    - x là nghịch đảo của a mod b và y là nghịch đảo của b mod a
  - Ta mở rộng thuật toán Euclide:
    - Tìm ước chung lớn nhất của a và b,
    - Tính nghịch đảo trong trường hợp GCD(a, b) = 1.

Phép tính nghịch đảo

Thuật toán 8: Euclide mở rộng

Input: Hai số nguyên dương a, b (a ≥ b)
Output: d = gcd(a, b) và số nguyên x, y thỏa mãn ax + by = d

1. If b = 0 then d ← a, x ← 1, y ← 0 and Return(d, x, y).
2. $x_2 \leftarrow 1, x_1 \leftarrow 0, y_2 \leftarrow 0, y_1 \leftarrow 1$
3. While b > 0 do
   - 3.1. $q \leftarrow \lfloor a/b \rfloor, r \leftarrow a - qb, \quad x \leftarrow x_2 - qx_1, y \leftarrow y_2 - qy_1$
   - 3.2. $a \leftarrow b, b \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x, y_2 \leftarrow y_1, y_1 \leftarrow y$
4. d ← a, x ← x_2, y ← y_2
5. Return(d, x, y)

Phép tính nghịch đảo

- Áp dụng thuật toán trên với các đầu vào:
  - 1) a = 1759, b = 550
  - 2) a = 3458, b = 4864

Phép tính nghịch đảo

q	r	x	y	a	b	x_2	x_1	y_2	y_1
-	-	-	-	1759	550	1	0	0	1
3	109	1	-3	550	109	0	1	1	-3
5	5	-5	16	109	5	1	-5	-3	16
21	4	106	-339	5	4	-5	106	16	-339
1	1	-111	335	4	1	106	-111	-339	355
4	0	550	-1759	1	0	-111	550	355	-1759

- a = 1759, b = 550

d = 1, x ← -111, y ← 355

Gcd(1759, 550) = 1
(x, y) = (-111, 355)
Hay $550^{-1} \bmod 1759 = 355$
$1759^{-1} \bmod 550 = -111$

Phép tính nghịch đảo

Algorithm 9 Inversion in $\mathbb{F}_p$ using the extended Euclidean algorithm

INPUT: Prime p and $a \in [1, p-1]$.
OUTPUT: $a^{-1} \bmod p$.

1. $u \leftarrow a, v \leftarrow p$.
2. $x_1 \leftarrow 1, x_2 \leftarrow 0$.
3. While $u \neq 1$ do
   3.1 $q \leftarrow \lfloor v/u \rfloor, r \leftarrow v - qu, x \leftarrow x_2 - qx_1$.
   3.2 $v \leftarrow u, u \leftarrow r, x_2 \leftarrow x_1, x_1 \leftarrow x$.
4. Return($x_1 \bmod p$).

Phép tính nghịch đảo

- Áp dụng thuật toán trên với các đầu vào:
  - a = 127, p = 319

Phép tính nghịch đảo

Algorithm 10 Binary gcd algorithm

INPUT: Positive integers a and b.
OUTPUT: $\gcd(a, b)$.

1. $u \leftarrow a, v \leftarrow b, e \leftarrow 1$.
2. While both u and v are even do: $u \leftarrow u/2, v \leftarrow v/2, e \leftarrow 2e$.
3. While $u \neq 0$ do
   3.1 While u is even do: $u \leftarrow u/2$.
   3.2 While v is even do: $v \leftarrow v/2$.
   3.3 If $u \geq v$ then $u \leftarrow u - v$; else $v \leftarrow v - u$.
4. Return($e \cdot v$).

Phép tính nghịch đảo

Algorithm 11 Binary algorithm for inversion in $\mathbb{F}_p$

INPUT: Prime p and $a \in [1, p-1]$.
OUTPUT: $a^{-1} \bmod p$.

1. $u \leftarrow a, v \leftarrow p$.
2. $x_1 \leftarrow 1, x_2 \leftarrow 0$.
3. While ($u \neq 1$ and $v \neq 1$) do
   3.1 While u is even do
   $u \leftarrow u/2$.
   If x_1 is even then $x_1 \leftarrow x_1/2$; else $x_1 \leftarrow (x_1 + p)/2$.
   3.2 While v is even do
   $v \leftarrow v/2$.
   If x_2 is even then $x_2 \leftarrow x_2/2$; else $x_2 \leftarrow (x_2 + p)/2$.
   3.3 If $u \geq v$ then: $u \leftarrow u - v, x_1 \leftarrow x_1 - x_2$;
   Else: $v \leftarrow v - u, x_2 \leftarrow x_2 - x_1$.
4. If $u = 1$ then return($x_1 \bmod p$); else return($x_2 \bmod p$).

Phép tính nghịch đảo

Algorithm 12 Partial Montgomery inversion in $\mathbb{F}_p$

INPUT: Odd integer $p > 2$, $a \in [1, p-1]$, and $n = \lceil \log_2 p \rceil$.
OUTPUT: Either "not invertible" or (x, k) where $n \le k \le 2n$ and $x = a^{-1}2^k \bmod p$.

1. $u \leftarrow a$, $v \leftarrow p$, $x_1 \leftarrow 1$, $x_2 \leftarrow 0$, $k \leftarrow 0$.
2. While $v > 0$ do
   2.1 If v is even then $v \leftarrow v/2$, $x_1 \leftarrow 2x_1$;
   else if u is even then $u \leftarrow u/2$, $x_2 \leftarrow 2x_2$;
   else if $v \ge u$ then $v \leftarrow (v-u)/2$, $x_2 \leftarrow x_2 + x_1$, $x_1 \leftarrow 2x_1$;
   else $u \leftarrow (u-v)/2$, $x_1 \leftarrow x_2 + x_1$, $x_2 \leftarrow 2x_2$.
   2.2 $k \leftarrow k+1$.
3. If $u \ne 1$ then return("not invertible").
4. If $x_1 > p$ then $x_1 \leftarrow x_1 - p$.
5. Return(x_1, k).

Phép tính nghịch đảo

Algorithm13 Montgomery inversion in $\mathbb{F}_p$

INPUT: Odd integer $p > 2$, $n = \lceil \log_2 p \rceil$, $R^2 \bmod p$, and $\widetilde{a} = aR \bmod p$ with $\gcd(a, p) = 1$.
OUTPUT: $a^{-1}R \bmod p$.

1. Use Algorithm 2.23 to find (x, k) where $x = \widetilde{a}^{-1}2^k \bmod p$ and $n \le k \le 2n$.
2. If $k < Wt$ then
   2.1 $x \leftarrow \text{Mont}(x, R^2) = a^{-1}2^k \bmod p$.
   2.2 $k \leftarrow k + Wt$. {Now, $k > Wt$.}
3. $x \leftarrow \text{Mont}(x, R^2) = a^{-1}2^k \bmod p$.
4. $x \leftarrow \text{Mont}(x, 2^{2Wt-k}) = a^{-1}R \bmod p$.
5. Return(x).

Phép tính nghịch đảo

Algorithm14 Simultaneous inversion

INPUT: Prime p and nonzero elements $a_1, \ldots, a_k$ in $\mathbb{F}_p$
OUTPUT: Field elements $a_1^{-1}, \ldots, a_k^{-1}$, where $a_i a_i^{-1} \equiv 1 \pmod p$.

1. $c_1 \leftarrow a_1$.
2. For i from 2 to k do: $c_i \leftarrow c_{i-1}a_i \bmod p$.
3. $u \leftarrow c_k^{-1} \bmod p$.
4. For i from k downto 2 do
   4.1 $a_i^{-1} \leftarrow uc_{i-1} \bmod p$.
   4.2 $u \leftarrow ua_i \bmod p$.
5. $a_1^{-1} \leftarrow u$.
6. Return$(a_1^{-1}, \ldots, a_k^{-1})$.

TÍNH TOÁN TRÊN SỐ NGUYÊN LỚN TRONG TRƯỜNG F_p

- Giới thiệu về các trường hữu hạn
- Phép tính cộng và trừ
- Phép tính nhân
- Phép tính bình phương
- Phép lấy modulo
- Phép lũy thừa
- Phép tính nghịch đảo
- Tính toán với modulo là số nguyên tố đặc biệt của NIST

Tính toán với modulo là số nguyên tố đặc biệt của NIST

Algorithm 15 Fast reduction modulo $p_{192} = 2^{192} - 2^{64} - 1$

INPUT: An integer $c = (c_5, c_4, c_3, c_2, c_1, c_0)$ in base 2^{64} with $0 \le c < p_{192}^2$.
OUTPUT: $c \bmod p_{192}$.

1. Define 192-bit integers:
   $s_1 = (c_2, c_1, c_0)$, $s_2 = (0, c_3, c_3)$,
   $s_3 = (c_4, c_4, 0)$, $s_4 = (c_5, c_5, c_5)$.
2. Return$(s_1 + s_2 + s_3 + s_4 \bmod p_{192})$.

Algorithm 16 Fast reduction modulo $p_{224} = 2^{224} - 2^{96} + 1$

INPUT: An integer $c = (c_{13}, \ldots, c_2, c_1, c_0)$ in base 2^{32} with $0 \le c < p_{224}^2$.
OUTPUT: $c \bmod p_{224}$.

1. Define 224-bit integers:
   $s_1 = (c_6, c_5, c_4, c_3, c_2, c_1, c_0)$, $s_2 = (c_{10}, c_9, c_8, c_7, 0, 0, 0)$,
   $s_3 = (0, c_{13}, c_{12}, c_{11}, 0, 0, 0)$, $s_4 = (c_{13}, c_{12}, c_{11}, c_{10}, c_9, c_8, c_7)$,
   $s_5 = (0, 0, 0, 0, c_{13}, c_{12}, c_{11})$.
2. Return$(s_1 + s_2 + s_3 - s_4 - s_5 \bmod p_{224})$.

Tính toán với modulo là số nguyên tố đặc biệt của NIST

Algorithm 17 Fast reduction modulo $p_{256} = 2^{256} - 2^{224} + 2^{192} + 2^{96} - 1$

INPUT: An integer $c = (c_{15}, \ldots, c_2, c_1, c_0)$ in base 2^{32} with $0 \le c < p_{256}^2$.
OUTPUT: $c \bmod p_{256}$.

1. Define 256-bit integers:
   $s_1 = (c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$,
   $s_2 = (c_{15}, c_{14}, c_{13}, c_{12}, c_{11}, 0, 0, 0)$,
   $s_3 = (0, c_{15}, c_{14}, c_{13}, c_{12}, 0, 0, 0)$,
   $s_4 = (c_{15}, c_{14}, 0, 0, 0, c_{10}, c_9, c_8)$,
   $s_5 = (c_8, c_{13}, c_{15}, c_{14}, c_{13}, c_{11}, c_{10}, c_9)$,
   $s_6 = (c_{10}, c_8, 0, 0, 0, c_{13}, c_{12}, c_{11})$,
   $s_7 = (c_{11}, c_9, 0, 0, c_{15}, c_{14}, c_{13}, c_{12})$,
   $s_8 = (c_{12}, 0, c_{10}, c_9, c_8, c_{15}, c_{14}, c_{13})$,
   $s_9 = (c_{13}, 0, c_{11}, c_{10}, c_9, 0, c_{15}, c_{14})$.
2. Return$(s_1 + 2s_2 + 2s_3 + s_4 + s_5 - s_6 - s_7 - s_8 - s_9 \bmod p_{256})$.

Tính toán với modulo là số nguyên tố đặc biệt của NIST

Algorithm 18 Fast reduction modulo $p_{384} = 2^{384} - 2^{128} - 2^{96} + 2^{32} - 1$

INPUT: An integer $c = (c_{23}, \ldots, c_2, c_1, c_0)$ in base 2^{32} with $0 \le c < p_{384}^2$.
OUTPUT: $c \bmod p_{384}$.

1. Define 384-bit integers:
   $s_1 = (c_{11}, c_{10}, c_9, c_8, c_7, c_6, c_5, c_4, c_3, c_2, c_1, c_0)$,
   $s_2 = (0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, 0, 0, 0, 0)$,
   $s_3 = (c_{23}, c_{22}, c_{21}, c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12})$,
   $s_4 = (c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{23}, c_{22}, c_{21})$,
   $s_5 = (c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{20}, 0, c_{23}, 0)$,
   $s_6 = (0, 0, 0, 0, c_{23}, c_{22}, c_{21}, c_{20}, 0, 0, 0, 0)$,
   $s_7 = (0, 0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, 0, 0, c_{20})$,
   $s_8 = (c_{22}, c_{21}, c_{20}, c_{19}, c_{18}, c_{17}, c_{16}, c_{15}, c_{14}, c_{13}, c_{12}, c_{23})$,
   $s_9 = (0, 0, 0, 0, 0, 0, 0, c_{23}, c_{22}, c_{21}, c_{20}, 0)$,
   $s_{10} = (0, 0, 0, 0, 0, 0, 0, c_{23}, c_{23}, 0, 0, 0)$.
2. Return$(s_1 + 2s_2 + s_3 + s_4 + s_5 + s_6 + s_7 - s_8 - s_9 - s_{10} \bmod p_{384})$.

Tính toán với modulo là số nguyên tố đặc biệt của NIST

Algorithm 19 Fast reduction modulo $p_{521} = 2^{521} - 1$

INPUT: An integer $c = (c_{1041}, \ldots, c_2, c_1, c_0)$ in base 2 with $0 \le c < p_{521}^2$.
OUTPUT: $c \bmod p_{521}$.

1. Define 521-bit integers:
   $s_1 = (c_{1041}, \ldots, c_{523}, c_{522}, c_{521})$,
   $s_2 = (c_{520}, \ldots, c_2, c_1, c_0)$.
2. Return$(s_1 + s_2 \bmod p_{521})$.