

Trường Đại học Khoa học Tự nhiên TP HCM
Khoa Toán - Tin học
Bộ môn Ứng dụng Tin học

Lý thuyết mã hóa thông tin

Tài liệu hướng dẫn thực hành

Version 1.0

Lưu hành nội bộ - 2010

Mục lục

Mục lục	3
1. Ngôn ngữ lập trình Python	5
1.1. Giới thiệu	5
1.2. Biến và kiểu dữ liệu	6
1.3. Phép toán cơ bản	7
1.4. Biểu thức	8
1.5. Cấu trúc điều khiển	8
1.6. Chuỗi ký tự và danh sách	9
1.7. Định nghĩa hàm	14
1.8. Các hàm toán học	15
2. Mật mã cổ điển	16
2.1. Tóm tắt lý thuyết	16
2.2. Bài tập thực hành	18
3. Mã đối xứng hiện đại và mã công khai	20
3.1. Tóm tắt lý thuyết	20
3.2. Bài tập thực hành	21
4. Hàm băm mật mã	23
4.1. Tóm tắt lý thuyết	23
4.2. Bài tập thực hành	24

	3
5. Hệ mã logarit rời rạc	25
5.1. Tóm tắt lý thuyết	25
5.2. Bài tập thực hành	26
6. Chữ ký điện tử	27
6.1. Tóm tắt lý thuyết	27
6.2. Bài tập thực hành	29
Tài liệu tham khảo	30

Chương 1.

Ngôn ngữ lập trình Python

Trong chương này sinh viên sẽ được làm quen với ngôn ngữ lập trình Python để sử dụng khi thực hành các bài tập của môn học Số học và thuật toán. Nội dung của chương sẽ giúp sinh viên có thể thực hiện một chương trình đơn giản thông qua việc giới thiệu các kiến thức cơ bản về Python như các phép toán số học, các kiểu dữ liệu, các cấu trúc điều khiển... Sinh viên cần tìm hiểu thêm các tài liệu khác ([?]) để tra cứu các hàm cũng như các gói hàm phục vụ cho mục đích thực hành của mình.

1.1. Giới thiệu

Python là một ngôn ngữ lập trình cấp cao tương tự như các ngôn ngữ lập trình khác như C, C++, Perl, Java... nhưng là một ngôn ngữ *thông dịch*. Điều này có nghĩa là mỗi câu lệnh sẽ được *trình thông dịch* thực thi một cách tuần tự khi chương trình Python được khởi chạy.

Trình thông dịch của Python có 2 cách sử dụng, sử dụng ở *chế độ tương tác* (interactive mode) và sử dụng ở *chế độ kịch bản* (script mode). Ở chế độ tương tác, câu lệnh được gõ và thực thi, kết quả sẽ được in ra ngay sau đó:

```
>>> 1 + 1
2
```

Ngược lại, ở chế độ kịch bản, các câu lệnh được lưu trong một tập dữ liệu gọi là *kịch bản* (script) và sẽ được thực thi một lần.

```
python myscript.py
```

Trong cả hai chế độ, ký hiệu # báo hiệu một dòng chú thích.

```
>>> #Mac dinh la 60 giay
>>> _time = 60
```

1.2. Biến và kiểu dữ liệu

Trong ngôn ngữ lập trình Python, một *biến* (variable) được khai báo và khởi tạo thông qua câu lệnh gán.

```
>>> message = 'And now for something completely different'
>>> n = 17
>>> pi = 3.1415926535897931
```

Tên biến bao gồm các ký tự chữ, số và ký hiệu underscore (_). Tên biến không được bắt đầu bằng số và không được trùng với các *từ khóa* (keyword) của Python (xem [?], trang 11).

```
>>> 76trombones = 'big parade'
SyntaxError: invalid syntax
>>> more@ = 1000000
SyntaxError: invalid syntax
>>> class = 'Advanced Theoretical Zymurgy'
SyntaxError: invalid syntax
```

Một khi được khai báo, biến sẽ có một *kiểu dữ liệu* được tự động xác định bởi Python. Kiểu dữ liệu của một biến có thể được nhận biết bằng lệnh `type`.

```
>>> type(message)
<type 'str'>
>>> type(n)
<type 'int'>
>>> type(pi)
<type 'float'>
```

Để kiểm tra một biến có một kiểu dữ liệu cụ thể nào đó, Python cung cấp lệnh `isinstance`.

```
>>> isinstance(message, str)
True
>>> isinstance(pi, int)
False
```

1.3. Phép toán cơ bản

Python cung cấp các phép toán số học cơ bản như cộng (+), trừ (-), nhân (*), chia (/), lũy thừa (**), và phép chia modulo (%).

```
>>> 20+32; hour-1; hour*60+minute; minute/60; 5**2; 9*(15-7);
```

Trong Python 2.0, phép chia các số nguyên cho kết quả là một số nguyên, phép chia các số thực sẽ cho kết quả là các số thực.

```
>>> minute = 59
>>> minute/60
0
>>> minute/60.0
0.98333333333333328
```

Ngoài ra còn có các phép toán so sánh như bằng (==), khác (!=), lớn hơn (>), nhỏ hơn (<), lớn hơn hoặc bằng (>=), và nhỏ hơn hoặc bằng (<=).

```
>>> x != y      # x is not equal to y
>>> x > y       # x is greater than y
>>> x < y       # x is less than y
>>> x >= y      # x is greater than or equal to y
>>> x <= y      # x is less than or equal to y
```

Cuối cùng là các phép toán luận lý bao gồm and, or, và not. Một số khác 0 được xem là một giá trị True trong các phép toán luận lý của Python.

```
>>> n = 9
>>> n % 2 == 0 or n % 3 == 0
True
>>> 17 and True
True
```

1.4. Biểu thức

Biểu thức là sự kết hợp giữa các giá trị, các biến và các phép toán. Biểu thức có thể chỉ gồm một giá trị, hoặc một biến, nhưng không thể là một phép toán.

```
>>> 17
>>> x
>>> 17 + x
```

1.5. Cấu trúc điều khiển

Cấu trúc điều kiện có thể đơn giản chỉ gồm một biểu thức `if`, hoặc gồm nhiều cặp `if-else` khác nhau.

```
>>> if x > 0:
...     print('x is positive')
>>> if x == y:
...     print('x and y are equal')
... else:
...     if x < y:
...         print('x is less than y')
...     else:
...         print('x is greater than y')
```

Cấu trúc lặp bao gồm các cấu trúc `for` và `while`.

```
>>> for i in range(4):
...     print('Hello!')
>>> while True:
...     print(x)
...     y = (x + a/x) / 2
...     if abs(y-x) < epsilon:
...         break
...     x = y
```

1.6. Chuỗi ký tự và danh sách

1.6.1. Chuỗi ký tự

Chuỗi ký tự trong Python có thể được xem như là một mảng các ký tự, có chỉ số bắt đầu từ 0, và có giá trị hằng số.

```
>>> fruit = 'banana'
>>> print(fruit[1])
a
>>> fruit[1] = 'i'
TypeError: object does not support item assignment
```

Các thao tác trên chuỗi ký tự: nối chuỗi, tính độ dài, lấy chuỗi con.

```
>>> fruit[:3]
'ban'
>>> fruit[3:]
'ana'
>>> fruit[0:5:2]    # the third index is the "step size"
'bnn'
>>> fruit = fruit + 'mango'
>>> len(fruit)
11
```

Chuyển chuỗi sang chữ hoa bằng thủ tục upper và tìm kiếm trong chuỗi bằng thủ tục find.

```
>>> word = 'banana'
>>> new_word = word.upper()
>>> print(new_word)
BANANA
>>> new_word.find('NA')
2
>>> new_word.find('NA', 3)    # start finding at 3
4
>>> new_word.find('B', 1, 4)  # finding starts at 1 and ends at 4
-1
```

```
>>> 'a' in word
True
>>> 'seed' in word
False
```

Các chuỗi có thể so sánh với nhau, trong đó, ký tự chữ hoa có giá trị nhỏ hơn ký tự chữ thường.

```
>>> if word < 'banana':
...     print 'Your word,' + word + ', comes before banana.'
... elif word > 'banana':
...     print 'Your word,' + word + ', comes after banana.'
... else:
...     print 'All right, bananas.'
```

1.6.2. Danh sách

Danh sách là một dãy các giá trị. Không giống như chuỗi ký tự có các giá trị là ký tự, một danh sách có thể chứa giá trị là bất kỳ kiểu nào.

```
>>> [10, 20, 30, 40]
>>> ['crunchy frog', 'ram bladder', 'lark vomit']
>>> ['spam', 2.0, 5, [10, 20]]
>>> empty = []
```

Không giống như chuỗi ký tự, các giá trị bên trong một danh sách có thể được cập nhật và thay đổi tùy ý.

```
>>> numbers = [17, 123]
>>> numbers[1] = 5
>>> print(numbers)
[17, 5]
>>> 5 in numbers
True
```

Để duyệt qua các phần tử của danh sách, chúng ta có thể sử dụng cấu trúc lặp `for`.

```
>>> for x in numbers:
```

```

...     print(x);
>>> for i in range(len(numbers)):
...     numbers[i] = numbers[i] * 2

```

Các phép toán trên danh sách bao gồm kết nối (+) và lặp (*).

```

>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> print c
[1, 2, 3, 4, 5, 6]
>>> [0] * 4
[0, 0, 0, 0]
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]

```

Truy xuất các thành phần của danh sách thông qua các chỉ số:

```

>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3]
['b', 'c']
>>> t[:4]
['a', 'b', 'c', 'd']
>>> t[3:]
['d', 'e', 'f']
>>> t[:]
['a', 'b', 'c', 'd', 'e', 'f']
>>> t[1:3] = ['x', 'y']
>>> print(t)
['a', 'x', 'y', 'd', 'e', 'f']

```

Các phương thức trên danh sách được cài đặt sẵn trong Python bao gồm phép thêm (append), sắp xếp (sort), xóa phần tử (pop, del, remove).

```

>>> # append a value or a list to current list
>>> t = ['a', 'b', 'c']
>>> t.append('d')
>>> print(t)

```

```

['a', 'b', 'c', 'd']
>>> t1 = ['a', 'b', 'c']
>>> t2 = ['d', 'e']
>>> t1.extend(t2)
>>> print(t1)
['a', 'b', 'c', 'd', 'e']
>>> # sort a list
>>> t = ['d', 'c', 'e', 'b', 'a']
>>> t.sort()
>>> print(t)
['a', 'b', 'c', 'd', 'e']
>>> # delete some items from a list
>>> t = ['a', 'b', 'c']
>>> x = t.pop(1)
>>> print(t)
['a', 'c']
>>> print(x)
b
>>> t = ['a', 'b', 'c']
>>> t.remove('b')
>>> print(t)
['a', 'c']
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del t[1:5]
>>> print(t)
['a', 'f']

```

1.6.3. Liên hệ giữa chuỗi và danh sách

Các phép toán chuyển đổi giữa chuỗi và danh sách:

Chuyển một chuỗi ký tự thành một danh sách

```

>>> s = 'spam'
>>> t = list(s)
>>> print(t)
['s', 'p', 'a', 'm']

```

Chuyển một câu thành danh sách các từ

```
>>> s = 'pining for the fjords'
>>> t = s.split()
>>> print(t)
['pining', 'for', 'the', 'fjords']
>>> s = 'spam-spam-spam'
>>> delimiter = '-'
>>> s.split(delimiter)
['spam', 'spam', 'spam']
```

Kết hợp các thành phần của danh sách thành một chuỗi ký tự

```
>>> t = ['pining', 'for', 'the', 'fjords']
>>> delimiter = ' '
>>> delimiter.join(t)
'pining for the fjords'
```

1.6.4. Sự khác nhau giữa chuỗi và danh sách

Do các thành phần của một chuỗi không thể bị thay đổi, nên các chuỗi giống nhau sẽ được tham chiếu đến cùng một object.

```
>>> a = 'banana'
>>> b = 'banana'
>>> a is b
True
```

Ngược lại, các thành phần của danh sách có thể được thay đổi bất kỳ lúc nào, nên một danh sách sẽ tham chiếu đến một object riêng.

```
>>> a = [1, 2, 3]
>>> b = [1, 2, 3]
>>> a is b
False
```

Tuy nhiên nếu gán một danh sách cho một biến khác, thì chúng sẽ cùng tham chiếu đến một object.

```

>>> a = [1, 2, 3]
>>> b = a
>>> b is a
True
>>> b[0] = 17
>>> print(a)
[17, 2, 3]

```

1.7. Định nghĩa hàm

Để định nghĩa một hàm, ta dùng từ khóa `def`, dùng từ khóa `return` để xác định giá trị trả về của hàm.

```

>>> def absolute_value(x):
...     if x < 0:
...         return -x
...     else:
...         return x
>>> def distance(x1, y1, x2, y2):
...     dx = x2 - x1
...     dy = y2 - y1
...     dsquared = dx**2 + dy**2
...     result = math.sqrt(dsquared)
...     return result

```

Hàm trong Python cũng cho phép đệ quy như các ngôn ngữ lập trình khác.

```

>>> def factorial(n):
...     if n == 0:
...         return 1
...     else:
...         recurse = factorial(n-1)
...         result = n * recurse
...         return result

```

Chúng ta có thể kiểm tra các tham số đầu vào của một hàm là hợp lệ trước khi thực hiện các tính toán.

```

>>> def factorial (n):
...     if not isinstance(n, int):
...         print 'Factorial is only defined for integers.'
...         return None
...     elif n < 0:
...         print 'Factorial is only defined for positive integers.'
...         return None
...     elif n == 0:
...         return 1
...     else:
...         return n * factorial(n-1)

```

1.8. Các hàm toán học

Python cung cấp một số hàm toán học cơ bản cho người sử dụng, như lũy thừa, lấy căn, logarit, ... Chúng ta có thể sử dụng để kiểm tra tính chính xác của các kết quả tính toán của mình.

Trước tiên chúng ta cần khai báo thư viện toán học

```
>>> import math
```

Sau đó ta có thể thực hiện các phép toán

```

>>> math.sqrt(2) / 2.0
0.707106781187

```

```

>>> radians = 0.7
>>> height = math.sin(radians)

```

```

>>> ratio = signal_power / noise_power
>>> decibels = 10 * math.log10(ratio)

```

```

>>> x = math.sin(degrees / 360.0 * 2 * math.pi)
>>> x = math.exp(math.log(x+1))

```

```

>>> e = math.exp(1.0)
>>> height = radius * math.sin(radians)

```

Chương 2.

Mật mã cổ điển

2.1. Tóm tắt lý thuyết

2.1.1. Mã dịch chuyển - Shift Cipher

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Định nghĩa 1. Đặt $\mathcal{P} = \mathcal{C} = \mathcal{K} = \mathbb{Z}_{26}$. Với mỗi $0 \leq K \leq 25$ và với $x, y \in \mathbb{Z}_{26}$, mã dịch chuyển được định nghĩa như sau:

$$\mathcal{E}_K(x) = (x + K) \bmod 26$$

$$\mathcal{D}_K(y) = (y - K) \bmod 26$$

2.1.2. Mã thay thế - Substitution Cipher

Định nghĩa 2. Cho π là một song ánh từ tập các ký tự thường vào tập các ký tự hoa:

$$\begin{aligned} \pi : \mathbf{a} &\longrightarrow \mathcal{A} \\ a &\longmapsto A \end{aligned}$$

Khi đó với $x \in \mathbf{a}$ và $y \in \mathcal{A}$, mã thay thế được định nghĩa như sau:

$$\mathcal{E}_\pi(x) = \pi(x), \quad \mathcal{D}_\pi(y) = \pi^{-1}(y)$$

Ví dụ 1. Khóa bí mật π của một mã thay thế cho bởi bảng sau:

π	A	B	C	D	E	F	G	H	I	J	K	L	M
	d	l	r	y	v	o	h	e	z	x	w	p	t
	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
	b	g	f	j	q	n	m	u	s	k	a	c	i

π^{-1}	a	b	c	d	e	f	g	h	i	j	k	l	m
	X	N	Y	A	H	P	O	G	Z	Q	W	B	T
	n	o	p	q	r	s	t	u	v	w	x	y	z
	S	F	L	R	C	V	M	U	E	K	J	D	I

2.1.3. Mã Affine

Định nghĩa 3. Đặt $\mathcal{P} = \mathcal{C} = \mathbb{Z}_{26}$ và

$$\mathcal{K} = \{(a, b) \in \mathbb{Z}_{26}^2 : \gcd(a, 26) = 1\}$$

Với mỗi khóa $K = (a, b) \in \mathcal{K}$ và với $x, y \in \mathbb{Z}_{26}$, mã Affine được định nghĩa như sau:

$$\mathcal{E}_K(x) = (ax + b) \bmod 26$$

$$\mathcal{D}_K(y) = a^{-1}(y - b) \bmod 26$$

2.1.4. Mã Vigenère

Định nghĩa 4. Chọn m là một số nguyên dương. Đặt $\mathcal{P} = \mathcal{C} = \mathcal{K} = (\mathbb{Z}_{26})^m$. Với mỗi khóa $K = (k_1, k_2, \dots, k_m) \in \mathcal{K}$ và với $x = (x_1, x_2, \dots, x_m) \in \mathcal{P}$, $y = (y_1, y_2, \dots, y_m) \in \mathcal{C}$, mã Vigenère được định nghĩa như sau:

$$\mathcal{E}_K(x) = (x_1 + k_1, x_2 + k_2, \dots, x_m + k_m) \bmod 26$$

$$\mathcal{D}_K(y) = (y_1 - k_1, y_2 - k_2, \dots, y_m - k_m) \bmod 26$$

2.1.5. Mã Hill

Định nghĩa 5. Chọn $m \geq 2$. Đặt $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ và

$$\mathcal{K} = \{M_{m \times m} \in \mathcal{M}_m(\mathbb{Z}_{26}) : \det(M) \neq 0\}$$

Khi đó với mỗi khóa $K \in \mathcal{K}$ và $x \in \mathcal{P}$, $y \in \mathcal{C}$, mã Hill được định nghĩa như sau:

$$\mathcal{E}_K(x) = xK \pmod{26}$$

$$\mathcal{D}_K(y) = yK^{-1} \pmod{26}$$

2.1.6. Mã chuyển vị - Permutation Cipher

Định nghĩa 6. Chọn m là một số nguyên dương. Đặt $\mathcal{P} = \mathcal{C} = (\mathbb{Z}_{26})^m$ và \mathcal{K} là tập các hoán vị m phần tử $\{1, \dots, m\}$. Khi đó với mỗi khóa K là một hoán vị m phần tử và $x \in \mathcal{P}$, $y \in \mathcal{C}$, mã hoán vị được định nghĩa như sau:

$$\mathcal{E}_K(x) = \mathcal{E}_K(x_1, \dots, x_m) = (x_{K(1)}, \dots, x_{K(m)})$$

$$\mathcal{D}_K(y) = \mathcal{D}_K(y_1, \dots, y_m) = (y_{K^{-1}(1)}, \dots, y_{K^{-1}(m)})$$

Ví dụ 2. Với $m = 6$, ta có một khóa là hoán vị sau:

x	1	2	3	4	5	6
$\pi(x)$	3	5	1	6	4	2
$\pi^{-1}(x)$	3	6	1	5	2	4

2.2. Bài tập thực hành

Bài tập 1. Viết hàm tính mã $\mathcal{E}_{sh(k)}$ và giải mã $\mathcal{D}_{sh(k)}$ dịch chuyển dựa trên khóa k cho trước là một số nguyên không âm không quá 25.

Bài tập 2. Một người tìm cách nâng cao tính an toàn cho văn bản mã hóa của mình. Anh ta nghĩ ra một khóa là một số nguyên dương có n chữ số $k = \overline{k_1 k_2 \dots k_n}$ với k_i là các chữ số thập phân, sau đó anh ta mã hóa văn bản p của mình như sau: $E_k(p) = \mathcal{E}_{sh(k_1)}(\mathcal{E}_{sh(k_2)}(\dots(\mathcal{E}_{sh(k_n)}(p)\dots)))$, biết $\mathcal{E}_{sh(k_i)}(x)$ là hàm mã hóa dịch chuyển có khóa k_i .

- Viết hàm tính mã và giải mã cho phương pháp mã hóa này.
- Nhận xét về tính an toàn của phương pháp này so với mã dịch chuyển gốc.

3. Những điều kiện nào của k làm cho $E_k(p) \neq p$?

Bài tập 3. Viết hàm tính mã $\mathcal{E}_{s(\pi)}$ và giải mã $\mathcal{D}_{s(\pi)}$ thay thế dựa trên khóa π trong ví dụ 1 ở trên.

Bài tập 4. Một người sử dụng hàm $\mathcal{E}_{s(\pi)}$ để mã hóa văn bản p của mình như sau:
 $E_\pi = \mathcal{E}_{s(\pi)}(\mathcal{E}_{s(\pi)}(p))$.

1. Hãy nhận xét về kết quả mã hóa c nhận được.

2. Thử cho một khóa ψ khác để E_ψ không gặp phải vấn đề như khóa π .

Bài tập 5. Viết hàm tính mã $\mathcal{E}_{a(K)}$ và giải mã $\mathcal{D}_{a(K)}$ Affine dựa trên khóa $K \in \mathbb{Z}_{26}^2$.

Bài tập 6. Một người mã hóa văn bản p của mình với khóa $K = (k_1, k_2) \in \mathbb{Z}_{26}^2$ theo phương pháp sau: $E_K(p) = \mathcal{E}_{a(K)}(\mathcal{E}_{sh(k_1)}(\mathcal{E}_{sh(k_2)}(p)))$. Hãy viết hàm tính mã và giải mã cho phương pháp này.

Bài tập 7. Một người mã hóa văn bản p của mình với khóa $K = (k_1, k_2) \in \mathbb{Z}_{26}^2$ theo phương pháp sau: $K' = (k_1, 2\mathcal{E}_{sh(k_2)}(p))$ và $E_K(p) = \mathcal{E}_{a(K')}(p)$. Hãy viết hàm tính mã và giải mã cho phương pháp này.

Bài tập 8. Viết hàm tính mã $\mathcal{E}_{v(K)}$ và giải mã $\mathcal{D}_{v(K)}$ Vigenère dựa trên khóa $K \in \mathbb{Z}_{26}^m$.

Bài tập 9. Một người mã hóa văn bản p chiều dài m của mình với khóa $K \in \mathbb{Z}_{26}^m$ theo phương pháp sau: $p' = (\mathcal{E}_{a((11, k_1))}(p_1), \mathcal{E}_{a((11, k_2))}(p_2), \dots, \mathcal{E}_{a((3, k_m))}(p_m))$ và $E_K(p) = \mathcal{E}_{v(K)}(p')$. Hãy viết hàm tính mã và giải mã cho phương pháp này.

Bài tập 10. Viết hàm tính mã $\mathcal{E}_{h(K)}$ và giải mã $\mathcal{D}_{h(K)}$ Hill dựa trên khóa $K \in \mathcal{M}_m(\mathbb{Z}_{26})$ và K khả nghịch trên \mathbb{Z}_{26} .

Bài tập 11. Một người mã hóa văn bản p chiều dài m của mình với khóa $K \in \mathcal{M}_m(\mathbb{Z}_{26})$ (K khả nghịch trên \mathbb{Z}_{26}) theo phương pháp sau: $E_K(p) = \mathcal{E}_{h(K)}(\mathcal{E}_{h(K^T)}(p))$. Hãy viết hàm tính mã và giải mã cho phương pháp này.

Bài tập 12. Viết hàm tính mã $\mathcal{E}_{p(\pi)}$ và giải mã $\mathcal{D}_{p(\pi)}$ chuyển vị dựa trên khóa π trong ví dụ 2 ở trên.

Chương 3.

Mã đối xứng hiện đại và mã công khai

3.1. Tóm tắt lý thuyết

3.1.1. Mã lặp

Một hệ mã lặp tiêu biểu bao gồm hàm lặp song ánh g và tập r khóa K_1, K_2, \dots, K_r , khi đó việc mã hóa một văn bản p được thực hiện qua r vòng lặp:

$$\begin{aligned}c &= w_r \\w_i &= g(w_{i-1}, K_i), 1 \leq i \leq r \\w_0 &= p\end{aligned}$$

Việc giải mã c cũng được thực hiện qua r bước lặp:

$$\begin{aligned}p &= w_0 \\w_{i-1} &= g^{-1}(w_i, K_i), 1 \leq i \leq r \\w_r &= c\end{aligned}$$

3.1.2. Mã DES

Mã DES là một mã lặp 16 vòng mã hóa một văn bản p dài 64 bit với một khóa dài 56 bit. Ở mỗi vòng, trạng thái w_i được chia làm hai phần dài 32 bit là L_i và R_i . Khóa K_i ở mỗi lần lặp dài 48 bit là một khóa được sinh ra từ khóa K ban đầu. Việc mã hóa được tiến hành như sau:

$$\begin{aligned}
c &= IP^{-1}(R_{16}, L_{16}) \\
w_i &= (L_i, R_i) = g(L_{i-1}, R_{i-1}, K_i) = (R_{i-1}, L_{i-1} \oplus f(R_{i-1}, K_i)) \\
w_0 &= (L_0, R_0) = IP(x)
\end{aligned}$$

Việc giải mã được thực hiện tương tự với hàm g^{-1} cho bởi công thức $(L_{i-1}, R_{i-1}) = g^{-1}(L_i, R_i, K_i) = (R_i \oplus f(L_i, K_i), L_i)$.

3.1.3. Mã AES

Mã AES là một mã lặp mã hóa một văn bản p dài 128 bit với khóa K có chiều dài 128, 192, hoặc 256 bit. Số vòng lặp của AES phụ thuộc vào chiều dài của khóa K , tương ứng là 10, 12, hoặc 14 vòng.

3.1.4. Hệ mã RSA

Chọn $n = pq$ với p, q là các số nguyên tố. Đặt $\mathcal{P} = \mathcal{C} = \mathbb{Z}_n$ và

$$\mathcal{K} = \{(n, p, q, a, b) : a \equiv b \pmod{n}\}$$

Với $K = (n, p, q, a, b) \in \mathcal{K}$ và $x, y \in \mathbb{Z}_n$, hàm mã hóa và giải mã RSA được định nghĩa như sau:

$$\mathcal{E}_K(x) = x^b \pmod{n}$$

$$\mathcal{D}_K(y) = y^a \pmod{n}$$

Khi đó (n, b) được gọi là khóa công khai, (p, q, a) được gọi là khóa bí mật.

3.2. Bài tập thực hành

Bài tập 13. *Hãy đề xuất và cài đặt thuật toán cho hàm song ánh một chuỗi ký tự có chiều dài bất kỳ và một chuỗi bit.*

Bài tập 14. *Hãy đề xuất và cài đặt thuật toán cho hàm song ánh một chuỗi bit có chiều dài bất kỳ thành một chuỗi bit (ngắn nhất) có chiều dài là bội số của n cho trước. (Padding)*

Bài tập 15. *Hãy cài đặt thuật toán DES để mã hóa và giải mã một chuỗi bit dài 64 bit.*

Bài tập 16. *Hãy cài đặt thuật toán DES để mã hóa và giải mã một chuỗi bit dài bất kỳ.*

Bài tập 17. *Hãy cài đặt thuật toán DES để mã hóa và giải mã một chuỗi ký tự có chiều dài bất kỳ.*

Bài tập 18. *Hãy cài đặt thuật toán AES để mã hóa và giải mã một chuỗi bit dài 64 bit.*

Bài tập 19. *Hãy cài đặt thuật toán AES để mã hóa và giải mã một chuỗi ký tự dài bất kỳ.*

Bài tập 20. *Hãy cài đặt thuật toán AES để mã hóa và giải mã một chuỗi ký tự có chiều dài bất kỳ.*

Bài tập 21. *Chọn hai số nguyên tố p, q và $n = pq$. Viết hàm tính khóa bí mật a từ khóa công khai b cho trước.*

Bài tập 22. *Viết hàm mã hóa RSA một văn bản m với khóa công khai là (n, b) cho trước.*

Bài tập 23. *Viết hàm giải mã RSA một đoạn mã c với khóa bí mật là (p, q, a) cho trước.*

Bài tập 24. *Viết hàm thám mã RSA một đoạn mã c với khóa công khai là (n, b) cho trước.*

Chương 4.

Hàm băm mật mã

4.1. Tóm tắt lý thuyết

Hàm băm hiểu theo nghĩa đơn giản là hàm cho tương ứng một mảng dữ liệu lớn với một mảng dữ liệu nhỏ hơn. Các hàm băm nhận một chuỗi bit m , gọi là một *thông điệp* (*message*), có chiều dài tùy ý (hữu hạn) làm dữ liệu đầu vào và tạo ra một chuỗi bit d có chiều dài cố định $n > 0$ gọi là *mã băm* (*hash code* hay *message digest*).

Nếu ký hiệu D là miền xác định (đầu vào) và R là miền giá trị (đầu ra) của hàm băm $h(m)$, ta nhận thấy số lượng phần tử của D thường lớn hơn rất nhiều so với số lượng phần tử trong R (vì chiều dài thông điệp đầu vào là tùy ý và thường lớn hơn n). Do đó $h(m)$ không có tính chất đơn ánh, nghĩa là luôn tồn tại những thông điệp đầu vào có cùng một mã băm, các thông điệp này gọi là các *xung đột* (*collision*). Tuy nhiên nếu với n đủ lớn thì xác suất để tìm ra một xung đột là rất khó và mất rất nhiều thời gian.

Định nghĩa 7. Hàm băm h là một hàm thỏa hai tính chất sau:

- *Tính nén (Compression):* h cho tương ứng một thông điệp đầu vào m có độ dài bất kỳ thành một mã băm $d = h(m)$ có chiều dài cố định n .
- *Dễ tính toán (Ease of computation):* Với mọi thông điệp đầu vào m có chiều dài hữu hạn tùy ý, mã băm $h(m)$ có thể được tính toán một cách dễ dàng.

Định nghĩa 8. Hàm băm mật mã h là hàm băm có các tính chất sau:

- *Tính kháng tiền ảnh (pre-image resistance):* với mọi mã băm cho trước, khó có thể tính toán để tìm được một thông điệp tương ứng với mã băm đó. Nghĩa là, với mọi d cho trước, khó có thể tìm được m sao cho $h(m) = d$.

- *Tính kháng tiền ảnh thứ hai (second pre-image resistance):* với mọi thông điệp cho trước, khó có thể tính toán để tìm được một thông điệp khác có cùng mã băm với thông điệp ban đầu. Nghĩa là, với mọi thông điệp m cho trước, khó có thể tìm được thông điệp $m' \neq m$ sao cho $h(m) = h(m')$.
- *Tính kháng xung đột (collision resistance):* khó có thể tính toán để tìm được hai thông điệp khác nhau mà có cùng mã băm. Nghĩa là, khó có thể tìm được $m \neq m'$ sao cho $h(m) = h(m')$.

Chú ý. Thuật ngữ "khó có thể tính toán" ở đây có nghĩa độ phức tạp của việc tính toán là trên đa thức, hoặc đòi hỏi tài nguyên vượt quá khả năng cung cấp. Cụ thể ta có định nghĩa sau:

4.2. Bài tập thực hành

Bài tập 25. Cài đặt thuật toán Merkle Damgård để ánh xạ một chuỗi bit có chiều dài bất kỳ (nhưng nhỏ hơn 2^{128} , 2^{256} , 2^{512}) thành một chuỗi bit có chiều dài là bội số của $n = 128, 256, 512$.

Bài tập 26. Hãy đề xuất và cài đặt thuật toán ánh xạ một chuỗi bit có chiều dài bất kỳ thành một chuỗi bit có chiều dài là bội số của 32.

Bài tập 27. Cài đặt thuật toán padding cho hàm băm SHA-1.

Bài tập 28. Cài đặt thuật toán SHA-1 để băm một chuỗi ký tự có chiều dài bất kỳ thành một chuỗi bit có chiều dài 160 bit.

Bài tập 29. Cài đặt thuật toán MD5 để băm một chuỗi ký tự có chiều dài bất kỳ thành một chuỗi bit có chiều dài 128 bit.

Chương 5.

Hệ mã logarit rời rạc

5.1. Tóm tắt lý thuyết

5.1.1. Bài toán logarit rời rạc

Cho nhóm nhân (G, \cdot) , một phần tử $\alpha \in G$ có bậc n và phần tử $\beta \in G$. Tìm số nguyên duy nhất a , $0 \leq a \leq n - 1$, sao cho

$$\alpha^a = \beta$$

5.1.2. Hệ mã ElGamal

Chọn p là một số nguyên tố sao cho bài toán logarit rời rạc trên \mathbb{Z}_p^* không thể giải dễ dàng, và đặt $\alpha \in \mathbb{Z}_p^*$ là một phần tử nguyên thủy (phần tử sinh). Đặt $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{C} = \mathbb{Z}_p^* \times \mathbb{Z}_p^*$, và

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

Khi đó bộ (p, α, β) được công khai và số a được giữ bí mật.

Với $K = (p, \alpha, a, \beta)$, và một số ngẫu nhiên bí mật $k \in \mathbb{Z}_{p-1}$, việc mã hóa và giải mã theo hệ ElGamal được thực hiện như sau:

$$c = \mathcal{E}_K(p, k) = (y_1, y_2) \in \mathbb{Z}_p^* \times \mathbb{Z}_p^*$$

trong đó

$$y_1 = \alpha^k \pmod{p}$$

$$y_2 = c\beta^k \pmod{p}$$

$$p = \mathcal{D}_K(y_1, y_2) = y_2(y_1^a)^{-1} \bmod p$$

5.2. Bài tập thực hành

Bài tập 30. Cài đặt thuật toán Pohlig-Hellman để giải bài toán logarit rời rạc.

Bài tập 31. Cài đặt thuật toán Shanks để giải bài toán logarit rời rạc.

Bài tập 32. Cài đặt thuật toán Pollard Rho để giải bài toán logarit rời rạc.

Bài tập 33. Cài đặt thuật toán tìm một phần tử nguyên thủy (phần tử sinh) của \mathbb{Z}_p^* .

Bài tập 34. Cài đặt thuật toán ElGamal để mã hóa và giải mã một chuỗi ký tự có chiều dài bất kỳ.

Chương 6.

Chữ ký điện tử

6.1. Tóm tắt lý thuyết

6.1.1. Chữ ký điện tử sử dụng hệ mã RSA

Chọn $n = pq$ với p, q là các số nguyên tố. Đặt $\mathcal{P} = \mathcal{A} = \mathbb{Z}_n$ và

$$\mathcal{K} = \{(n, p, q, a, b) : a \equiv b \pmod{n}\}$$

(n, b) được công khai, (p, q, a) được giữ bí mật. Khi đó với $K = (n, p, q, a, b) \in \mathcal{K}$ và $x, y \in \mathbb{Z}_n$, việc ký và kiểm tra chữ ký điện tử được thực hiện như sau:

$$\mathbf{sig}_K(x) = x^a \pmod{n}$$

$$\mathbf{ver}_K(y) = \mathbf{true} \iff x \equiv y^b \pmod{n}$$

6.1.2. Chữ ký ElGamal

Chọn p là một số nguyên tố sao cho bài toán logarit rời rạc trên \mathbb{Z}_p không thể giải dễ dàng (triệt tiêu, suy biến). Đặt $\alpha \in \mathbb{Z}_p^*$ là một phần tử nguyên thủy (phần tử sinh), và $\mathcal{P} = \mathbb{Z}_p^*$, $\mathcal{A} = \mathbb{Z}_p^* \times \mathbb{Z}_{p-1}$. Ta định nghĩa

$$\mathcal{K} = \{(p, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

Bộ (p, α, β) được công khai, số a được giữ bí mật. Khi đó với một khóa $K = (p, \alpha, a, \beta)$ và một số ngẫu nhiên (bí mật) $k \in \mathbb{Z}_{p-1}^*$, việc ký và kiểm tra chữ ký điện tử ElGamal được thực hiện như sau:

- Chữ ký điện tử

$$\mathbf{sig}_K(x, k) = (\gamma, \delta)$$

trong đó

$$\gamma = \alpha^k \bmod p$$

$$\delta = (x - a\gamma)k^{-1} \bmod (p - 1)$$

- Kiểm tra chữ ký: với $x, \gamma \in \mathbb{Z}_p^*$ và $\delta \in \mathbb{Z}^{p-1}$,

$$\mathbf{ver}_K(x, (\gamma, \delta)) = \mathbf{true} \iff \beta^\gamma \gamma^{\delta} \equiv \alpha^x \pmod{p}$$

6.1.3. Chữ ký điện tử dựa trên hàm băm SHA-1

Chọn p là một số nguyên tố L -bit sao cho bài toán logarit rời rạc trên \mathbb{Z}_p không thể giải dễ dàng (triệt tiêu, suy biến) và $L \equiv 0 \pmod{64}$ và $512 \leq L \leq 1024$, và chọn q là một số nguyên tố 160-bit là ước số của $p - 1$. Đặt $\alpha \in \mathbb{Z}_p^*$ là một căn bậc q của đơn vị theo modulo p , và đặt $\mathcal{P} = \{0, 1\}^*$, $\mathcal{A} = \mathbb{Z}_q^* \times \mathbb{Z}_q^*$. Ta định nghĩa

$$\mathcal{K} = \{(p, q, \alpha, a, \beta) : \beta \equiv \alpha^a \pmod{p}\}$$

trong đó $0 \leq a \leq q - 1$. Bộ (p, q, α, β) được công khai và a được giữ bí mật.

Với $K = (p, q, \alpha, a, \beta)$ và một số ngẫu nhiên (bí mật) k sao cho $1 \leq k \leq q - 1$, việc ký và kiểm tra chữ ký điện tử dựa trên hàm băm SHA-1 được thực hiện như sau:

- Chữ ký điện tử

$$\mathbf{sig}_K(x, k) = (\gamma, \delta)$$

trong đó

$$\gamma = (a^k \bmod p) \bmod q$$

$$\delta = (\mathbf{SHA} - \mathbf{1}(x) + a\gamma)k^{-1} \bmod q$$

γ, δ không đồng thời bằng 0.

- Kiểm tra chữ ký: với $x \in \{0, 1\}^*$ và $\gamma, \delta \in \mathbb{Z}_q^*$,

$$\mathbf{ver}_K(x, (\gamma, \delta)) = \mathbf{true} \iff (\alpha^{e_1} \beta^{e_2} \bmod p) \bmod q = \gamma$$

trong đó

$$e_1 = \mathbf{SHA} - \mathbf{1}(x)\delta^{-1} \bmod q$$

$$e_2 = \gamma\delta^{-1} \bmod q$$

6.2. Bài tập thực hành

Bài tập 35. Cài đặt chương trình tính chữ ký điện tử và kiểm tra chữ ký điện tử dựa trên thuật toán RSA.

Bài tập 36. Cài đặt chương trình tính chữ ký điện tử và kiểm tra chữ ký điện tử dựa trên ElGamal.

Bài tập 37. Cài đặt chương trình tính chữ ký điện tử và kiểm tra chữ ký điện tử dựa trên hàm băm SHA-1.

Bài tập 38. Cài đặt chương trình tính chữ ký điện tử và kiểm tra chữ ký điện tử dựa trên hàm băm MD5.

Tài liệu tham khảo

- [1] Douglas R. Stinson, *Cryptography: Theory and Practice*, 3rd. ed., Chapman & Hall/CRC, 2006.
- [2] A. Menezes, P. van Oorschot, S. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.
- [3] Phạm Huy Điển, Hà Huy Khoái, *Mã hoá thông tin: Cơ sở toán học và ứng dụng*, NXB ĐHQG Hà Nội, 2003.