ĐỒ ÁN 02 – CÁC HỆ MÃ DỰA TRÊN BÀI TOÁN LOGARITHM RỜI RẠC

October 30, 2023

1 (1 điểm) Căn nguyên thủy modulo p

1.1 Định nghĩa

Cho $p \in \mathcal{P}$ là một số nguyên tố lẻ. Khi đó, luôn tồn tại một số nguyên dương $g \in \mathbb{Z}^+$ thỏa tính chất sau:

$$\forall a \in \{1, \ldots, p-1\}, \exists! k \in \{1, \ldots, p-1\}, a = g^k \pmod{p}$$

Số nguyên g như vậy được gọi là một căn nguyên thủy modulo p.
Ghi chú:

- Ký hiệu $\forall$ có nghĩa là "với mọi".
- Ký hiệu $\exists!$ có nghĩa là "tồn tại duy nhất".

1.2 Cách xác định căn nguyên thủy modulo p

Có thể có nhiều cách khác nhau để kiểm tra một số có phải là căn nguyên thủy hay không, dưới đây là một cách như thế:
Cho $p \in \mathcal{P}$ là một số nguyên tố lẻ. Đặt $\mathcal{U}(p)$ là tập hợp các ước nguyên tố dương của $p-1$. Số nguyên dương $g \in \mathbb{Z}^+$ là căn nguyên thủy modulo p khi và chỉ khi g thỏa tính chất sau:

$$g^{\frac{p-1}{k}} \neq 1 \pmod{p}, \forall k \in \mathcal{U}(p)$$

1.3 Yêu cầu

Trong bài này, sinh viên viết chương trình dùng để kiểm tra một số nguyên có phải là căn nguyên thủy hay không. Chương trình thỏa các yêu cầu sau:

- Ngôn ngữ lập trình là `C++`.
- Toàn bộ mã nguồn được lưu thành một file duy nhất là `main.cpp`.
- Ngoại trừ các tính năng từ ngôn ngữ lập trình `C++` (phiên bản `C++17` trở xuống) và thư viện chuẩn của `C++` (`C++` Standard Library) là được cho phép sử dụng, toàn bộ các thư viện khác bị cấm.
- Sau khi biên dịch mã nguồn thành file `main`, chương trình được chạy như sau.

```
$ .\main test.inp test.out
```

Trong đó:

- `test.inp` là file text chứa dữ liệu đầu vào của chương trình, gồm có 4 dòng:
  - Dòng 01 chứa số nguyên tố lẻ p.
  - Dòng 02 chứa số nguyên dương n là số lượng phần tử của tập $\mathcal{U}(p)$.
  - Dòng 03 chứa n số nguyên là các phần tử của tập $\mathcal{U}(p)$ (tập các ước nguyên tố dương của $p-1$). Các số nguyên cách nhau bởi 1 khoảng trắng.
  - Dòng 04 chứa số nguyên dương g.
  - **Lưu ý:** Tất cả các số ở tất cả các dòng đều là số nguyên không âm, được viết bằng các chữ số thập lục phân in hoa dưới dạng little endian.
- `test.out` là file text chứa dữ liệu đầu ra của chương trình. Trong file này, chương trình trả ra 1 dòng chứa số 0 nếu g không phải là căn nguyên thủy modulo p hoặc số 1 nếu g là một căn nguyên thủy modulo p.
- **Lưu ý:** File text chứa dữ liệu đầu vào và đầu ra không nhất thiết phải có tên `test.inp` và `test.out`.

- Thời gian chạy tối đa của chương trình là **60 giây** cho mỗi test. Khi quá thời gian quy định, chương trình sẽ bị ngắt.

1.4 Test mẫu

Phần này chỉ liệt kê một số test mẫu. Đối với các test mẫu còn lại, sinh viên xem trong thư mục chứa các test cho bài này.

- Test 01:

test.inp	test.out
3 1 2 2	1

Giải thích: Do $p = 3$ và $\mathcal{U}(p) = \mathcal{U}(3) = \{\, 2 \,\}$ nên với $g = 2$ ta có:

$$2^{\frac{2}{2}} = 2^1 = 2 \neq 1 \pmod 3$$

Suy ra, kết quả trả ra sẽ là `1`.

- Test 02:

test.inp	test.out
BF68CD1901 A 2 3 5 7 B D 11 31 71 D1 4	0

Giải thích: `0x1091DC86FB = 71166625531 = 2 * 3 * 5 * 7 * 11 * 11 * 13 * 17 * 19 * 23 * 29 + 1, 0xA = 10, 0x2 = 2, 0x3 = 3, 0x5 = 5, 0x7 = 7, 0xB = 11, 0xD = 13, 0x11 = 17, 0x13 = 19, 0x17 = 23, 0x1D = 29, 0x4 = 4`.

Do $p = 71166625531$ và $\mathcal{U}(p) = \mathcal{U}(71166625531) = \{\, 2, 3, 5, 7, 11, 13, 17, 19, 23, 29 \,\}$ nên với

$g = 4$ ta có:

$$4^{\frac{71166625530}{2}} = 4^{35583312765} = 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{3}} = 4^{23722208510} = 59559195843 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{5}} = 4^{14233325106} = 5987651467 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{7}} = 4^{10166660790} = 37749305871 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{11}} = 4^{6469693230} = 40329262424 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{13}} = 4^{5474355810} = 18345981194 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{17}} = 4^{4186272090} = 40362536818 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{19}} = 4^{3745611870} = 65391610887 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{23}} = 4^{3094201110} = 40653699497 \neq 1 \pmod{71166625531}$$
$$4^{\frac{71166625530}{29}} = 4^{2454021570} = 56654508055 \neq 1 \pmod{71166625531}$$

Suy ra, kết quả trả ra sẽ là `0`.

1.5 Cơ cấu các test và cách tính điểm

Có tất cả 100 test, mỗi test sẽ tương ứng với 1% điểm của bài này. Trong đó:

- Có 10/100 test mẫu (có đủ cả `test.inp` và `test.out`).
- Có 10/100 test công khai (chỉ có `test.inp`).
- Có 80/100 test bí mật (không được công bố như test mẫu và test công khai, mà chỉ dùng riêng cho việc tính điểm).

Các test mẫu và test công khai sẽ được cung cấp trong thư mục chứa các test trong bài này. Xét về độ khó, cơ cấu các test như sau:

- Kiểm tra căn nguyên thủy với p từ trên 256 đến 512 bit: 10/100 test.
- Kiểm tra căn nguyên thủy với p từ trên 128 đến 256 bit: 20/100 test.
- Kiểm tra căn nguyên thủy với p từ trên 64 đến 128 bit: 30/100 test.
- Kiểm tra căn nguyên thủy với p từ 64 bit trở xuống: 40/100 test.
- **Lưu ý:** Tất cả 100/100 test đều có g thỏa $g < p$.

2 (3 điểm) Hệ trao đổi khóa Diffie Hellman

2.1 Định nghĩa

Cho một số nguyên tố $p \in \mathcal{P}$ lẻ và số nguyên dương $g \in \mathbb{Z}^+$ là căn nguyên thủy modulo p. Alice và Bob thực hiện giao thức trao đổi khóa Diffie Hellman như sau:

1. Alice chọn số nguyên dương bí mật là a.
2. Alice gửi cho Bob giá trị công khai của Alice là $A = g^a \pmod{p}$.
3. Bob chọn số nguyên dương bí mật là b.

4. Bob gửi cho Alice giá trị công khai của Bob là $B = g^b \pmod p$.
5. Alice thu được khóa chung là $K = B^a = g^{ab} \pmod p$.
6. Bob thu được khóa chung là $K = A^b = g^{ab} \pmod p$.

2.2 Yêu cầu

Trong bài này, sinh viên viết chương trình dùng để mô phỏng giao thức trao đổi khóa Diffie Hellman. Chương trình thỏa các yêu cầu sau:

- Ngôn ngữ lập trình là `C++`.
- Toàn bộ mã nguồn được lưu thành một file duy nhất là `main.cpp`.
- Ngoại trừ các tính năng từ ngôn ngữ lập trình `C++` (phiên bản `C++17` trở xuống) và thư viện chuẩn của `C++` (`C++` Standard Library) là được cho phép sử dụng, toàn bộ các thư viện khác bị cấm.
- Sau khi biên dịch mã nguồn thành file `main`, chương trình được chạy như sau.

  ```
  $ .\main test.inp test.out
  ```

  Trong đó:

  - `test.inp` là file text chứa dữ liệu đầu vào của chương trình, gồm có 4 dòng:
    - Dòng 01 chứa số nguyên tố lẻ p.
    - Dòng 02 chứa số nguyên dương g là một căn nguyên thủy modulo p.
    - Dòng 03 chứa số nguyên dương a bí mật của Alice.
    - Dòng 04 chứa số nguyên dương b bí mật của Bob.
    - **Lưu ý:** Tất cả các số ở tất cả các dòng đều là số nguyên không âm, được viết bằng các chữ số thập lục phân in hoa dưới dạng little endian.
  - `test.out` là file text chứa dữ liệu đầu ra của chương trình, gồm có 3 dòng:
    - Dòng 01 chứa số nguyên dương $A = g^a \pmod p$ là giá trị công khai của Alice.
    - Dòng 02 chứa số nguyên dương $B = g^b \pmod p$ là giá trị công khai của Bob.
    - Dòng 03 chứa số nguyên dương $K = g^{ab} \pmod p$ là khóa chung của Alice và Bob.
    - **Lưu ý:** Tất cả các số ở tất cả các dòng đều là số nguyên không âm, được viết bằng các chữ số thập lục phân in hoa dưới dạng little endian.
  - **Lưu ý:** File text chứa dữ liệu đầu vào và đầu ra không nhất thiết phải có tên `test.inp` và `test.out`.
- Thời gian chạy tối đa của chương trình là **60 giây** cho mỗi test. Khi quá thời gian quy định, chương trình sẽ bị ngắt.

2.3 Test mẫu

Phần này chỉ liệt kê một số test mẫu. Đối với các test mẫu còn lại, sinh viên xem trong thư mục chứa các test cho bài này.

- Test 01:

test.inp	test.out
56 D1 21 12	9 8 55

Giải thích: `0x65 = 101, 0x1D = 29, 0x12 = 18, 0x21 = 33`. Khi đó, với $p = 101$ ta có:

$$A = g^a = 29^{18} = 9 \pmod{101}$$
$$B = g^b = 29^{33} = 8 \pmod{101}$$
$$K = g^{ab} = A^b = B^a = 29^{18\times 33} = 85 \pmod{101}$$

Suy ra, kết quả trả ra sẽ là `0x9 = 9, 0x8 = 8, 0x55 = 85`.

- Test 02:

test.inp	test.out
76 B2 A1 14	42 33 31

Giải thích: `0x67 = 103, 0x2B = 43, 0x1A = 26, 0x41 = 65`. Khi đó, với $p = 103$ ta có:

$$A = g^a = 43^{26} = 36 \pmod{103}$$
$$B = g^b = 43^{65} = 51 \pmod{103}$$
$$K = g^{ab} = A^b = B^a = 43^{26\times 65} = 19 \pmod{103}$$

Suy ra, kết quả trả ra sẽ là `0x24 = 36, 0x33 = 51, 0x13 = 19`.

2.4 Cơ cấu các test và cách tính điểm

Có tất cả 100 test, mỗi test sẽ tương ứng với 1% điểm của bài này. Trong đó:

- Có 10/100 test mẫu (có đủ cả `test.inp` và `test.out`).
- Có 10/100 test công khai (chỉ có `test.inp`).
- Có 80/100 test bí mật (không được công bố như test mẫu và test công khai, mà chỉ dùng riêng cho việc tính điểm).

Các test mẫu và test công khai sẽ được cung cấp trong thư mục chứa các test trong bài này.
Xét về độ khó, cơ cấu các test như sau:

- Mô phỏng giao thức Diffie Hellman với p từ trên 256 đến 512 bit: 10/100 test.
- Mô phỏng giao thức Diffie Hellman với p từ trên 128 đến 256 bit: 20/100 test.
- Mô phỏng giao thức Diffie Hellman với p từ trên 64 đến 128 bit: 30/100 test.
- Mô phỏng giao thức Diffie Hellman với p từ 64 bit trở xuống: 40/100 test.
- **Lưu ý:** Tất cả 100/100 test đều có g, a, b thỏa $g, a, b < p$.

3 (3 điểm) Hệ mã hóa ElGamal

3.1 Định nghĩa

Hệ mã hóa ElGamal gồm có 3 phần:

- **Sinh khóa.**

  Trong phần này, khóa được tạo ra như sau:

  1. Sinh ngẫu nhiên số nguyên tố $p \in \mathcal{P}$ lẻ và số nguyên dương $g \in \mathbb{Z}^+$ là căn nguyên thủy modulo p.
  2. Sinh ngẫu nhiên số nguyên dương $x \in \mathbb{Z}^+$ thỏa $x < p$.
  3. Tính $h = g^x \pmod p$
  4. Khóa công khai dùng để mã hóa là $e = (p, g, h)$. Khóa bí mật dùng để giải mã là $d = (p, g, x)$.

- **Mã hóa.**

  Trong phần này, với khóa công khai $e = (p, g, h)$, thông điệp m được mã hóa như sau:

  1. Sinh ngẫu nhiên số nguyên dương $y \in \mathbb{Z}^+$ thỏa $y < p$.
  2. Tính $s = h^y \pmod p$.
  3. Tính $c_1 = g^y \pmod p$.
  4. Tính $c_2 = ms \pmod p$.
  5. Bản mã của m là $c = (c_1, c_2)$.

  Lưu ý: Do $h = g^x \pmod p$ và $s = h^y \pmod p$ nên $s = g^{xy} \pmod p$.

- **Giải mã.**

  Trong phần này, với khóa bí mật $d = (p, g, x)$, bản mã $c = (c_1, c_2)$ được giải mã ra như sau:

  1. Tính $s = c_1^x \pmod p$.
  2. Tính s^{-1} là phần tử nghịch đảo modulo p của s, tức là $s^{-1}s = 1 \pmod p$.
  3. Thông điệp giải ra từ bản mã $c = (c_1, c_2)$ là $m = c_2 s^{-1} \pmod p$.

  Lưu ý: Do $c_1 = g^y \pmod p$ và $s = c_1^x \pmod p$ nên $s = g^{xy} \pmod p$, tức giống với s ở phần mã hóa.

3.2 Yêu cầu

Trong bài này, sinh viên viết chương trình dùng để mô phỏng một số bước trong hệ mã hóa ElGamal. Chương trình thỏa các yêu cầu sau:

- Ngôn ngữ lập trình là `C++`.
- Toàn bộ mã nguồn được lưu thành một file duy nhất là `main.cpp`.
- Ngoại trừ các tính năng từ ngôn ngữ lập trình `C++` (phiên bản `C++17` trở xuống) và thư viện chuẩn của `C++` (`C++` Standard Library) là được cho phép sử dụng, toàn bộ các thư viện khác bị cấm.
- Sau khi biên dịch mã nguồn thành file `main`, chương trình được chạy như sau.

```
$ .\main test.inp test.out
```

Trong đó:

- `test.inp` là file text chứa dữ liệu đầu vào của chương trình, gồm có 5 dòng:
  - Dòng 01 chứa số nguyên tố lẻ p.
  - Dòng 02 chứa số nguyên dương g là một căn nguyên thủy modulo p.
  - Dòng 03 chứa số nguyên dương x, cùng với $(p, g, x) = d$ tạo thành khóa bí mật.
  - Dòng 04 chứa số nguyên c_1 là phần thứ nhất của bản mã $c = (c_1, c_2)$.
  - Dòng 05 chứa số nguyên c_2 là phần thứ hai của bản mã $c = (c_1, c_2)$.
  - **Lưu ý:** Tất cả các số ở tất cả các dòng đều là số nguyên không âm, được viết bằng các chữ số thập lục phân in hoa dưới dạng little endian.
- `test.out` là file text chứa dữ liệu đầu ra của chương trình, gồm có 2 dòng:
  - Dòng 01 chứa số nguyên dương $h = g^x \pmod p$ với $h < p$, là phần thứ ba trong khóa công khai $e = (p, g, h)$.
  - Dòng 02 chứa số nguyên dương m với $m < p$, là thông điệp của bản mã $c = (c_1, c_2)$.
  - **Lưu ý:** Tất cả các số ở tất cả các dòng đều là số nguyên không âm, được viết bằng các chữ số thập lục phân in hoa dưới dạng little endian.
- **Lưu ý:** File text chứa dữ liệu đầu vào và đầu ra không nhất thiết phải có tên `test.inp` và `test.out`.

- Thời gian chạy tối đa của chương trình là **60 giây** cho mỗi test. Khi quá thời gian quy định, chương trình sẽ bị ngắt.

3.3 Test mẫu

Phần này chỉ liệt kê một số test mẫu. Đối với các test mẫu còn lại, sinh viên xem trong thư mục chứa các test cho bài này.

- Test 01:

test.inp	test.out
16 71 A1 D5 14	F5 F5

Giải thích: `0x61 = 97, 0x17 = 23, 0x1A = 26, 0x5D = 93, 0x41 = 65`. Khi đó, với $p = 97$ ta có:

$$h = g^x = 23^{26} = 95 \pmod{97}$$
$$s = c_1^x = 93^{26} = 16 \pmod{97}$$
$$16 \times 91 = 1 \pmod{97} \implies s^{-1} = 91 \pmod{97}$$
$$m = c_2 s^{-1} = 65 \times 91 = 95 \pmod{97}$$

Suy ra, kết quả trả ra sẽ là `0x5F = 95, 0x5F = 95`.

- Test 02:

test.inp	test.out
35 31 D2 B2 15	84 34

Giải thích: `0x53 = 83, 0x13 = 19, 0x2D = 45, 0x2B = 43, 0x51 = 81`. Khi đó, với $p = 83$ ta có:

$$h = g^x = 19^{45} = 72 \pmod{83}$$
$$s = c_1^x = 43^{45} = 52 \pmod{83}$$
$$52 \times 8 = 1 \pmod{83} \implies s^{-1} = 8 \pmod{83}$$
$$m = c_2 s^{-1} = 81 \times 8 = 67 \pmod{83}$$

Suy ra, kết quả trả ra sẽ là `0x48 = 72, 0x43 = 67`.

3.4 Cơ cấu các test và cách tính điểm

Có tất cả 100 test, mỗi test sẽ tương ứng với 1% điểm của bài này. Trong đó:

- Có 10/100 test mẫu (có đủ cả `test.inp` và `test.out`).
- Có 10/100 test công khai (chỉ có `test.inp`).
- Có 80/100 test bí mật (không được công bố như test mẫu và test công khai, mà chỉ dùng riêng cho việc tính điểm).

Các test mẫu và test công khai sẽ được cung cấp trong thư mục chứa các test trong bài này. Xét về độ khó, cơ cấu các test như sau:

- Mô phỏng hệ mã hóa ElGamal với p từ trên 256 đến 512 bit: 10/100 test.
- Mô phỏng hệ mã hóa ElGamal với p từ trên 128 đến 256 bit: 20/100 test.
- Mô phỏng hệ mã hóa ElGamal với p từ trên 64 đến 128 bit: 30/100 test.
- Mô phỏng hệ mã hóa ElGamal với p từ 64 bit trở xuống: 40/100 test.
- **Lưu ý:** Tất cả 100/100 test đều có g, x, c_1, c_2 thỏa $g, x, c_1, c_2 < p$.

4 (3 điểm) Hệ chữ ký ElGamal

4.1 Định nghĩa

Hệ chữ ký ElGamal gồm có 3 phần:

- **Sinh khóa.**

  Trong phần này, khóa được tạo ra như sau:

  1. Sinh ngẫu nhiên số nguyên tố $p \in \mathcal{P}$ lẻ và số nguyên dương $g \in \mathbb{Z}^+$ là căn nguyên thủy modulo p.
  2. Sinh ngẫu nhiên số nguyên dương $x \in \mathbb{Z}^+$ thỏa $x < p$.

3. Tính $y = g^x \pmod{p}$
4. Khóa công khai dùng để xác thực là $v = (p, g, y)$. Khóa bí mật dùng để ký là $s = (p, g, x)$.

- **Ký.**

  Trong phần này, với khóa bí mật $s = (p, g, x)$, thông điệp m được ký như sau:

  1. Sinh ngẫu nhiên số nguyên dương $k \in \mathbb{Z}^+$ thỏa k và $p - 1$ là nguyên tố cùng nhau, tức là $(k, p - 1) = 1$.
  2. Tính $r = g^k \pmod{p}$.
  3. Tính $h = (m - xr)k^{-1} \pmod{p - 1}$ với k^{-1} là phần tử nghịch đảo modulo $p - 1$ của k, tức là $kk^{-1} = 1 \pmod{p - 1}$
  4. Chữ ký cho thông điệp m là $c = (r, h)$.

- **Xác thực.**

  Trong phần này, với khóa công khai $v = (p, g, y)$, thông điệp m cùng với chữ ký $c = (r, h)$ được xác thực như sau:

  1. Kiểm tra r và h là các số nguyên thỏa $0 < r < p$ và $0 < h < p - 1$ hay không?
  2. Kiểm tra giá trị g^m và $y^r r^h$ có bằng nhau hay không?
  3. Nếu cả hai điều kiện trên thì chữ ký xác thực thành công. Nếu ngược lại thì chữ ký xác thực thất bại.

Lưu ý:

- Do $h = (m - xr)k^{-1} \pmod{p - 1}$ nên $m = xr + hk \pmod{p - 1}$. Suy ra:

$$g^m = g^{xr+hk} = y^r r^h \pmod{p}$$

- Trong thực tế, nhằm đảm bảo an toàn chống lại tấn công giả mạo chữ ký dựa trên việc chọn m, hệ chữ ký ElGamal yêu cầu sử dụng thêm một hàm băm an toàn H dùng để băm toàn bộ m thành $H(m)$. Tuy nhiên, trong đồ án này, ta sẽ tạm thời bỏ qua hàm H.

4.2 Yêu cầu

Trong bài này, sinh viên viết chương trình dùng để mô phỏng quá trình xác thực của hệ chữ ký ElGamal. Chương trình thỏa các yêu cầu sau:

- Ngôn ngữ lập trình là `C++`.
- Toàn bộ mã nguồn được lưu thành một file duy nhất là `main.cpp`.
- Ngoại trừ các tính năng từ ngôn ngữ lập trình `C++` (phiên bản `C++17` trở xuống) và thư viện chuẩn của `C++` (`C++` Standard Library) là được cho phép sử dụng, toàn bộ các thư viện khác bị cấm.
- Sau khi biên dịch mã nguồn thành file `main`, chương trình được chạy như sau.

```
$ .\main test.inp test.out
```

  Trong đó:

  – `test.inp` là file text chứa dữ liệu đầu vào của chương trình, gồm có 6 dòng:

    - Dòng 01 chứa số nguyên tố lẻ p.
    - Dòng 02 chứa số nguyên dương g là một căn nguyên thủy modulo p.
    - Dòng 03 chứa số nguyên dương y, cùng với $(p, g, y) = v$ tạo thành khóa công khai dùng để xác thực.
    - Dòng 04 chứa số nguyên dương m ($m < p - 1$), là thông điệp được ký.
    - Dòng 05 chứa số nguyên r là phần thứ nhất của chữ ký $c = (r, h)$.
    - Dòng 06 chứa số nguyên h là phần thứ hai của chữ ký $c = (r, h)$.
    - **Lưu ý:** Tất cả các số ở tất cả các dòng đều là số nguyên không âm, được viết bằng các chữ số thập lục phân in hoa dưới dạng little endian.
  - `test.out` là file text chứa dữ liệu đầu ra của chương trình. Trong file này, chương trình trả ra 1 dòng chứa số 0 nếu chữ ký xác thực thất bại hoặc số 1 nếu chữ ký xác thực thành công.
  - **Lưu ý:** File text chứa dữ liệu đầu vào và đầu ra không nhất thiết phải có tên `test.inp` và `test.out`.

- Thời gian chạy tối đa của chương trình là **60 giây** cho mỗi test. Khi quá thời gian quy định, chương trình sẽ bị ngắt.

4.3 Test mẫu

Phần này chỉ liệt kê một số test mẫu. Đối với các test mẫu còn lại, sinh viên xem trong thư mục chứa các test cho bài này.

- Test 01:

test.inp	test.out
16 71 F5 24 A5 A5	1

Giải thích: `0x61 = 97, 0x17 = 23, 0x5F = 95, 0x42 = 66, 0x5A = 90`. Khi đó, với $p = 97$ ta có:

$$\begin{aligned} g^m &= 23^{66} = 85 \pmod{97} \\ y^r &= 95^{90} = 47 \pmod{97} \\ r^h &= 90^{90} = 8 \pmod{97} \\ y^r r^h &= 47 \times 8 = 85 \pmod{97} \end{aligned}$$

Vì $g^m = 85 = y^r r^h \pmod{97}$ nên kết quả trả ra sẽ là `1`.

- Test 02:

test.inp	test.out
35 31 84 24 93 E4	0

Giải thích: `0x53 = 83, 0x13 = 19, 0x48 = 72, 0x42 = 66, 0x39 = 57, 0x4E = 78`. Khi đó, với $p = 83$ ta có:

$$\begin{aligned} g^m &= 19^{66} = 78 \pmod{83} \\ y^r &= 72^{57} = 66 \pmod{83} \\ r^h &= 57^{78} = 49 \pmod{83} \\ y^r r^h &= 66 \times 49 = 80 \pmod{83} \end{aligned}$$

Vì $g^m = 78 \neq 80 = y^r r^h \pmod{83}$ nên kết quả trả ra sẽ là `0`.

4.4 Cơ cấu các test và cách tính điểm

Có tất cả 100 test, mỗi test sẽ tương ứng với 1% điểm của bài này. Trong đó:

- Có 10/100 test mẫu (có đủ cả `test.inp` và `test.out`).
- Có 10/100 test công khai (chỉ có `test.inp`).
- Có 80/100 test bí mật (không được công bố như test mẫu và test công khai, mà chỉ dùng riêng cho việc tính điểm).

Các test mẫu và test công khai sẽ được cung cấp trong thư mục chứa các test trong bài này. Xét về độ khó, cơ cấu các test như sau:

- Xác thực hệ chữ ký ElGamal với p từ trên 256 đến 512 bit: 10/100 test.
- Xác thực hệ chữ ký ElGamal với p từ trên 128 đến 256 bit: 20/100 test.
- Xác thực hệ chữ ký ElGamal với p từ trên 64 đến 128 bit: 30/100 test.
- Xác thực hệ chữ ký ElGamal với p từ 64 bit trở xuống: 40/100 test.
- **Lưu ý:** Tất cả 100/100 test đều có g, y, m, r, h thỏa $g, y, r < p$ và $m, h < p - 1$.

5 Các quy định khác về đồ án

- Đồ án được thực hiện cá nhân.
- Thời gian thực hiện là 3 tuần tính từ lúc đồ án được công bố chính thức bằng thông báo.
- Cấu trúc bài làm như sau:

```
<MSSV>.zip
├── project_02_01
│   └── main.cpp
├── project_02_02
│   └── main.cpp
├── project_02_03
│   └── main.cpp
└── project_02_04
    └── main.cpp
```

  Trong đó:

  - `<MSSV>` là mã số sinh viên của người nộp.
  - `.zip` là định dạng nén cho bài làm (định dạng ZIP).

  - `project_02_01` là nơi chứa mã nguồn cho phần Căn nguyên thủy modulo p.
  - `project_02_02` là nơi chứa mã nguồn cho phần Hệ trao đổi khóa Diffie Hellman.
  - `project_02_03` là nơi chứa mã nguồn cho phần Hệ mã hóa ElGamal.
  - `project_02_04` là nơi chứa mã nguồn cho phần Hệ chữ ký ElGamal.
  - `main.cpp` là các file chứa toàn bộ mã nguồn tương ứng với từng phần nêu trên.
- Trường hợp chương trình của phần nào bị lỗi không biên dịch được thì phần đó không có điểm.
- Trường hợp chương trình biên dịch được nhưng không trả ra được kết quả (do gặp lỗi khi chạy hoặc quá thời gian quy định) ở test nào thì test đó không có điểm.
- Thư viện chuẩn của `C++` (`C++` Standard Library) là các thư viện được liệt kê ở đây: `https://en.cppreference.com/w/cpp/standard_library`

  Cần lưu ý về phiên bản `C++` để lựa chọn thư viện thích hợp.
- Mọi thắc mắc vui lòng gửi về email: `nvqhuy@fit.hcmus.edu.vn`