TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN, ĐHQG-HCM
KHOA CÔNG NGHỆ THÔNG TIN

THỰC HÀNH

MÃ HÓA MẬT MÃ

Đồ án vấn đáp: Bảo mật Cơ sở dữ liệu

Sinh viên:
21120107 - Nguyễn Minh Nhật

Giảng viên:
PGS. TS. Nguyễn Đình Thúc
ThS. Nguyễn Văn Quang Huy

Mục lục

1 Trình bày bài toán

$$D = \{f_1, ..., f_n\}$$
$$f_i : \text{file/record/row} \rightarrow \text{Item}$$

- Cơ sở dữ liệu trong ngữ cảnh này được định nghĩa là một tập hợp các tệp tin, dòng dữ liệu. Một tệp f_i chứa các trường thông tin.
- Bảo mật cơ sở dữ liệu là công việc đòi hỏi thiết kế kỹ lưỡng. Trong quá trình sử dụng cơ sở dữ liệu, nhu cầu **tìm kiếm thông tin** phải được cân bằng với nhu cầu **bảo vệ thông tin** trên cơ sở dữ liệu.
- Ví dụ:

MSSV	**Họ tên**	**Trung Bình**
21120107	Tony	6.9
21120350	Steve	1.1
21120507	Bruce	7.5

  - Khi đó item/attribute là các lựa chọn để mình bảo mật
  - Chẳng hạn cần bảo mật điểm, thì ta cần chọn bảo mật giữa:
    * MSSV
    * Họ tên
    * Trung bình

2 Trình bày ý tưởng

2.1 Pha mã hóa

1. Đọc database (dạng đuôi **.csv**) theo vị trí cột cần mã hóa.
2. Tạo khóa độ dài 256 bit để mã hóa và giải mã.
3. Thực hiện mã hóa:
   - Chia dữ liệu thành các block 256 bit.
   - Sử dụng mã hóa ChaCha20, thuật toán mã hóa đối xứng Stream Cipher được đánh giá là nhanh và rất bảo mật [1].
4. Sử dụng định lý đồng dư Trung Hoa

$$\begin{cases} C \equiv f_1 \pmod{p_1} \\ C \equiv f_2 \pmod{p_2} \\ \vdots \\ C \equiv f_n \pmod{p_n} \end{cases}$$

5. Lưu lại cơ sở dữ liệu sau khi mã hóa được thay bằng các số nguyên tố p_i.

2.2 Pha giải mã

1. Đọc vào database, kèm vị trí cột, hàng và số nguyên tố p (khóa người dùng).
2. Kiểm tra khóa p có khớp với khóa tại hàng và cột được cung cấp.
3. Nếu khớp, chia đoạn mã hóa thành các block và giải mã bằng thuật toán ChaCha20.
4. Trả ra kết quả.

3 Code

3.1 Code mã hóa

1. Luồng điều khiển chính của mã hóa:

```
# Install the library...

if __name__ == "__main__":
    # ...

    # Load database
    col = read_csv(file_path, column_index)

    # Generate key
    gen_key()

    # Encode
    en_c = encrypt_column(col)

    # CRT
    crt_c = generate_prime(en_c)
    chinese_remainder_theorem(en_c, crt_c)

    # Update database
    output_file = file_path.replace('.csv', '_encode.csv')
    update_csv(file_path, column_index, crt_c, output_file)
```

2. Hàm `read_csv()` đọc file csv theo đường dẫn và cột quy định.

3. Hàm `gen_key()` sinh khóa random 256 bit.

4. Hàm `encrypt_column()` mã hóa dữ liệu theo từng dòng, mỗi dòng gọi hàm `encrypt_data` để mã hóa:

```
def chacha20_encrypt(key, plaintext):
    cipher = ChaCha20.new(key=key)
    ciphertext = cipher.encrypt(plaintext)
    return cipher.nonce + ciphertext

def encrypt_data(plaintext, key):
    # Break the text into smaller blocks
    block_size = 32
    plaintext_blocks = [plaintext[i:i+block_size] for i in range(0,
        len(plaintext), block_size)]

    # Encrypt each small block using ChaCha20
    ciphertext_blocks = []
    for block in plaintext_blocks:
```

```
            block_bytes = block
            ciphertext_block = chacha20_encrypt(key, block_bytes)
            ciphertext_blocks.append(ciphertext_block)

        # Combine the encryption blocks into a complete encryption string
        ciphertext = b"".join(ciphertext_blocks)

        # Convert the encoded string to hex format
        ciphertext_hex = binascii.hexlify(ciphertext).decode("utf-8")

        return ciphertext_hex
```

- Kích thước mỗi block là 32 byte.
- Thêm phần nonce kích thước 8 byte thì phần sau khi mã có thể tới 40 byte.
- Nối tất cả các block thành 1 xâu mã hóa hoàn chỉnh và chuyển về dạng hex.

5. Hàm `generate_prime()` sinh ra ngẫu nhiên số nguyên tố có số bit lớn hơn từ 1 đến 7 bit xâu sau khi mã hóa.
6. Hàm `chinese_remainder_theorem()` tính số C.
7. Lưu kết quả sau khi mã hóa, bằng các số nguyên tố.

3.2 Code giải mã

1. Luồng hoạt động chính của giải mã:

```
# Install the library...

if __name__ == "__main__":
    # ...

    # Load database
    col = read_csv(file_path, column_index)

    # Return results
    if pass_user != col[row_index - 1]:
        print("Access failed!")
    else:
        print("Accessed successfully!")
        print(f"Result: {decrypt_row(col, pass_user)}")
```

2. Hàm `read_csv()` đọc file csv theo đường dẫn và cột quy định.
3. Hàm `decrypt_row()` giải mã dữ liệu tại cột và khóa tương ứng.

```
def decrypt_row(encrypted_column, encrypted_hex):
    with open("../Key/key.txt", "rb") as file:
        key = file.read()
    with open("../Key/CRT.txt", "rb") as file:
        crt = file.read()

    block_size = 32

    encrypted_value = mod_hex(crt, encrypted_hex)
    if len(encrypted_value) % 2 != 0:
        encrypted_value = '0' + encrypted_value

    # Decode the encrypted string
    ciphertext_bytes = binascii.unhexlify(encrypted_value)
    decrypted_text_blocks = []
    for i in range(0, len(ciphertext_bytes), block_size+8):
        ciphertext_block = ciphertext_bytes[i:i+block_size+8]
        decrypted_block = chacha20_decrypt(key, ciphertext_block)
        decrypted_text_blocks.append(decrypted_block)

    # Combine decoding blocks into a complete text string
    decrypted_text = b"".join(decrypted_text_blocks).decode("utf-8")

    return decrypted_text
```

4. Lấy C mod cho số nguyên tố là khóa đầu vào để lấy xâu mã hóa.
5. Ngược lại với việc mã hóa, ta phân nhỏ xâu mã hóa thành các block con để giải mã.
6. Lưu ý một block con sẽ có kích thước 40 byte (block size + nonce size).
7. Hợp các xâu sau khi giải mã và trả kết quả.

4 Thực thi chương trình

4.1 Mã hóa

```
python main_enc.py <file_path_csv> <column_index>
```

- <file_path_csv> là đường dẫn đến file dữ liệu, lưu trữ ở dạng csv.
- <column_index> là vị trí cột cần được mã hóa.

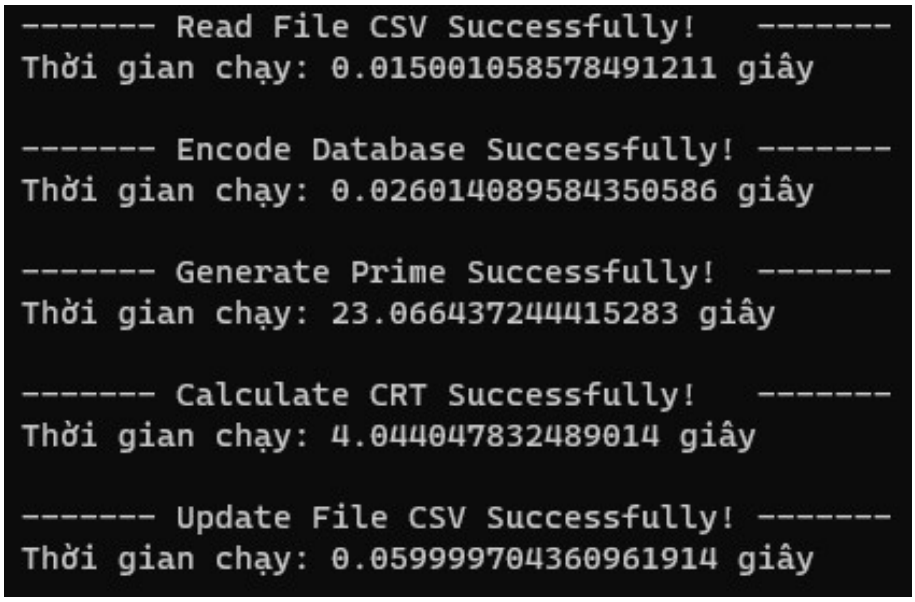
4.2 Giải mã

```
python main_dec.py <file_path_csv> <column_index> <row_index> <pass>
```

- <file_path_csv> là đường dẫn đến file dữ liệu, lưu trữ ở dạng csv.
- <column_index> là vị trí cột cần được giải mã.
- <row_index> là vị trí hàng cần được giải mã.
- <pass> là khóa cần được xác thực.

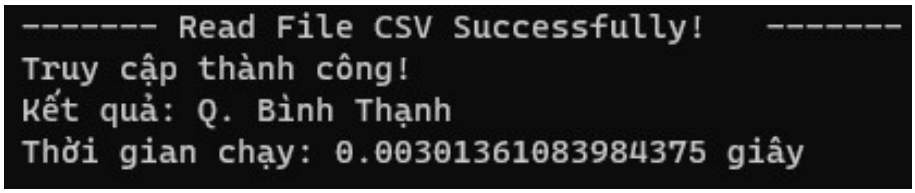
5 Kết quả chương trình

- Kết quả thời gian chạy mã hóa từng pha:



Hình 1: Kết quả mã hóa cho dữ liệu 3407 dòng

- Kết quả thời gian giải mã:



Hình 2: Thời gian cho giải mã thành công

6 Thách thức & Mở rộng

6.1 Thách thức

- Kích thước cho CRT tương đối lớn khi lượng dữ liệu tăng lên:

Số dòng	Kích thước CRT	Thời gian mã hóa	Thời gian giải mã
58	2230 byte	0.716	0.002
260	7481 byte	2.224	0.001
3406	122902 byte	27.212	0.003

6.2 Mở rộng

6.2.1 Vấn đề

Phần mở rộng dưới đây để cố gắng giải quyết vấn đề CRT sinh ra tương đối lớn, chiếm không gian lưu trữ.

6.2.2 Ý tưởng

- Dựa trên ý tưởng cũ, giữ nguyên các thuật toán CRT, mã hóa ChaCha20, sinh số nguyên tố.
- Chia dữ liệu có n dòng thành $\sqrt{n}$ nhóm, mỗi nhóm có $\sqrt{n}$ hàng.
- Thực hiện mã hóa hai tầng:
  - Tầng mã hóa cho từng nhóm: dùng thuật toán CRT để có được số C_i đại diện cho nhóm i.
  - Tầng mã hóa cho các số C_i đại diện nhóm:
    1. Đặt $m = \sqrt{n}$, ta có:

$$X = \{x_1, x_2, ..., x_m\}, Y = \{y_1, y_2, ..., y_m\}$$

$$CRT = \{C_1, C_2, ..., C_m\}$$

    2. Khi đó, định nghĩa:

$$X_{pow} \times CRT = Y$$

Với:

$$X_{pow} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^{m-1} \\ 1 & x_2 & x_2^2 & \dots & x_2^{m-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_m & x_m^2 & \dots & x_m^{m-1} \end{bmatrix}$$

$$CRT = \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_m \end{bmatrix}, Y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}$$

3. Ý nghĩa của phép nhân ma trận là: ánh xạ dãy CRT sang một tập X sinh ngẫu nhiên (khác nhau từng đôi một).
4. Nhờ vậy, nếu X là tập các số thực tương đối nhỏ, thì kích thước lưu trữ sẽ ít hơn nhiều (trên lý thuyết là vậy).
5. Cuối cùng là thực hiện CRT lần nữa cho tập X.

- Để kiểm chứng suy nghĩ, em đã thực hiện code cho tập X là số nguyên nhỏ hơn 1000.

6.2.3 Code

- **Quá trình mã hóa:**

1. Luồng điều khiển chính của mã hóa:

```
# Install the library...
if __name__ == "__main__":
    # ...

    # Load database
    col = read_csv(file_path, column_index)

    # Generate key
    gen_key()

    # Encode
    en_c = encrypt_column(col)

    # CRT
    crt_c = generate_prime(en_c)
    crts = chinese_remainder_theorem_step1(en_c, crt_c)

    # Mapping through set x
    x_hex = gen_x(crts)
    prime_x = generate_prime(x_hex)
    with open("../Key/prime_s2.txt", "w") as file:
        for p in prime_x:
            file.write(p + "\n")
    crt_x = chinese_remainder_theorem_step2(x_hex, prime_x)

    # Update database
    output_file = file_path.replace('.csv', '_encode.csv')
    update_csv(file_path, column_index, crt_c, output_file)
```

2. Hàm `chinese_remainder_theorem_step1()` được sử dụng lại từ trước để thực hiện CRT trên các nhóm nhỏ.
3. Hàm `gen_x()` thực hiện ánh xạ các số CRT qua tập X bằng nhân ma trận.
4. Tiếp tục thực hiện CRT lần 2 cho tập các số X.
5. Lưu kết quả mã hóa.

- **Quá trình giải mã:**

1. Luồng hoạt động chính của giải mã:

```
# Install the library...
if __name__ == "__main__":
    # ...

    # Load database
    col = read_csv(file_path, column_index)

    # Return results
    if pass_user != col[row_index - 1]:
        print("Access failed!")
    else:
        print("Accessed successfully!")
        print(f"Result: {decrypt_row(len(col), row_index,
            pass_user)}")
```

2. Hàm `decrypt_row()` thực hiện giải mã hai lớp.

6.2.4 Kết quả

- **Thời gian thực hiện mã hóa:**

```
------- Read File CSV Successfully!   -------
Thời gian chạy: 0.0476827621459960 94 giây

------- Encode Database Successfully! -------
Thời gian chạy: 0.10107564926147461 giây

------- Generate Prime Successfully!  -------
Thời gian chạy: 29.531025171279907 giây

---- Calculate CRT Step 1 Successfully!  ----
Thời gian chạy: 0.12499928474426 27 giây

------- Generate Prime Successfully!  -------
---- Calculate CRT Step 2 Successfully!  ----
Thời gian chạy: 1.34801888465881 35 giây

------- Update File CSV Successfully! -------
Thời gian chạy: 0.11299324035644531 giây
```

Hình 3: Thời gian cho mã hóa dữ liệu 3406 dòng

- **Thời gian thực hiện giải mã:**

```
------- Read File CSV Successfully!   -------
Truy cập thành công!
Kết quả: Q. Bình Thạnh
Thời gian chạy: 16.35087752342224 giây
```

Hình 4: Thời gian cho giải mã dữ liệu thành công

7 Tham khảo

Tài liệu

[1] Karthikeyan Nagaraj. *Understanding ChaCha20 Encryption: A Secure and Fast Algorithm for Data Protection | 2023*. https://cyberw1ng.medium.com/understanding-chacha20-encryption-a-secure-and-fast-algorithm-for-data-protection-2023-a80c208c1401. Jan. 2023.