BÁO CÁO ĐỒ ÁN CUỐI KÌ MÔN MẬT MÃ HỌC

Image Encryption using Homomorphic Encryption

Bùi Quốc Huy, Ngô Đức Trí

19520588,19521044

July, 2021

University of Information Technology, VNU-HCM

TABLE OF CONTENTS

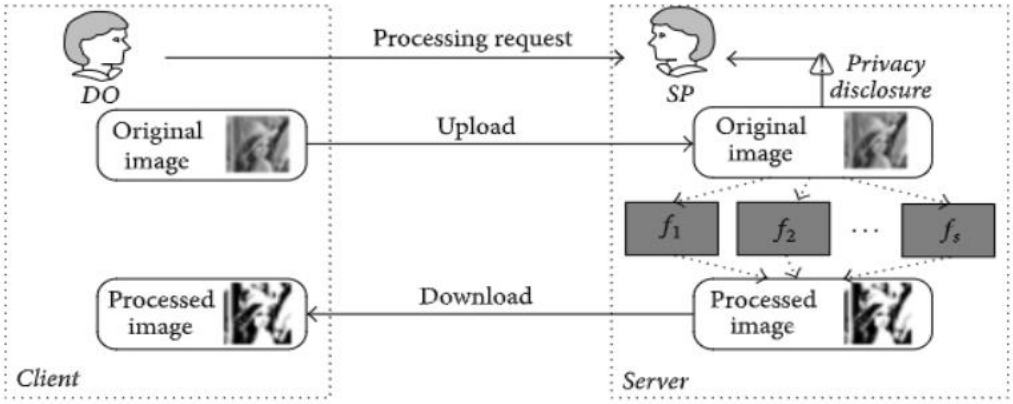
Page

1. SCENERIO

Trong thời đại công nghệ phát triển như hiện nay, việc lưu trữ các thông tin thông qua các bên thứ ba như các dịch vụ đám mây đã trở nên rất phổ biến. Người dùng có thể lưu trữ rất nhiều dữ liệu, âm thanh, hình ảnh và có thể truy cập các dữ liệu này ở mọi nơi, mọi lúc. Cũng như có thể chỉnh sửa, phân tích nó trên các đám mây. Việc sử dụng các tài nguyên tại các bên thứ 3 để tính toán sẽ trở nên dễ dàng , tiết kiệm chi phí hơn so với việc sử dụng tài nguyên nội bộ. Tuy nhiên kéo theo đó là việc bị lộ các thông tin nhạy cảm mà người dùng không muốn chia sẻ khi sử dụng các dịch vụ của bên thứ ba. Nếu người dùng sử dụng các thuật toán mã hóa thông thường thì việc bên thứ ba chỉ có thể tính toán trên nó nếu khi giải mã nó ra bản rõ, điều đó ảnh hưởng tới dữ liệu cá nhân của người dùng.

Figure 1:

Các thông tin của chúng ta sẽ bị lộ với bên thứ ba nếu như không mã hóa hoặc mã hóa theo các thuật toán cổ điển



2. RESEARCH MOTIVATIONS

Để thông tin của người dùng được an toàn không chỉ với các tin tặc mà còn với cả bên thứ ba, chúng ta sẽ sử dụng Homomorphic Encryption. Đây là một thuật toán nổi tiếng với khả năng tính toán trên bảng mã, ta không cần phải giải mã ra mà vẫn có thể thay đổi được nội dung của dữ liệu. Ta hoàn toàn có thể cho phép dịch vụ của bên thứ ba thực hiện tính toán, chỉnh sửa trên chính dữ liệu đã mã hóa của ta trong khi họ không có cách nào để có thể tìm hiểu được dữ liệu gốc của ta là gì. Thực tế thì Homomorphic Encryption đã được áp dụng rất nhiều trong lĩnh vực như y học để bảo vệ cho các thông tin của bệnh nhân như thông tin cá nhân, thông tin bệnh lý, trong các lĩnh vực như e-voting, e-cash(chuyển tiền mà mà không muốn hệ thống biết được giá trị) … nhờ vào những đặc trưng nổi bật của nó.

Figure 2

Một bài báo nói về việc sử dụng Homomorphic-Encryption trong việc bảo mật thông tin

Homomorphic-Encrypted Volume Rendering

Sebastian Mazza, Daniel Patel, and Ivan Viola

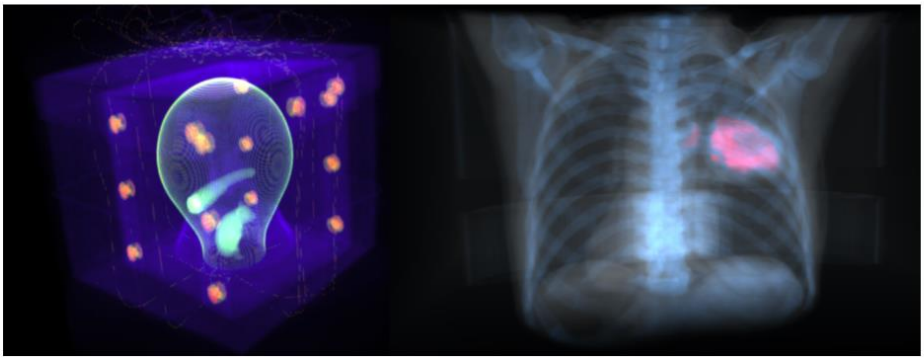


Fig. 1. Two decrypted images rendered by our simplified transfer function approach. The left image is rendered from a CT scan of a water globe and other objects inside a box that is wrapped inside a present box [12]. The dataset used for rendering the right image is a CT/PET scan of the thorax from a patient with lung cancer (*G0061/Adult-47358/7.0* from the *Lung-PET-CT-Dx* dataset [4, 20]).

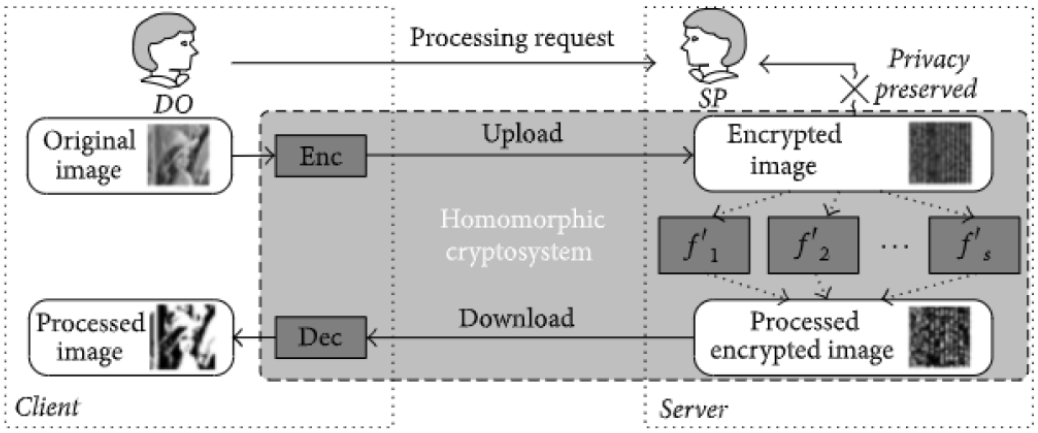
Abstract—Computationally demanding tasks are typically calculated in dedicated data centers, and real-time visualizations also follow this trend. Some rendering tasks, however, require the highest level of confidentiality so that no other party, besides the owner, can read or see the sensitive data. Here we present a direct volume rendering approach that performs volume rendering directly on encrypted volume data by using the homomorphic Paillier encryption algorithm. This approach ensures that the volume data and rendered image are uninterpretable to the rendering server. Our volume rendering pipeline introduces novel approaches for encrypted-data compositing, interpolation, and opacity modulation, as well as simple transfer function design, where each of these routines maintains the highest level of privacy. We present performance and memory overhead analysis that is associated with our privacy-preserving scheme. Our approach is open and secure by design, as opposed to secure through obscurity. Owners of the data only have to keep their secure key confidential to guarantee the privacy of their volume data and the rendered images. Our work is, to our knowledge, the first privacy-preserving remote volume-rendering approach that does not require that any server involved be trustworthy; even in cases when the server is compromised, no sensitive data will be leaked to a foreign party.

Index Terms—Volume Rendering, Transfer Function, Homomorphic-Encryption, Paillier

Tương tự như bài báo ở trên, ta sẽ sử dụng Homomorphic Encryption vào việc mã hóa ảnh. Việc lưu trữ ảnh trên các dịch vụ của bên thứ ba đã không còn xa lạ nữa, vì vậy mà vấn đề bảo mật cũng được đặt lên hàng đầu. Với sự trợ giúp của Homomorphic Encryption, ta có thể yên tâm khi lưu trữ ảnh và chỉnh sửa ảnh ngay trước sự chứng kiến của bên thứ ba trong khi họ không thể biết được dữ liệu gốc của ta là gì.

Figure 3

Mã hóa ảnh thông qua Homomorphic cryptosystem để đảm bảo được bên thứ ba không thể biết được dữ liệu gốc là gì ngay cả khi họ được ủy thác quyền xử lý bức ảnh



Ta sẽ thực hiện mã hóa ảnh bằng homomorphic, cụ thể ở đây là Paillier Homomorphic Encryption. Fully homomorphic encryption cũng có thể áp dụng vào việc mã hóa ảnh, tuy nhiên khi được áp dụng vào thực tiễn thì Fully Homomorphic Encryption tỏ ra quá phức tạp và tốc độ mã hóa của nó cũng khá chậm so với Paillier mặc dù key length của FHE rất nhỏ.

3. PROPOSED SCHEME

Trước khi đi vào thuật toán, ta cần biết rằng một bức ảnh được cấu tạo từ nhiều đơn vị nhỏ gọi là pixel, mỗi pixel sẽ mang một giá trị độ sáng từ 0-255. Với những bức ảnh RGB thì 1 đơn vị pixel của nó sẽ có giá trị theo dạng (Red, Green, Blue). Ví dụ: 1 pixel màu đỏ sẽ có giá trị là (255,0,0).

Paillier Homomorphic Encryption Algorithm

Ta cũng sẽ nói về Paillier Homomorphic Encryption:

- Đầu tiên, ta chọn hai số nguyên tố rất lớn, p và q ngẫu nhiên, và đảm bảo q*p và (p-1)*(q-1) là 2 số nguyên tố cùng nhau tức là gcd(pq,(p-1)(q-1))=1.
- Tính λ: λ = lcm(p-1,q-1). Lcm là bội chung nhỏ nhất
- Ta phải chọn g ϵ Z_{n^2} sao cho đảm bảo đảm bảo rằng số n phải là ước của order của số g, tức là phương trình nghịch đảo theo modulo: μ = $\left(L\left(g^{\lambda} \bmod n^2\right)\right)^{-1} \bmod n$ phải tồn tại. Với L được định nghĩa:

$$L(x) = \frac{x-1}{n}$$

- Khóa public(encryption) là n, g
- Khóa private(decryption) là λ, μ

Mã hóa với message là m, chọn ngẫu nhiên 1 số r sao cho 0 < r < n và r ϵ Z_n (đảm bảo gcd(r,n) = 1) :

$$c = Enc(m,r) = g^m * r^n \bmod n^2$$

Giải mã với ciphertext là c:

$$m = Dec(c) = L\left(c^{\lambda} \bmod n^2\right) * \mu \bmod n$$

Trong thuật toán Paillier, với public key là n và g, việc mã hóa 2 bản rõ là m_1 và m_2 theo phương trình $Enc()$ đã đề cập ở trên, với r_1, r_2 là các số được chọn ngẫu nhiên sao cho $r_2, r_1 \epsilon Z_n$. Thì tính đồng cấu cộng tính của thuật toán Paillier sẽ được thể hiện như sau:

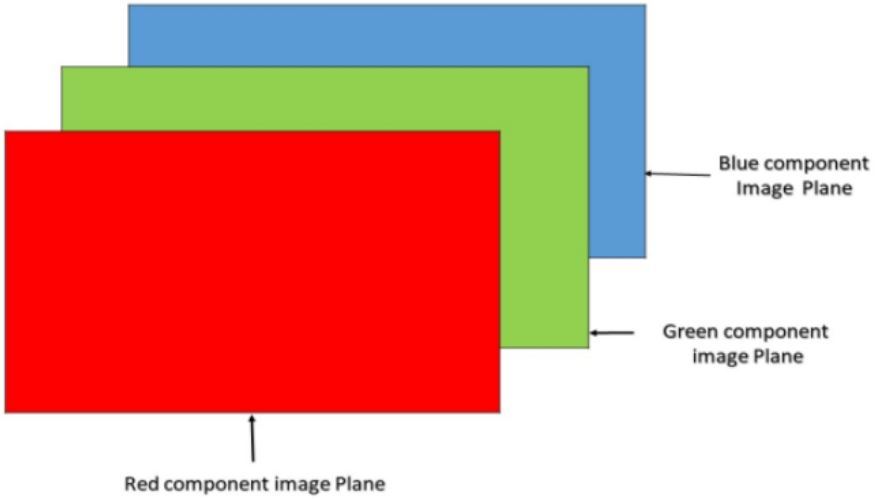
$$\begin{aligned} Enc(m_1, r_1) \,.\, Enc(m_2, r_2) &= (g^{m_1} {r_1}^n)(g^{m_2} {r_2}^n) \bmod n^2 \\ &= (g^{m_1} g^{m_2})(r_1^n r_2^n) \bmod n^2 \\ &= g^{m_1 + m_2} (r_1 r_2)^n \bmod n^2 \\ &= Enc(m_1 + m_2 \,, r_1 r_2) \end{aligned}$$

Vậy sau khi ta giải mã ra bản rõ, ta sẽ nhận được bản rõ(recovered text) bằng 2 bản rõ m_1 và m_2 ban đầu cộng lại.

PHẦN MÃ HÓA ẢNH

Để thực hiện việc mã hóa ảnh, ta sẽ mã hóa từng pixel trong ảnh bằng Paillier. Vì vậy ảnh gốc sẽ được tách ra thành 3 kênh màu riêng là đỏ (Red), xanh lá (Green) và xanh dương (Blue). Lúc này ta sẽ thu được 3 bức ảnh đơn sắc tương ứng với các kênh màu. Mỗi pixel sẽ chỉ còn mang giá trị của 1 kênh màu.

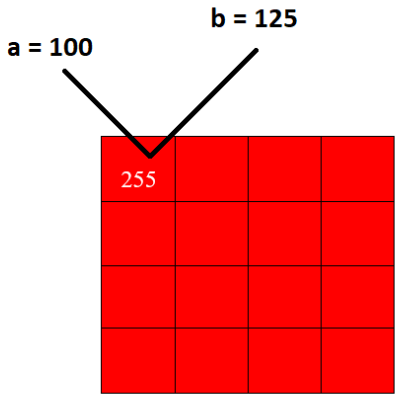
Figure 4



Ứng với mỗi kênh màu. Việc tách ảnh này sẽ được thực hiện bằng cách: từ giá trị x của pixel ban đầu, ta sẽ random ra 1 giá trị số nguyên a với điều kiện là 0 < a < x. Sau đó ta sẽ tính được giá trị số nguyên b bằng cách lấy giá trị x - a = b. Hai giá trị a và b nãy sẽ là giá trị của pixel trong 2 bức ảnh con tách ra từ ảnh gốc

Figure 5

Giá trị pixel ban đầu được tách ra làm 2 giá trị mới bằng cách random sau đó đưa vào 2 ảnh con



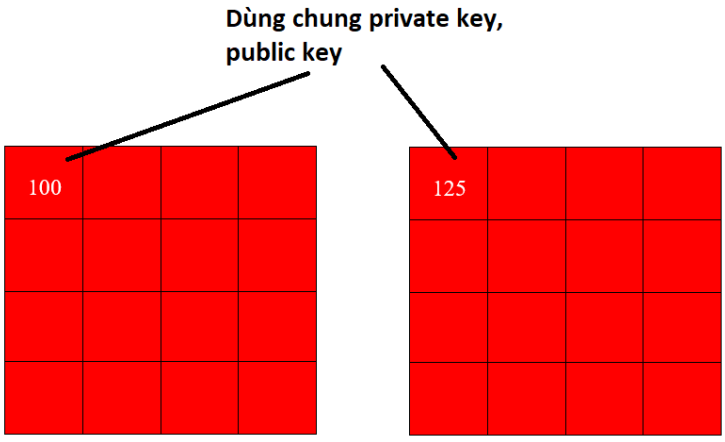
Lúc này ta sẽ thu được 6 bức ảnh con từ 3 ảnh các kênh màu ban đầu, với mỗi 2 ảnh con là do 1 ảnh gốc tách ra (Red->Red1, Red2; Blue->Blue1, Blue2; Green->Green1, Green2). Sau đó ta sẽ áp dụng Paillier Homomorphic Encryption đối với 6 bức ảnh con này.

Mã hóa các pixel bằng homomorphic encryption: loại Homomorphic Encryption mà ta dùng cho việc mã hóa này phải ánh xạ từ trường finite field Z sao cho khi ta thực hiện phép nhân trên giá trị mã hóa lại ta sẽ thu được tổng của 2 giá trị gốc. Ở đây ta sẽ chọn giải thuật Paillier.

Ta sẽ thực hiện mã hóa giá trị của các pixel trong 2 ảnh con tách ra từ ảnh gốc với cùng 1 key để sau này có thể thực hiện cộng 2 giá trị pixel của ảnh con ra giá trị của ảnh mẹ. Tuy nhiên ta chỉ dùng 1 key cho để mã hóa 6 ảnh con, mục đích là các bên thứ ba (chẳng hạn như các nhà cung cấp dịch vụ cloud) có thể thực hiện chỉnh sửa, tính toán được trên ảnh mã hóa.

Figure 6

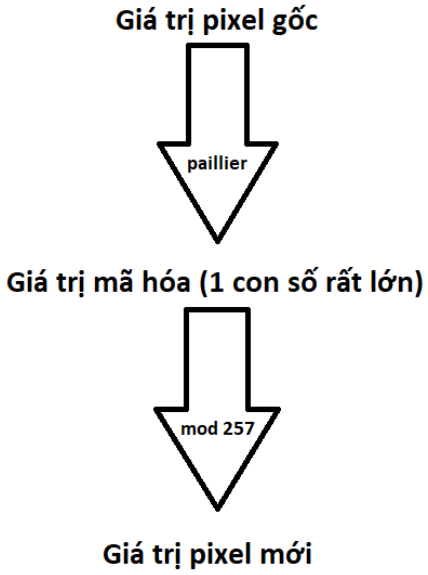
Tất cả các ảnh con dùng chung 1 key



Sau đó ta thực hiện mã hóa 2 giá trị pixel của 2 ảnh con này thành 1 giá trị mới bằng Paillier Homomorphic Encryption. Giá trị này sẽ có số bit phụ thuộc vào số bit mà người dùng nhập vào, tuy nhiên vì vấn đề bảo mật nên số bit này sẽ thường rất lớn, do đó giá trị mã hóa mà ta nhận được cũng sẽ rất lớn. Giá trị mã hóa này sẽ được mod cho 257 để giá trị mới mà ta nhận được sẽ nằm trong khoảng 0-255 vì giá trị của 1 pixel chỉ nằm trong khoảng 0-255 thôi. Ta sẽ mod cho 257 chứ không phải 255 vì thuật toán homomorphic mà ta dùng được ánh xạ từ finite field Z. Nếu có 1 pixel nào đó khi thực hiện mod mà giá trị trả về vượt quá 255 thì ta sẽ thực hiện lại quá trình mã hóa rồi mod cho 257 cho đến khi giá trị pixel mới ≤ 255. Ta sẽ lấy giá trị mới này gán lại cho các ảnh con. Việc mã hóa các ảnh còn lại cũng thực hiện tương tự. Ta cũng phải ghi lại các giá trị thương khi thực hiện các phép mod để sau này thuận tiện cho việc khôi phục giá trị mã hóa gốc để thực hiện giải mã.

Figure 7

Quá trình mã hóa 1 pixel



PHẦN GIẢI MÃ ẢNH

Sau khi thực hiện xong phần mã hóa, ta sẽ thu được 6 bức ảnh mới lần lượt là Red1, Red2, Blue1, Blue2 và Green1, Green2. Ta sẽ thực hiện giải mã lần lượt, mỗi lần 2 ảnh là các ảnh con thành phần của ảnh mẹ ban đầu.

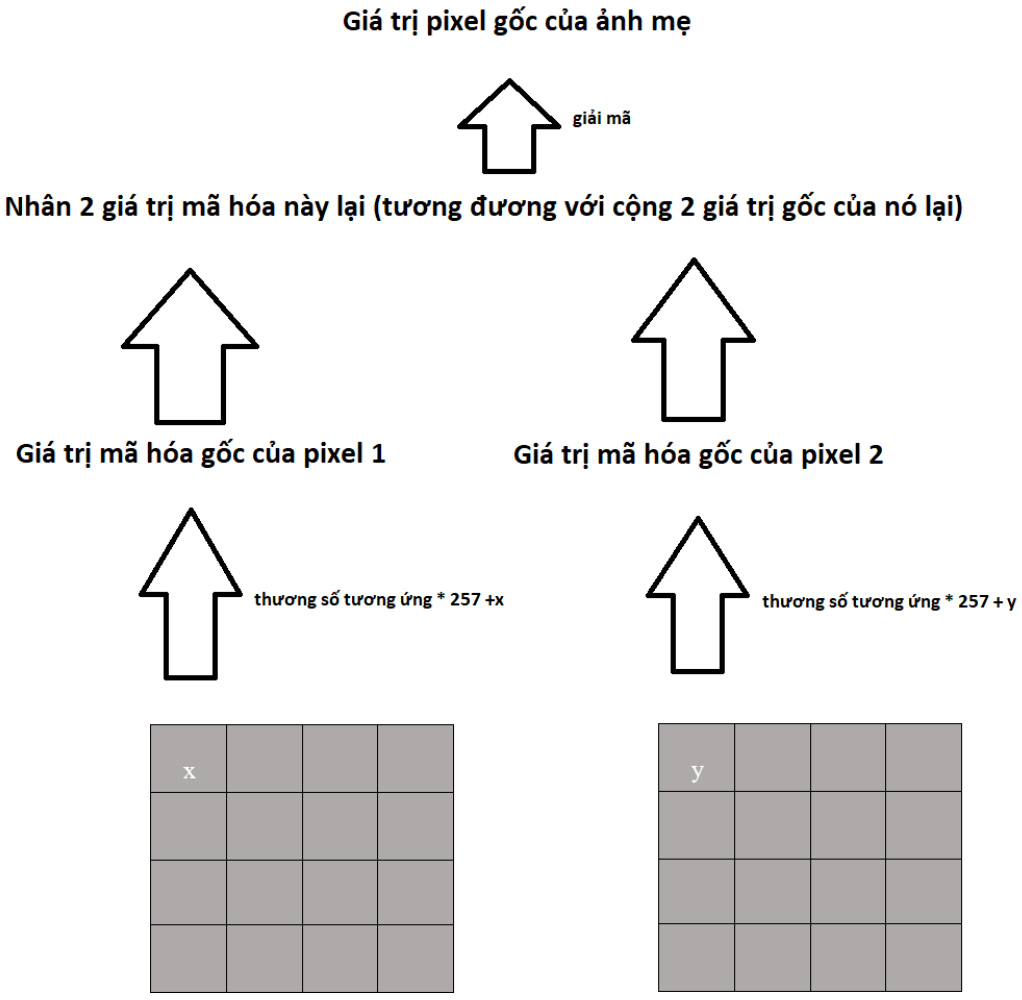
Ta sẽ khôi phục giá trị mã hóa ban đầu của các pixel bằng cách lấy các giá trị thương tương ứng của từng pixel (ta đã lưu lại ở phần mã hóa) bằng công thức như sau:

giá trị mã hóa gốc = thương số*257 + giá trị pixel hiện tại

Sau khi đã khôi phục được các giá trị mã hóa ban đầu, ta sẽ nhân 2 giá trị mã hóa này lại (thực chất là cộng 2 giá trị pixel trước khi chưa mã hóa lại) rồi sau đó giải mã, lúc này ta sẽ lấy lại được giá trị pixel gốc của bức ảnh trước khi bị tách làm 2 ảnh con.

Figure 8

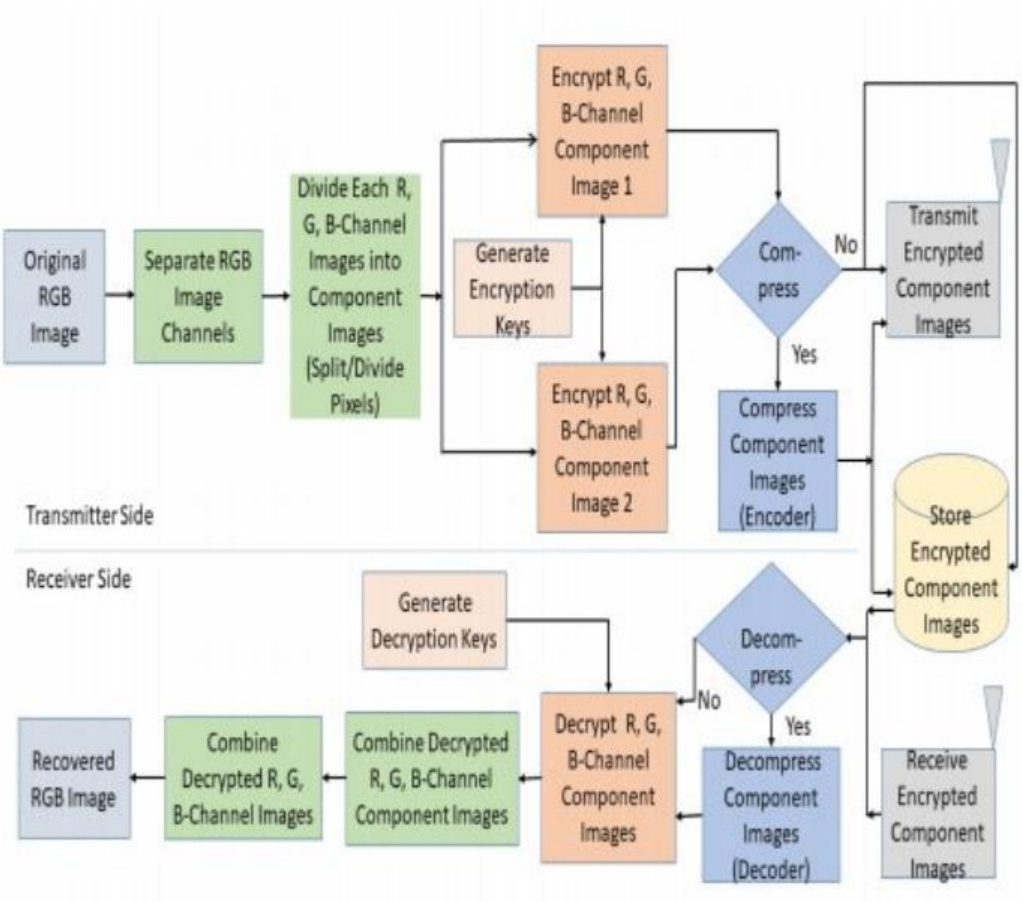
Quá trình gộp 2 bức ảnh mã hóa lại thành một thông qua khả năng tính toán trên bảng mã của Homomorphic Encryption



Sau khi thực hiện xong bước này, ta sẽ thu được 3 ảnh kênh màu gốc (Red, Blue, Green), tiến hành gộp 3 ảnh kênh màu này lại với nhau thì ta sẽ thu được ảnh gốc trước khi giải mã

Figure 9

Sơ đồ chung cho quá trình thực hiện mã hóa và giải mã ảnh bằng Homomorphic Encryption



KẾT QUẢ CHÍNH

Khi thực hiện mã hóa ảnh thông qua thuật toán này, kết quả ta thu được là các bức ảnh được mã hóa có thể chống lại các loại tấn công như Histogram analysis, Chosen plaintext attack, Brute Force Attack, Cipher Cycle, Statistical correlation attack, Entropy attacks…Và sau khi giải mã chúng ta hoàn toàn có dược bức ảnh rõ rang như ban đầu.

a. Histogram Analysis

Histogram của ảnh thể hiện thông tin về giá trị cường độ sáng của các pixel trong bức ảnh được phân tích. Với bức ảnh có cường độ sáng nằm trong khoảng từ (0,255) thì histogram được tính nhờ hàm h(l) = nl, trong đó l là giá trị cường độ sáng của pixel thứ l, và nl đại diện cho số pixel có giá trị cường độ sáng là l. Ảnh ở phía dưới là sự khác nhau

giữa 2 histogram của ảnh gốc kênh màu đỏ và histogram của 2 ảnh con được mã hóa từ ảnh này. Ảnh phía dưới cho thấy 2 cảnh con đã cung cấp 2 histogram gần như là đồng nhất về giá trị cường độ sáng của ảnh mã hóa, vì vậy nó có thể chống lại histogram analysis.

Figure 10

Ảnh gốc kênh màu đỏ (a) và histogram của nó (d), tương tự với ảnh mã hóa R1(b)(e) và ảnh mã hóa R2 (c)(f)

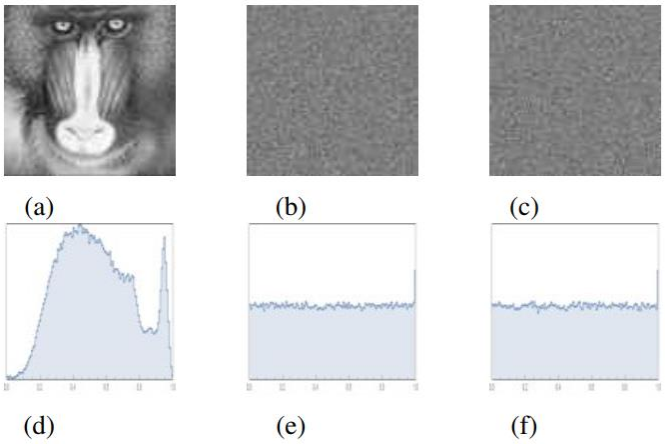


Fig. 4: (a) Original R-Channel Baboon Image R;
(b) Encrypted Baboon Component Image R1;
(c) Encrypted Baboon Component Image R2;
(d) Histogram of Original R-Channel Baboon Image R;
(e) Histogram of Encrypted Baboon Component Image R1;
(f) Histogram of Encrypted Baboon Component Image R2

b. Chosen plaintext attack

Đối với chosen plaintext attack, giả sử rằng bên attacker đã có được một phần của plaintext và ciphertext và phát hiện được thuật toán mà ta đã dùng để mã hóa ảnh. Lúc này attacker có thể bằng một cách nào đó tìm ra key mà ta đã dùng và thế là hắn có thể giải mã được bức ảnh mà ta dùng để mã hóa. Tuy nhiên mã hóa ảnh bằng Homomorphic Encryption có thể chống lại được điều này. Các bức ảnh được mã hóa luôn khác nhau khi được mã hóa cùng một key và cùng một ảnh gốc.

Figure 11

So sánh sự khác nhau về histogram của ảnh R11 và R12 khi dùng chung 1 key để mã hóa, tương tự với ảnh R21 và R22

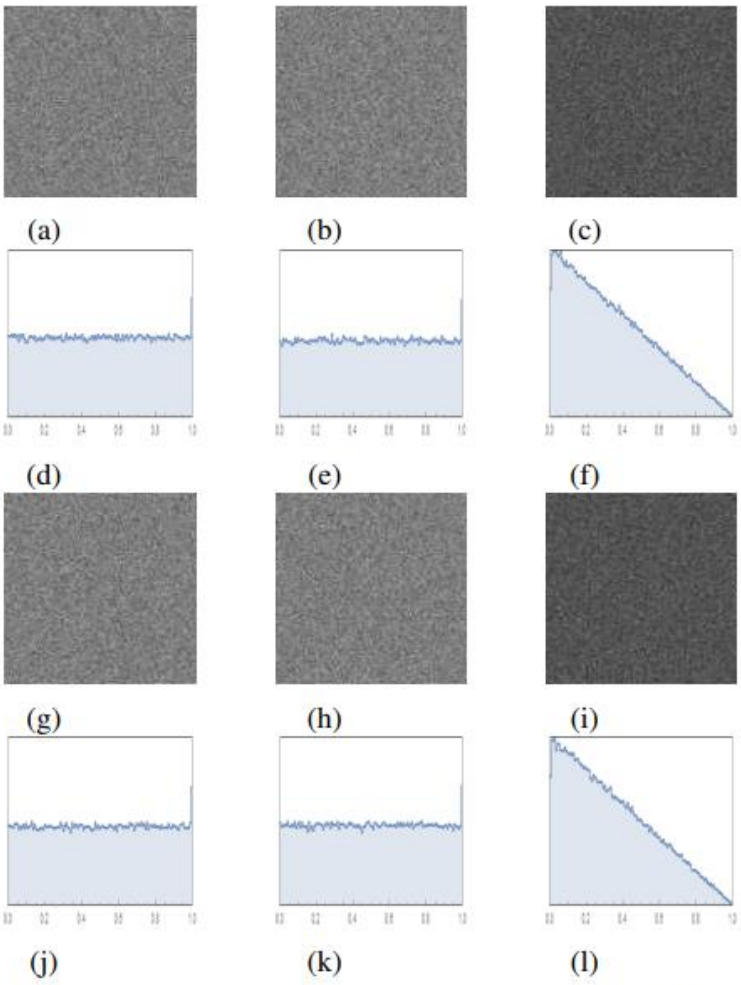


Fig. 6: (a) Encrypted Comp. Image R11 and its Hist. in (d);
(b) Encrypted Comp. Image R12 and its Histogram in (e);
(c) Pixel-wise difference $|R11 - R12|$ and its Histogram in (f);
(g) Encrypted Component Image R21 and its Histogram in (j);
(h) Encrypted Component Image R22 and its Histogram in (k);
(i) Pixel-wise difference $|R21 - R22|$ and its Histogram in (l).

c. Cipher Cycle

Một yêu cầu quan trọng của những thuật toán mã hóa ảnh là nó phải tạo được bức ảnh mã hóa khác hoàn toàn so với ảnh gốc. Để kiểm tra điều này thì ta sẽ nhờ vào Number

of Pixel Change Rate (NPCR) và Unified Average Change Intensity (UACI). NPCR và UCAI thường được dùng như là một bài test về bảo mật bằng cách thực hiện chỉnh sửa lên ảnh đã được mã hóa và quan sát sự thay đổi trên kết quả.

NCPR

Thực hiện đo lường giá trị trung bình của số các pixel khác nhau giữa ảnh mã hóa trước khi thực hiện thay đổi và sau khi thực hiện thay đổi. Công thức của NPCR như sau:

$$NPCR_{R,G,B} = \frac{\sum_{i,j} D_{R,G,B}(i,j)}{N} \times 100\%$$

Với N là tổng số pixel của ảnh và DR,G,B được tính như sau:

$$D_{R,G,B}(i,j) \triangleq \begin{cases} 0, & if \quad C_{R,G,B}(i,j) = C'_{R,G,B}(i,j) \\ 1, & if \quad C_{R,G,B}(i,j) \neq C'_{R,G,B}(i,j) \end{cases}$$

Trong đó CR,G,B(i,j) và C'R,G,B(i,j) thể hiện giá trị tương ứng của pixel ở ảnh C và C'. Khi thực hiện trên 2 bức ảnh thì giá trị để thuật toán mã hóa được đánh giá là an toàn được tính bằng công thức sau:

$$\mathbb{E}(NPCR) = (1 - 2^{-L_{R,G,B}}) \times 100\%$$

Với 2-LR,G,B là số bit dùng để biểu thị giá trị của pixel tạo nên các màu R,G,B. Ví dụ như ta có 1 bức ảnh với size là 512x512 và có 24bit với 8bit cho mỗi kênh màu R,G,B (LR = LG = LB = 8 bits) thì giá trị ta tính được khi dùng NPCR phải là

$$NPCR_R = NPCR_G = NPCR_B = 99.609375\%$$

thì thuật toán mới có thể gọi là an toàn.

UACI

Được tính bằng công thức như sau:

$$UACI_{R,G,B} = \frac{1}{N}\left[\sum_{i,j}\frac{\left|C_{R,G,B}(i,j) - C'_{R,G,B}(i,j)\right|}{2^{L_{R,G,B}}-1}\right] \times 100\%$$

Trong đó LR,G,B là số bit dùng để hiện thị các màu R,G,B cho bức ảnh. Khi ta dùng 2 bức ảnh tùy ý, giá trị mong muốn để thuật toán mã hóa được là an toàn là:

$$\mathbb{E}(UACI_{R,G,B}) = \frac{\frac{1}{2^{2L_{R,G,B}}-1}\left(\sum_{i=1}^{2^{L_{R,G,B}}-1} i(i+1)\right)}{2^{L_{R,G,B}}-1} \times 100\%$$

Với thuật toán mã hóa mà ta đang dùng sử dụng ảnh RGB, tức là ảnh sẽ dùng 24bits để biểu diễn các kênh màu, tức là mỗi kênh màu dùng 8bits để biểu diễn, giá trị mà ta mong muốn là:

$$\mathbb{E}(UACI_R) = \mathbb{E}(UACI_G) = \mathbb{E}(UACI_B) = 33.46354\%$$

NPCR và UACI sẽ phân tích và so sánh giữa ảnh gốc và ảnh được mã hóa để xem liệu ảnh mã hóa và ảnh gốc có đạt được giá trị mong muốn hay không, và đây là kết quả khi họ thực hiện trên một bức ảnh có kênh màu xanh (Blue). Các giá trị ta tính ra được đều rất gần với Expected Value, chứng tỏ thuật toán này an toàn.

Figure 12

Bảng so sánh giá trị theo NPCR và UACI giữa ảnh gốc và ảnh mã hóa

B-Channel			
Test Type	B and B1	B and B2	Expected Value
NPCR (%)	99.6136	99.6273	99.60937
UACI (%)	31.3253	31.3231	33.4635

4. CONTRIBUTIONS

- Chúng em đã tìm hiểu về Homomorphic Encryption và ứng dụng của nó cho các lĩnh vực trong cuộc sống.
- Chúng em giải thích được lý do lựa chọn Paillier để áp dụng vào việc mã hóa ảnh và thuật toán của nó.
- Dựa vào thuật toán được nêu trong bài báo, chúng em đã tự code được chương trình mô phỏng lại thuật toán mã hóa ảnh được đề cập trong bài báo khoa học, kiểm tra xem hiệu năng của nó như thế nào khi áp dụng thực tế.
- Chúng em cũng đã áp dụng tính năng chạy các tiến trình bất đồng bộ để tăng tốc độ mã hóa ảnh

5. DEMO

Cấu hình của máy dùng để demo trên chương trình do bọn em tự code mô phỏng lại thuật toán

Figure 13

Current Date/Time:	Tuesday, June 29, 2021, 8:13:46 PM
Computer Name:	DESKTOP-DAMBMDG
Operating System:	Windows 10 Home Single Language 64-bit (10.0, Build 19043)
Language:	English (Regional Setting: English)
System Manufacturer:	HP
System Model:	HP ProBook 445 G7
BIOS:	S79 Ver. 01.05.00
Processor:	AMD Ryzen 7 4700U with Radeon Graphics (8 CPUs), ~2.0GHz
Memory:	8192MB RAM
Page file:	7940MB used, 7832MB available
DirectX Version:	DirectX 12

- Ảnh gốc được sử dụng để mã hóa với độ phân giải là 500x500
- Thực hiện mã hóa ảnh có độ phân giải là 500x500 bằng key có độ dài là 2048bits

Figure 14

Ảnh gốc ban đầu

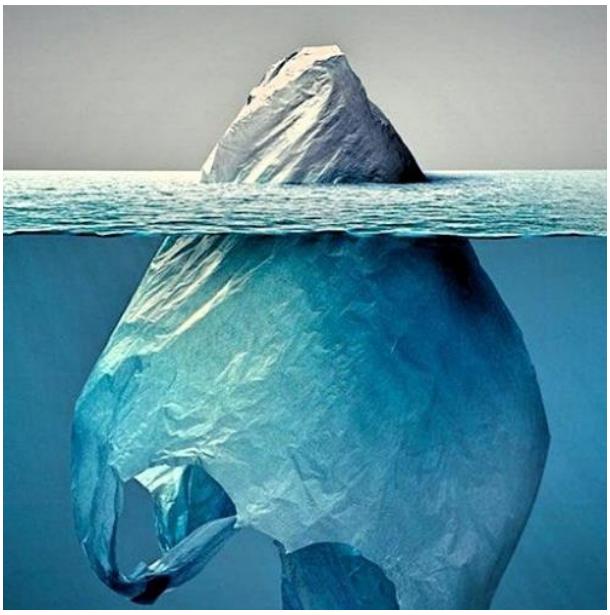
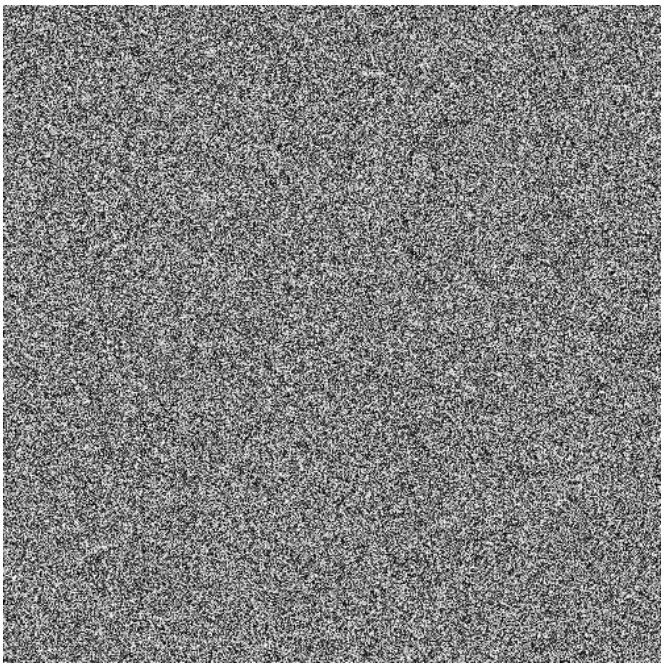
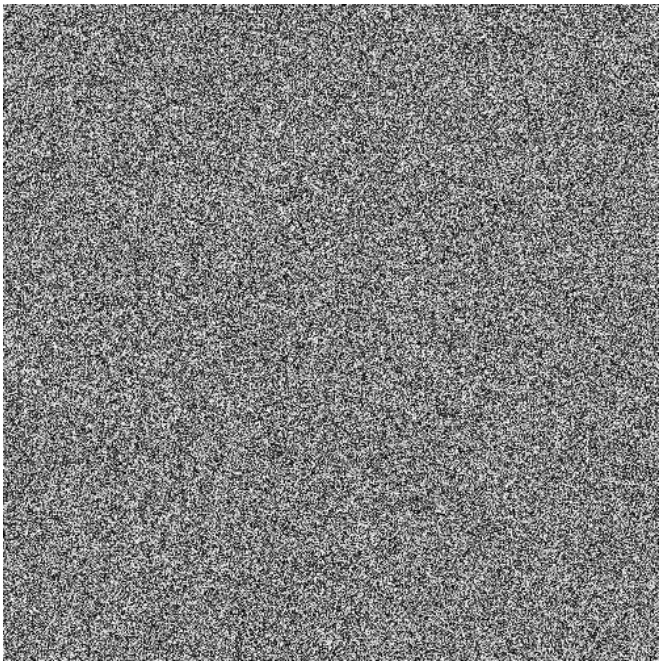
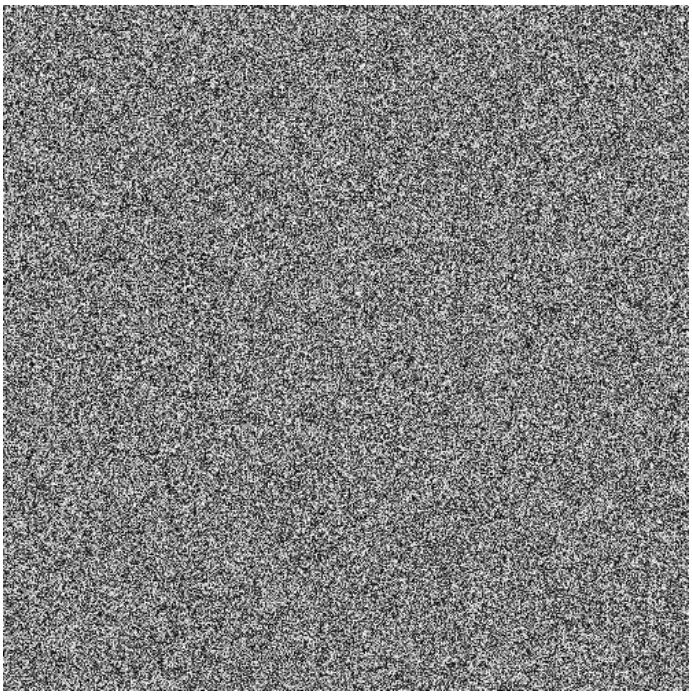


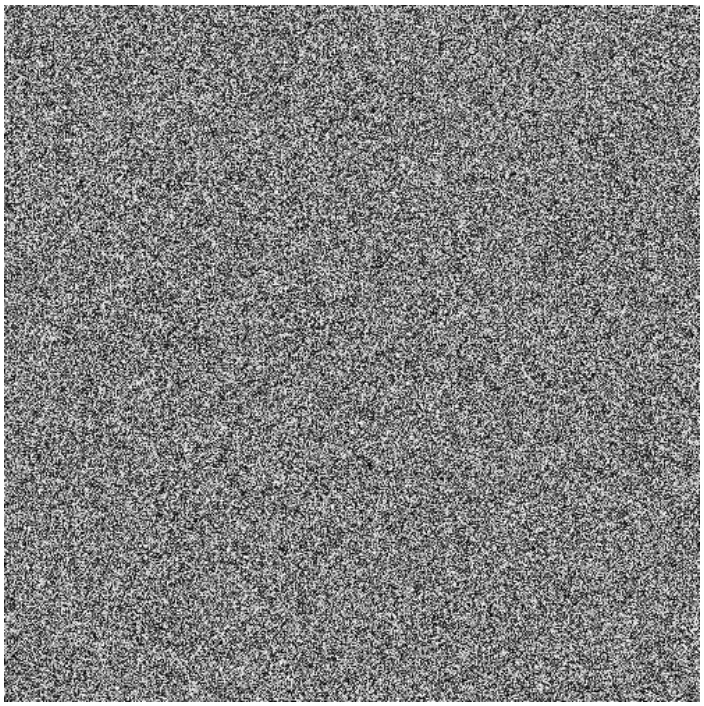
Figure 15->20

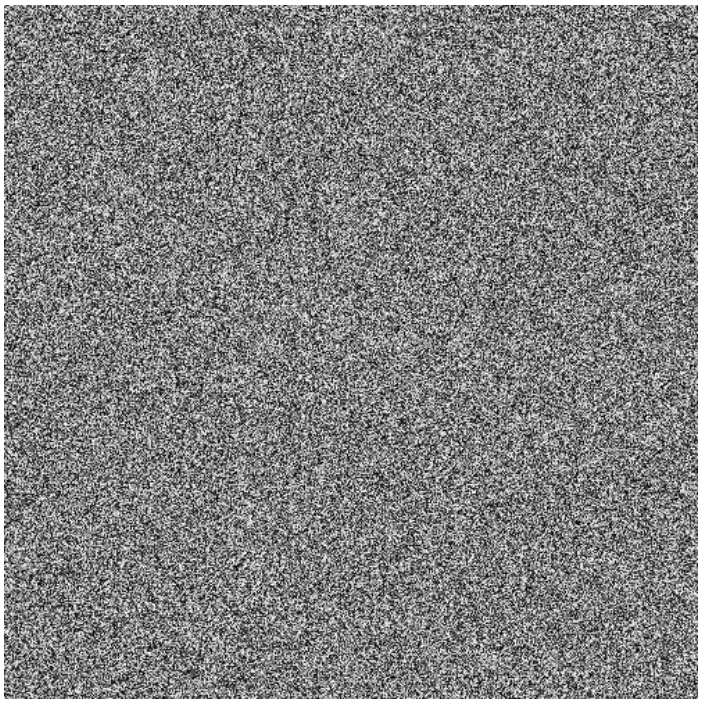
Các bức ảnh được mã hóa (từ 1 ảnh gốc -> 6 ảnh con):

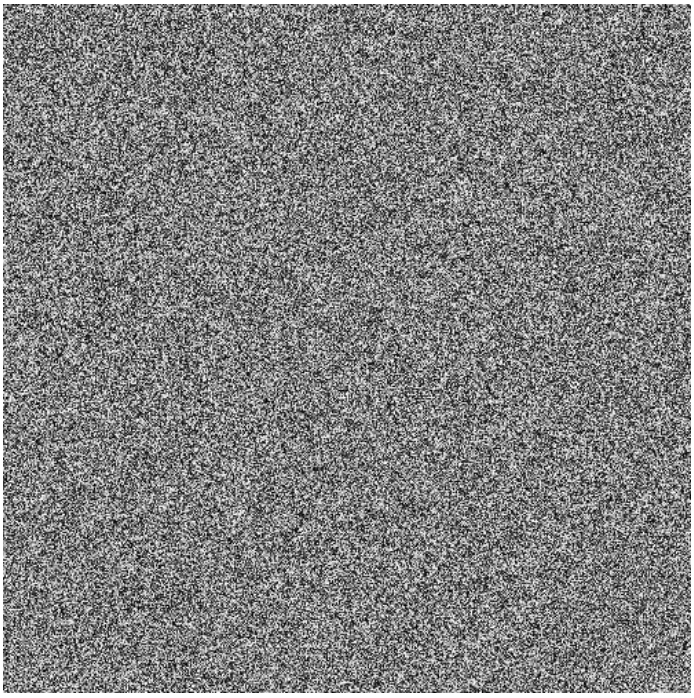












Thời gian mã hóa và giải mã:

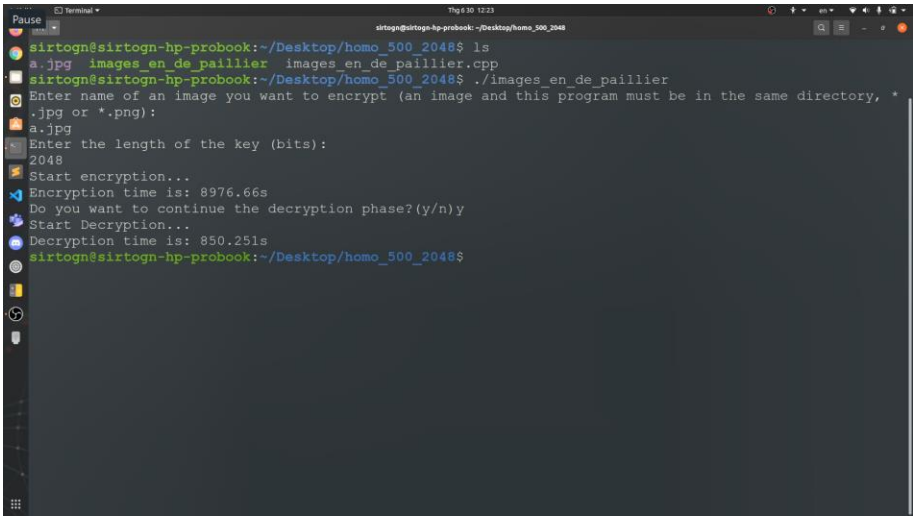
- Do khi thực hiện mã hóa, bọn em xem mỗi ảnh là 1 mảng 2 chiều, sau đó lặp qua mảng để mã hóa từng pixel nên dẫn tới việc mã hóa diễn ra rất lâu

- Mã hóa: 8976.66s ≈ 150 min

- Giải mã: 850.251s ≈ 14 min

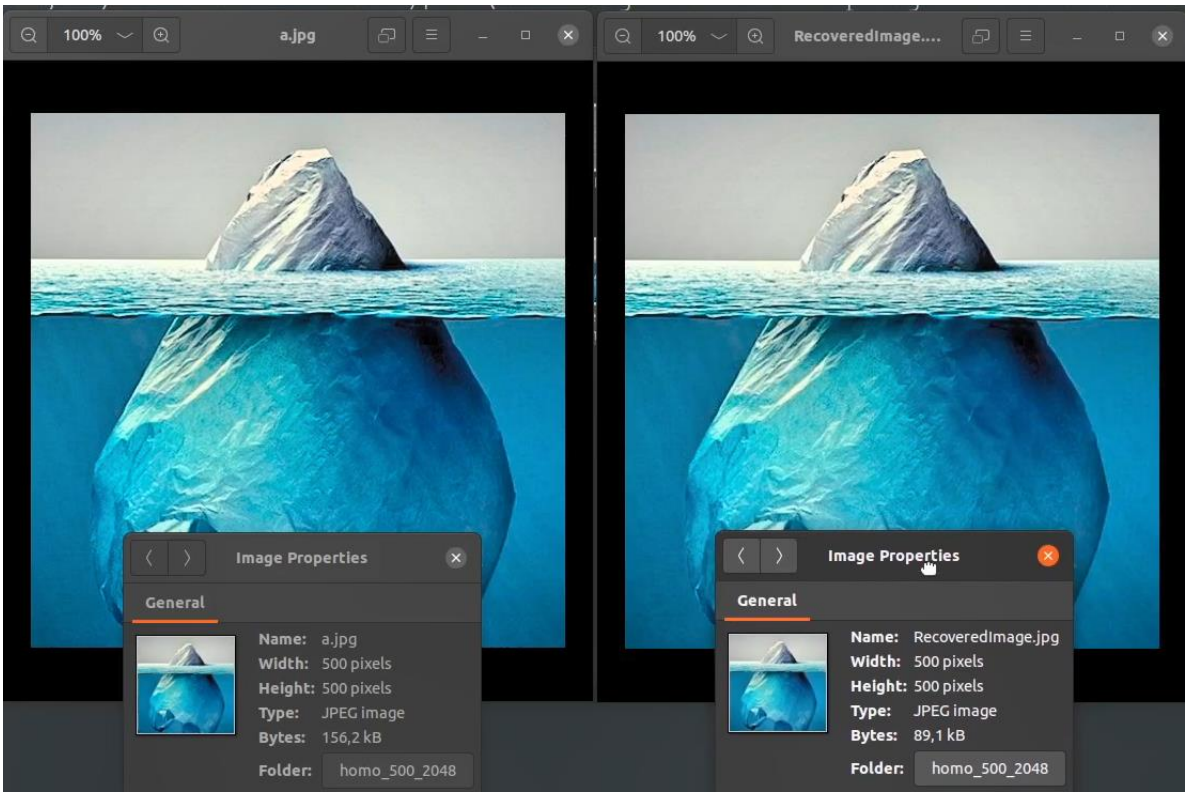
Figure 21

Thực hiện demo trên hệ điều hành Ubuntu



So sánh giữa ảnh gốc và ảnh được giải mã

Figure 22 *So sánh ảnh gốc và ảnh được giải mã*



Note: Ảnh a.jpg là ảnh ban đầu, RecoveredImage.jpg là ảnh sau khi khôi phục sau khi giải mã. Đoạn code do bọn em code mô phỏng theo bài báo này được đính kèm trong file nén.

6. REFERENCES

1. **Wade, M. I., Chouikha, M., Gill, T., Patterson, W., Washington, T. M., & Zeng, J. (2019, 10-12 Oct. 2019). Distributed Image Encryption Based On a Homomorphic Cryptographic Approach. 2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON).**

2. Mazza, S., Patel, D., & Viola, I. (2021). Homomorphic-Encrypted Volume Rendering. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 635-644. https://doi.org/10.1109/TVCG.2020.3030436

3. Qin, Z., Weng, J., Cui, Y., & Ren, K. (2018). Privacy-Preserving Image Processing in the Cloud. *IEEE Cloud Computing*, *5*(2), 48-57. https://doi.org/10.1109/MCC.2018.022171667

4. Fu, W., Lin, R., & Inge, D. (2018). Fully Homomorphic Image Processing. *CoRR*, *abs/1810.03249*. http://arxiv.org/abs/1810.03249

5. Shah, M., Zhang, W., Hu, H., & Yu, N. (2019). Paillier Cryptosystem based Mean Value Computation for Encrypted Domain Image Processing Operations. *ACM Trans. Multimedia Comput. Commun. Appl.*, *15*(3), Article 76. https://doi.org/10.1145/3325194