Báo cáo bài thực hành: tìm hiểu về thuật toán ECDSA

Mục lục

[1.1. Nội dung và hướng dẫn bìa thực hành 1](#_Toc182407830)

[1.1.1. Mục đích 1](#_Toc182407831)

[1.1.2. Yêu cầu đối với sinh viên 1](#_Toc182407832)

[1.1.3. Nội dung thực hành 1](#_Toc182407833)

[1.1.4. Phân tích yêu cầu bài thực hành 3](#_Toc182407834)

[1.1.5. Thiết kế bài thực hành 3](#_Toc182407835)

[1.1.6. Cài đặt và cấu hình các máy ảo 4](#_Toc182407836)

[1.1.8. Thử nghiệm và đánh giá 7](#_Toc182407837)

1. Bài thực hành: Tìm hiểu về thuật toán ECDSA
   1. Nội dung và hướng dẫn bài thực hành
      1. Mục đích

Giúp sinh viên tìm hiểu về thuật toán ECDSA, lập trình cơ bản thuật toán và sử dụng công cụ GnuPG để thực hiện tạo và xác thực chữ ký số.

* + 1. Yêu cầu đối với sinh viên

Có kiến thức cơ bản về lập trình và mật mã.

Có kiến thức cơ bản về thuật toán ECDSA

* Tạo public key:
  + Giả sử ta đã có một private key là một số ngẫu nhiên *dA*​.
  + Trên đường cong Elliptic ta chọn một điểm **G**, gọi là điểm sinh (generator point hay reference point).
  + Public key *QA*​ được sinh ra bằng kết quả của phép nhân:

*QA* ​= *dA* ​ \* *G*

* + Mối quan hệ giữa *dA*​ và *QA*​ là cố định, và chỉ tính được theo một chiều từ *dA*​ đến *QA*​
* Tạo chữ ký:
  + Để tạo ra cặp (r, s) này đầu tiên ra sẽ phải chọn ra một số ngẫu nhiên k (lưu ý đây là một số ngẫu nhiên khác với private key).
  + Sau đó nhân k với điểm sinh G giống như phép toán ta dùng để tạo ra pubic key bên trên:

P = k \* G

khi này ta có được một điểm P(x, y), toạ độ x của P chính là giá trị r.

* + Để tính được s, đầu tiên ta tính hash của message mà ta cần ký, gọi là z (trong Ethereum thì hàm hash là Keccak256), khi này:

s=k−1(z+dA\*r)(mod p)

Ta lưu ý ở đây k-1 là **nghịch đảo theo modulo** của k​.

* Xác thực chữ ký:
  + Để xác minh tính hợp lệ của chữ ký, ta chỉ cần public key *QA*​ là đủ. Bằng cách tính:

*P*=*s*−1 \* *z* \* *G* + *s*−1 \* *r* \* *QA*​

*Nếu toạ độ x của P bằng r có nghĩa là chữ ký hợp lệ*

* + 1. Nội dung thực hành

Khởi động bài lab:

Vào terminal gõ: labtainer –r ecdsa

*(chú ý: sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)*

Sau khi khởi động xong 2 terminal sẽ xuất hiện là user1 và user 2 với password là ubuntu. Bài lab đã chuẩn bị 2 file chứa code đơn giản của thuật toán ECDSA, sinh viên sẽ chỉnh sửa mã nguồn để tạo chữ ký số cho thông điệp nhập từ bàn phím và lưu lại khoá và file chữ ký số sau đó gửi sang user2 để xác thực với dữ liệu nhập từ bàn phím tiếp đó để hoàn thành task.

Tiếp theo nhiệm vụ của sinh viên là sử dụng công cụ GnuPG để tạo và xác thực chữ ký số của filetoview.txt.

Sinh viên thực hiện tạo cặp khoá bằng lệnh:

*sudo gpg --full-generate-key --expert*

Sau đó chọn *(9) ECC and ECC* tiếp theo chọn đường cong và tiếp tục theo hướng dẫn. Ta thu được cặp khoá công khai và bí mật và lưu chúng vào hệ thống.

Sử dụng khoá bí mật để tạo chữ ký số cho file bằng lệnh:

*sudo gpg --detach-sign -a <file\_name>*

Sau khi thu được file chữ ký số của file cần kiếm tra, sinh viên thực hiện xuất khoá công khai

sudo gpg --export --armor "<Your Key ID or Email>" > public\_key.asc

Gửi file khoá công khai cùng với file chữ ký số và file xác thực sang container user2

Sinh viên thực hiện nhập khoá vào máy khác:

gpg --import public\_key.asc

Sinh viện thực hiện xác thực file:

gpg --verify <file chữ ký số> <file cần kiểm tra>

Công cụ sẽ đưa ra Good signature khi xác thực thành công và BAD signature khi xác thực thất bại

Sinh viên xác thực thành công coi như hoàn thành task

Sau đó thử lại với trường hợp file bị sửa đổi trong quá trình truyền

Kết thúc bài lab:

stoplab ecdsa

Khởi động lại bài lab:

labtainer -r ecdsa

* + 1. Phân tích yêu cầu bài thực hành

Bài thực hành gồm 1 máy tính để thực hiện bài thực hành. Để hoàn thành bài thực hành sinh viên cần chạy code và sử dụng Gnupg để tạo và xác thực chữ ký số.

* + 1. Thiết kế bài thực hành

Trên môi trường máy ảo Ubuntu được cung cấp, sử dụng docker tạo ra 2 container mang tên user1 và user2

Tạo mạng LAN có cấu hình: 192.168.10.1 và gateway: 192.168.10.0/24

Cấu hình docker cần có:

User1: lưu cấu hình cho máy thực hành

Tên máy: user1

Địa chỉ trong mạng LAN: 192.168.10.3

Gateway: 192.168.10.1

User2: lưu cấu hình cho máy thực hành

Tên máy: user2

Địa chỉ trong mạng LAN: 192.168.10.4

Gateway: 192.168.10.1

config: lưu cấu hình hoạt động của hệ thống

dockerfiles: mô tả cấu hình của container ecdsa

user1: sử dụng các công cụ và file có sẵn giúp thực hiện các lệnh và chức năng của tool này

user2: sử dụng các công cụ và file có sẵn giúp thực hiện các lệnh và chức năng của tool này

docs: lưu phần mô tả hướng dẫn làm bài thực hành cho sinh viên.

Các nhiệm vụ cần phải thực hiện để thực hành thành công

Chỉnh sửa code có sẵn để tạo chữ ký số của thông điệp ở máy user1, màn hình hiện chữ “signed” vì bài lab đã được cài đặt chấm điểm dựa vào vết này.

Chỉnh sửa code có sẵn để kiểm tra chữ ký số của thông điệp ở máy user2, màn hình hiện chữ “valid” vì bài lab đã được cài đặt chấm điểm dựa vào vết này.

Sử dụng Gnupg để tạo và xác thực chữ ký số của file có sẵn ở máy user2, màn hình hiện chữ “Good” vì bài lab đã được cài đặt chấm điểm dựa vào vết này

Sử dụng Gnupg để tạo và xác thực chữ ký số của file có sẵn ở máy user2, màn hình hiện chữ “Bad” vì bài lab đã được cài đặt chấm điểm dựa vào vết này

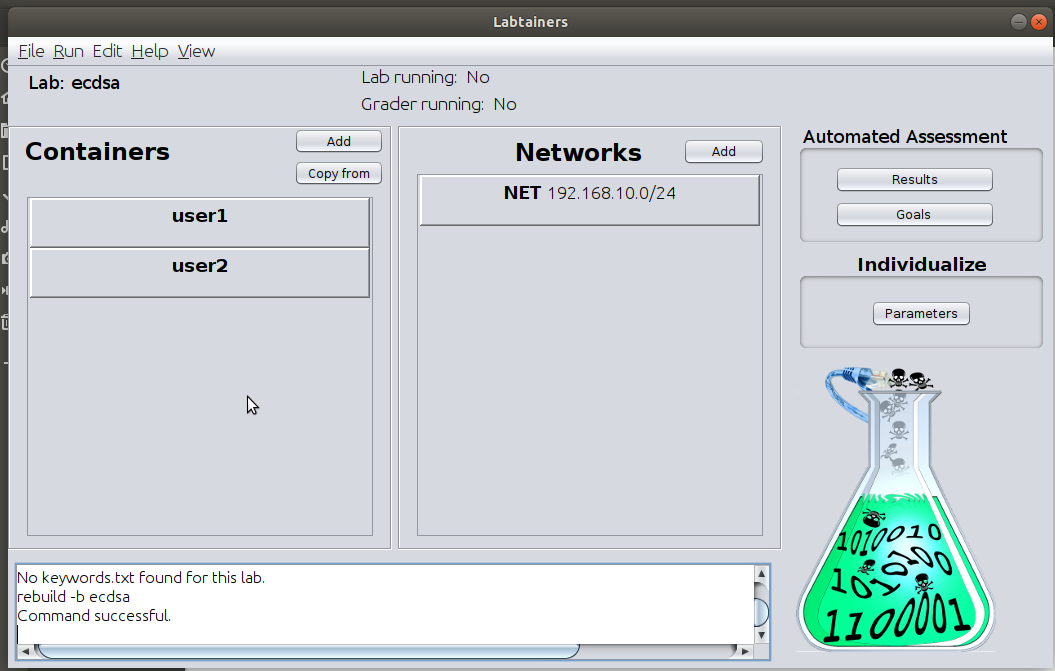
Kết thúc bài lab và đóng gói kết quả

Để đánh giá được sinh viên đã hoàn thành bài thực hành hay chưa, cần chia bài thực hành thành các nhiệm vụ nhỏ, mỗi nhiệm vụ cần phải chỉ rõ kết quả để có thể dựa vào đó đánh giá, chấm điểm. Do vậy, trong bài thực hành này hệ thống cần ghi nhận các thao tác, sự kiện được mô tả và cấu hình như bảng 1

Bảng 1. Bảng Result

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Result Tag | Container | File | Field Type | Field ID | Timestamp Type |
| create\_sig | user1 | create\_signature.py.stdout | CONTAINS | signed | File |
| verify\_sig | user2 | verify\_signature.py.stdout | CONTAINS | valid | File |
| gpg\_success | user2 | gpg.stdout | CONTAINS | Good | File |
| gpg\_failed | user2 | gpg.stdout | CONTAINS | BAD | File |

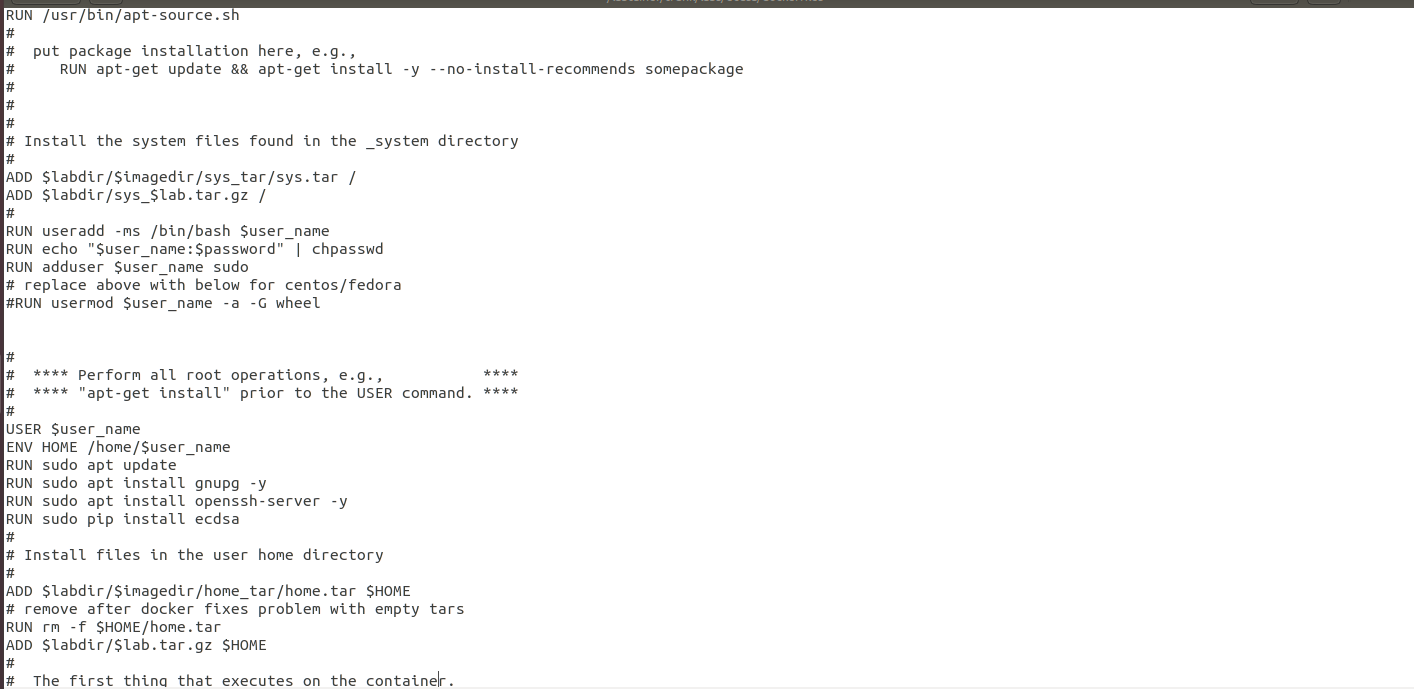
* create\_sig: hiển thị thông báo tạo chữ kí số thành công
* verify\_sig: hiển thị xác thực thông điệp thành công
* gpg\_success: hiển thị xác thực thành công
* gpg\_failed: hiển thị xác thực thất bại
  + 1. Cài đặt và cấu hình các máy ảo



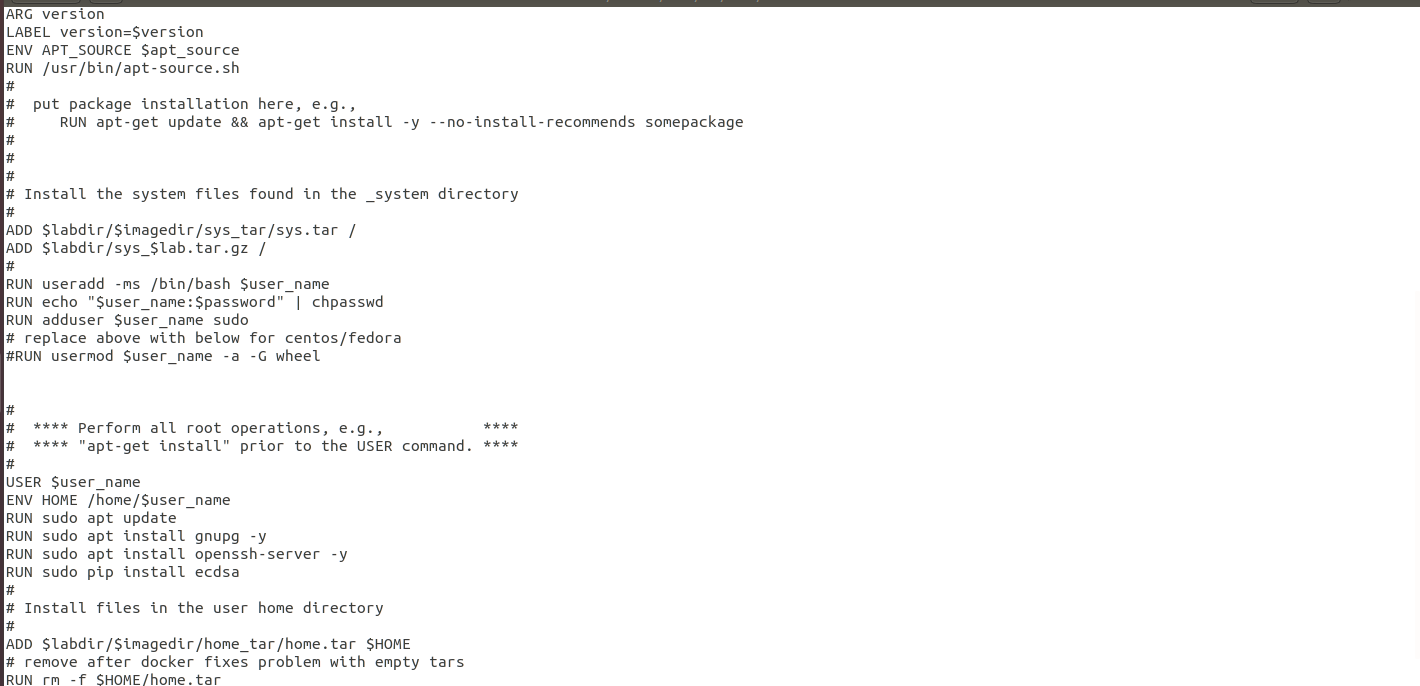
Hình 1. Giao diện Labedit của bài lab



Hình 2. Cài đặt phần Result



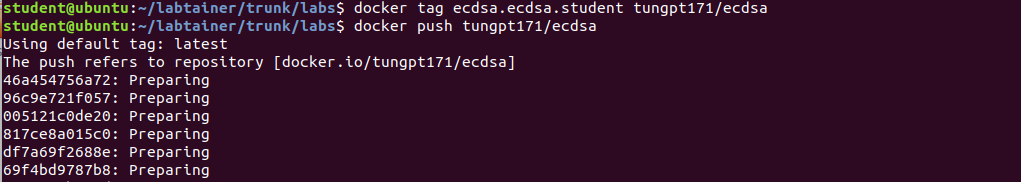
Hình 3. Dockerfile của máy user1



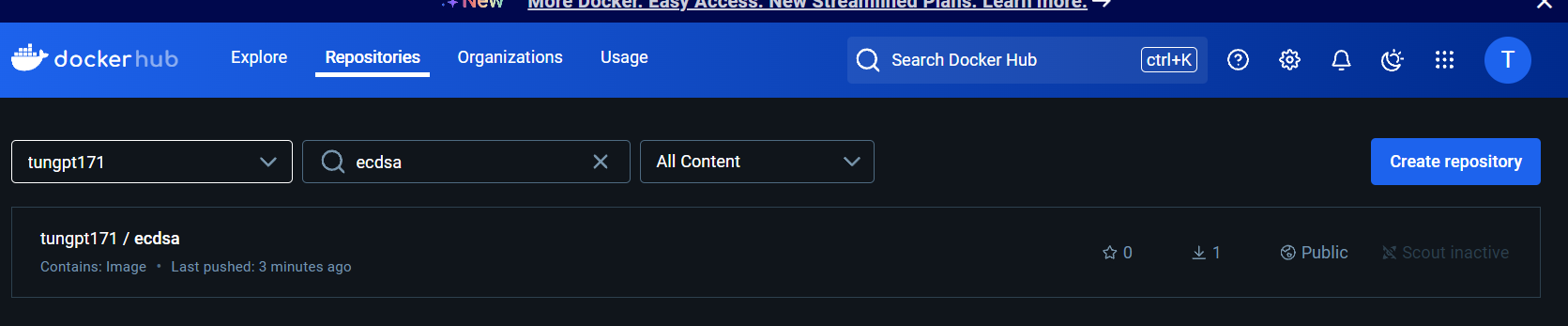
Hình 4.Dockfile của máy user2

* + 1. Tích hợp và triển khai
       1. Dockerhub

<https://hub.docker.com/u/tungpt171>



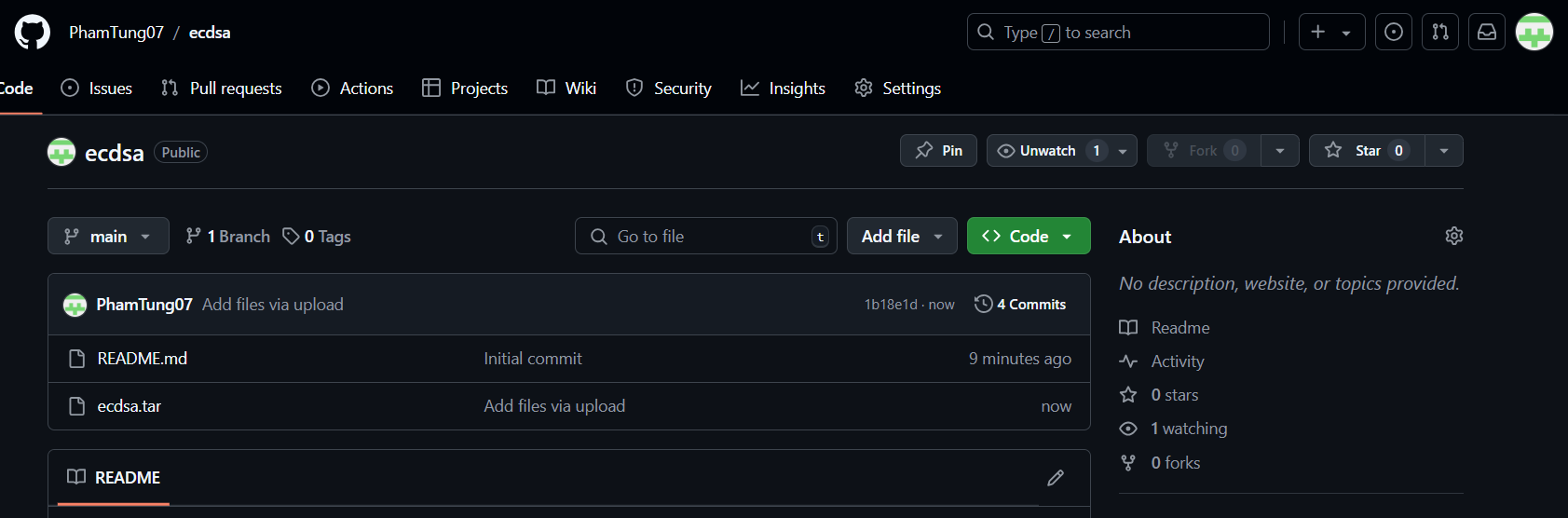
Hình 5.Đẩy các container lên Dockerhub



Hình 6. Các container được đẩy lên dockerhub

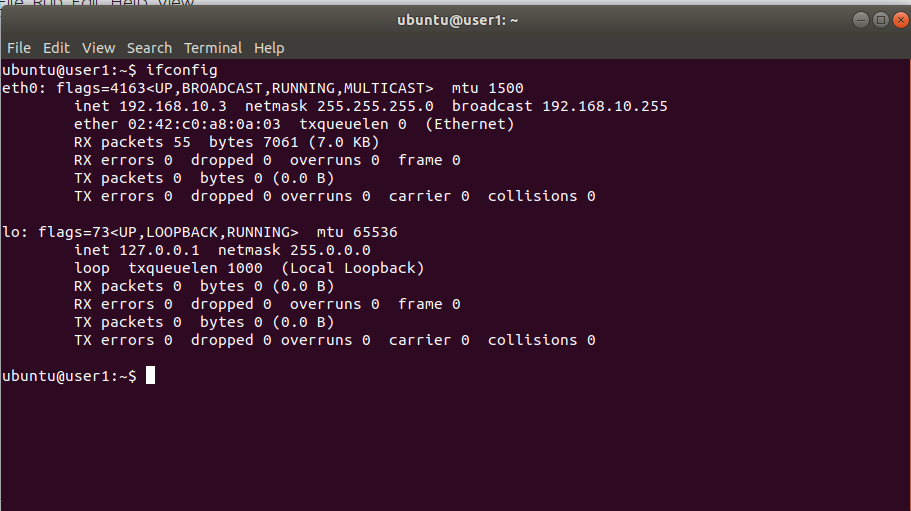
* + - 1. Github

<https://github.com/PhamTung07/ecdsa>

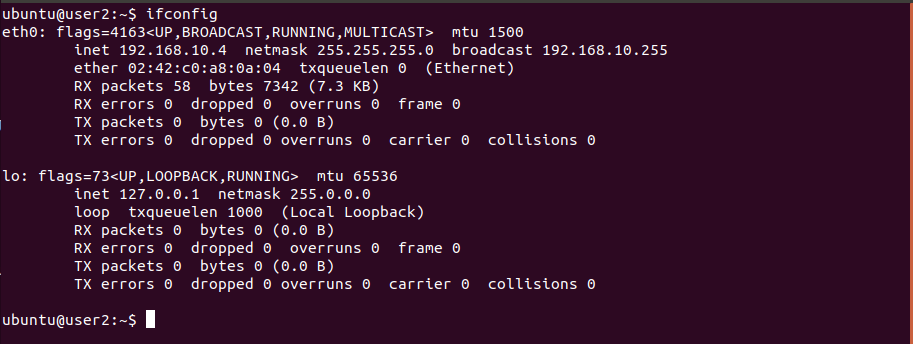


Hình 7.File ecdsa.tar chứa bài thực hành

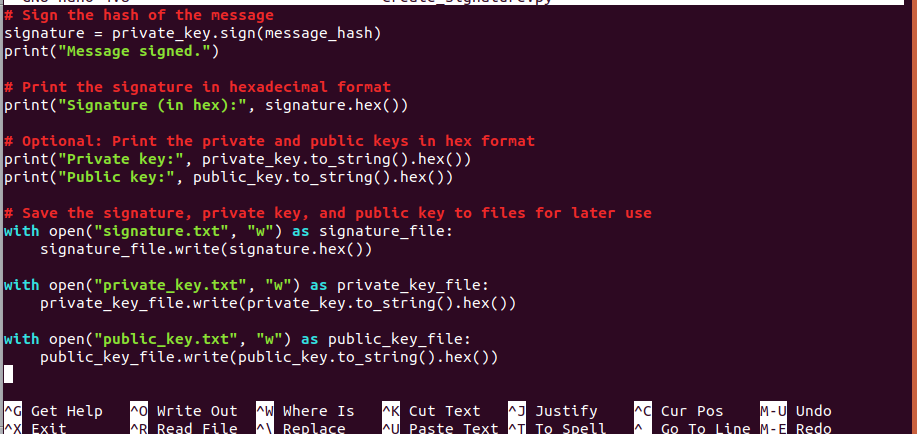
* + 1. Thử nghiệm và đánh giá



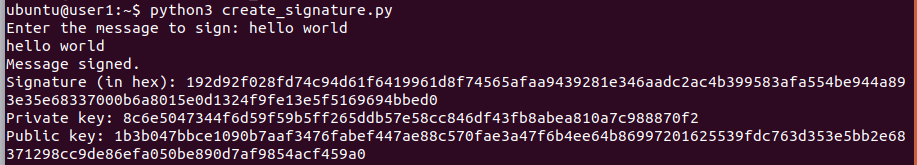
Hình 8. IP của máy user1



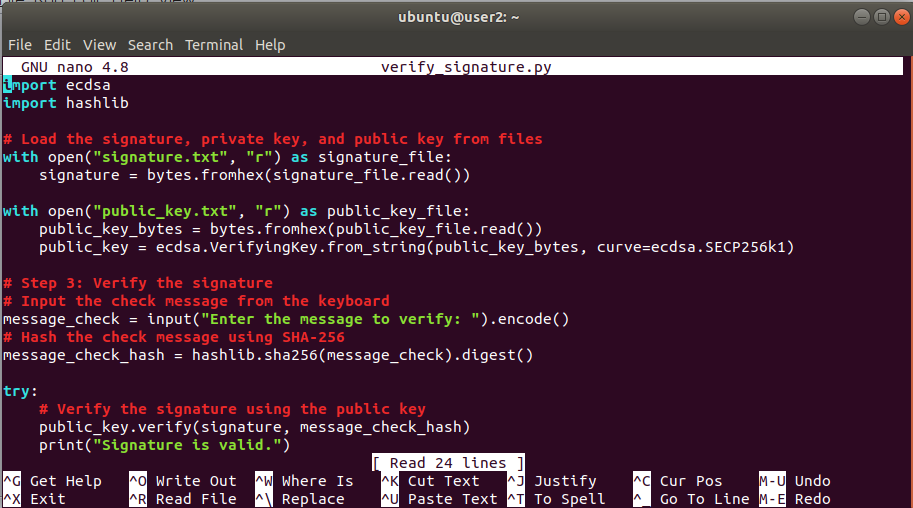
Hình 9.IP của máy user2



Hình 10. Chỉnh sửa file python create\_signature.py



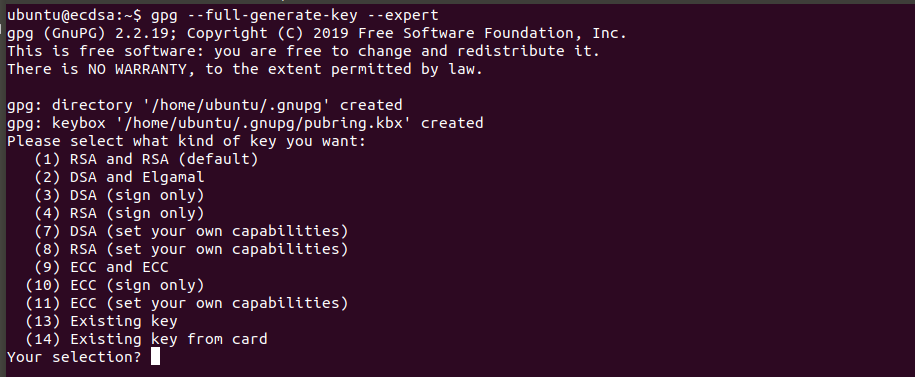
Hình 11.Tạo chữ ký số cho thông điệp



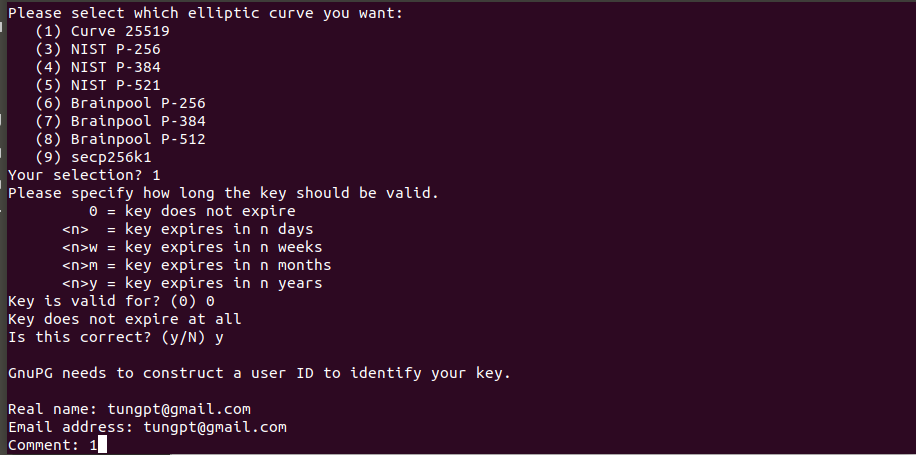
Hình 12.Chỉnh sửa file verify\_signature.py



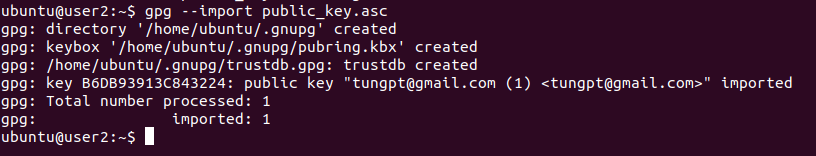
Hình 13.Kiểm tra chữ ký số của thông điệp



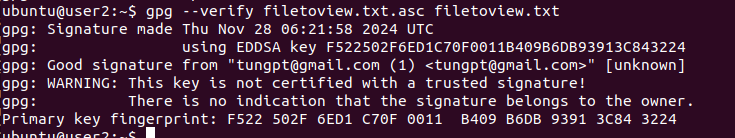
Hình 14. Sử dụng Gnupg để tạo cặp khoá ECC



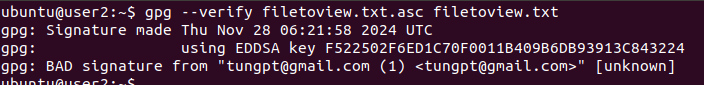
Hình 15. Tạo cặp khoá ECC



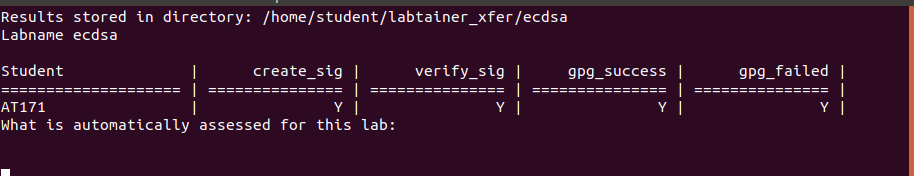
Hình 16.Import khoá công khai vào máy user2



Hình 17.Xác thực thành công



Hình 18. Xác thực thất bại



Hình 19. Đánh giá kết quả bài thực hành