TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**MÔN DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

**Xây dựng hệ thống mã hóa dòng**

*Giảng viên hướng dẫn:* **Thầy NGUYỄN QUỐC BÌNH**

*Người thực hiện*: **TRẦN BẢO DUY – 51603080**

**NGUYỄN THỊ NHẬT LỆ – 51603178**

Lớp: **16050303**

**16050304**

Khoá: **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

TỔNG LIÊN ĐOÀN LAO ĐỘNG VIỆT NAM

**TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO**

**MÔN DỰ ÁN CÔNG NGHỆ THÔNG TIN 2**

**Xây dựng hệ thống mã hóa dòng**

*Giảng viên hướng dẫn:* **Thầy NGUYỄN QUỐC BÌNH**

*Người thực hiện*: **TRẦN BẢO DUY – 51603080**

**NGUYỄN THỊ NHẬT LỆ – 51603178**

Lớp: **16050303**

**16050304**

Khoá: **20**

**THÀNH PHỐ HỒ CHÍ MINH, NĂM 2019**

LỜI CẢM ƠN

Chúng em xin chân thành cảm ơn Thầy Nguyễn Quốc Bình – Giảng viên hướng dẫn đã tận tình giảng dạy, hướng dẫn chúng em trong suốt quá trình làm dự án công nghệ thông tin 2.

Chúng em cũng xin chân thành cảm ơn quý Thầy/Cô giảng viên khoa Công nghệ thông tin trường Đại học Tôn Đức Thắng đã nhiệt tình cung cấp những kiến thức bổ ích cho chúng em trong suốt quá trình học tập cũng như tạo điều kiện cho chúng em hoàn thành báo cáo môn dự án công nghệ thông tin 2.

Với vốn kiến thức còn hạn hẹp của bản thân và tài liệu tham khảo có hạn nên chúng em không tránh khỏi được những thiếu sót trong quá trình làm báo cáo. Chúng em rất mong nhận được những ý kiến nhận xét, đóng góp, phê bình từ Thầy. Đó sẽ là hành trang quý giá giúp em có thể hoàn thiện kiến thức của mình sau này.

Chúng em xin chân thành cảm ơn.

**BÁO CÁO ĐƯỢC HOÀN THÀNH**

**TẠI TRƯỜNG ĐẠI HỌC TÔN ĐỨC THẮNG**

Chúng tôi xin cam đoan đây là sản phẩm báo cáo của riêng chúng tôi và được sự hướng dẫn của Thầy Nguyễn Quốc Bình. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa công bố dưới bất kỳ hình thức nào trước đây. Những số liệu trong các bảng biểu phục vụ cho việc phân tích, nhận xét, đánh giá được chính tác giả thu thập từ các nguồn khác nhau có ghi rõ trong phần tài liệu tham khảo.

Ngoài ra, trong báo cáo còn sử dụng một số nhận xét, đánh giá cũng như số liệu của các tác giả khác, cơ quan tổ chức khác đều có trích dẫn và chú thích nguồn gốc.

**Nếu phát hiện có bất kỳ sự gian lận nào chúng tôi xin hoàn toàn chịu trách nhiệm về nội dung báo cáo của mình.** Trường Đại học Tôn Đức Thắng không liên quan đến những vi phạm tác quyền, bản quyền do chúng tôi gây ra trong quá trình thực hiện (nếu có).

*TP. Hồ Chí Minh, ngày 06 tháng 01 năm 2020*

*Tác giả*

*(ký tên và ghi rõ họ tên)*

*Trần Bảo Duy*

*Nguyễn Thị Nhật Lệ*

PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN

**Phần xác nhận của GV hướng dẫn**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày.........tháng.........năm.........

(ký và ghi họ tên)

**Phần đánh giá của GV chấm bài**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Tp. Hồ Chí Minh, ngày.........tháng.........năm.........

(ký và ghi họ tên)

TÓM TẮT

Ngày nay một trong những xu hướng truyền dữ liệu nổi bật được ứng dụng phổ biến đó là dữ liệu dòng. Dữ liệu dòng được áp dụng ở nhiều lĩnh vực như trong RFID, Streaming, big data, phân tích dữ liệu. Nó cũng là một trong những đối tượng nổi bật của các việc tấn công mạng. Trong dự án công nghệ thông tin 2 này, chúng em xin xây dựng hệ thống stream video và áp dụng mã hóa dòng để giúp mô tả cách bảo mật cho dữ liệu dòng.

MỤC LỤC

[LỜI CẢM ƠN i](#_Toc29299904)

[PHẦN XÁC NHẬN VÀ ĐÁNH GIÁ CỦA GIẢNG VIÊN iii](#_Toc29299905)

[TÓM TẮT iv](#_Toc29299906)

[MỤC LỤC 1](#_Toc29299907)

[DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT 3](#_Toc29299908)

[DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ 4](#_Toc29299909)

[CHƯƠNG 1 – KHÁI QUÁT VỀ MÃ HÓA DÒNG 6](#_Toc29299910)

[1.1 Dữ liệu 6](#_Toc29299911)

[1.1.1 Dữ liệu dòng 6](#_Toc29299912)

[1.1.2 Truyền dữ liệu 6](#_Toc29299913)

[1.1.3 Thuận lợi, khó khăn 6](#_Toc29299914)

[1.1.4 Ứng dụng 7](#_Toc29299915)

[1.2 Bảo mật dữ liệu dòng với mã hóa dòng 7](#_Toc29299916)

[1.2.1 Mã hóa dòng 8](#_Toc29299917)

[1.2.2 Phân loại mã hóa dòng 9](#_Toc29299918)

[1.2.3 Điểm mạnh và điểm yếu của mã hóa dòng 11](#_Toc29299919)

[1.2.4 Thuật toán RC4 12](#_Toc29299920)

[1.2.5 Thuật toán Salsa20 (Chacha20) 15](#_Toc29299921)

[1.3 Thách thức đối với mã hóa dữ liệu 17](#_Toc29299922)

[1.3.1 Tấn công Fluhrer, Mantin và Shamir 17](#_Toc29299923)

[1.3.2 Tấn công Nomore 17](#_Toc29299924)

[1.3.3 Tấn công Chosen – IV 18](#_Toc29299925)

[CHƯƠNG 2 – XÂY DỰNG HỆ THỐNG MÃ HÓA DÒNG 19](#_Toc29299926)

[2.1 Khái niệm Jitsi 19](#_Toc29299927)

[2.2 Tính năng của Jitsi 19](#_Toc29299928)

[2.3 Hoạt động của Jitsi 20](#_Toc29299929)

[2.4 Hệ thống Jitsi với mã hóa RC4 33](#_Toc29299930)

[2.5 Kiểm tra sự kháng chịu với các tấn công mã hóa khác nhau 33](#_Toc29299931)

[2.5.1 Kiểm tra trên hệ thống sử dụng cipher Chacha20 34](#_Toc29299932)

[2.5.2 Kiểm tra trên hệ thống sử dụng cipher RC4 36](#_Toc29299933)

[CHƯƠNG 3 – KẾT QUẢ THỰC NGHIỆM 38](#_Toc29299934)

[3.1 Bảo mật bằng WEP 38](#_Toc29299935)

[3.2 Tấn công WEP 38](#_Toc29299936)

[3.3 Kiểm tra hiệu suất của thuật toán 40](#_Toc29299937)

[CHƯƠNG 4 – KẾT LUẬN 45](#_Toc29299938)

[TÀI LIỆU THAM KHẢO 46](#_Toc29299939)

DANH MỤC KÍ HIỆU VÀ CHỮ VIẾT TẮT

**CÁC KÍ HIỆU**

XOR Hàm tính toán

**CÁC CHỮ VIẾT TẮT**

OTP One – time – pad

ARX Add – rotation – xor

KSA Key Scheduling Algorithm

PRGA Pseudo Random Generation Algorithm

DANH MỤC CÁC BẢNG BIỂU, HÌNH VẼ, ĐỒ THỊ

**DANH MỤC HÌNH**

[Hình 1.1: Plaintext XOR với keystream để tạo ciphertext 8](#_Toc29299940)

[Hình 1.2: Ví dụ về OTP 9](#_Toc29299941)

[Hình 1.3: Stream cipher đồng bộ 10](#_Toc29299942)

[Hình 1.4: Stream cipher tự đồng bộ 11](#_Toc29299943)

[Hình 1.5: Miêu tả điểm yếu của việc sử dụng lại key của sream cipher 12](#_Toc29299944)

[Hình 1.6: Hoạt động của RC4 14](#_Toc29299945)

[Hình 1.7: Cơ chế mã hóa của Salsa20 16](#_Toc29299946)

[Hình 2.1: Sơ đồ hoạt động của Jitsi 21](#_Toc29299947)

[Hình 2.2: Kiến trúc xây dựng hệ thống Jitsi trên mạng LAN 22](#_Toc29299948)

[Hình 2.3: Cấu hình máy ảo cho môi trường Ubuntu 16.04 23](#_Toc29299949)

[Hình 2.4: Hoàn thành cài đặt Ubuntu 16.04 trên máy ảo 24](#_Toc29299950)

[Hình 2.5: Tường lửa Ubuntu sau khi them các cổng 25](#_Toc29299951)

[Hình 2.6: Nginx 1.17.6 được tích hợp OpenSSL 1.1.1d 26](#_Toc29299952)

[Hình 2.7: Bảng thông báo đặt tên miền / public ip 27](#_Toc29299953)

[Hình 2.8: Lựa chọn cách thiết lập chứng chỉ 28](#_Toc29299954)

[Hình 2.9: Nginx được khởi động lại thành công 32](#_Toc29299955)

[Hình 2.10: Giao diện trang chủ hệ thống Jitsi 33](#_Toc29299956)

[Hình 2.11: Thử nghiệm giao thức và các cipher 35](#_Toc29299957)

[Hình 2.12: Thử nghiệm chuyển tiếp bí mật 35](#_Toc29299958)

[Hình 2.13: Thử nghiệm các điểm yếu 36](#_Toc29299959)

[Hình 2.14: Thử nghiệm giao thức và các cipher trên RC4 36](#_Toc29299960)

[Hình 2.15: Thử nghiệm chuyển tiếp bí mật của hệ thống RC4 37](#_Toc29299961)

[Hình 2.16: Thử nghiệm các điểm yếu của hệ thống RC4 37](#_Toc29299962)

[Hình 3.1: Khởi tạo giá trị IV 39](#_Toc29299963)

[Hình 3.2: Nhập key cho WEP và xuất file WEPOutputSim.csv 39](#_Toc29299964)

[Hình 3.3: Thuật toán tìm key gốc 40](#_Toc29299965)

[Hình 3.4: Key gốc được tìm bởi người tấn công 40](#_Toc29299966)

[Hình 3.5: Áp dụng thuật toán RC4 41](#_Toc29299967)

[Hình 3.6: Hàm tính thời gian bắt đầu và kết thúc mã hóa, giải mã 42](#_Toc29299968)

[Hình 3.7: Biểu đồ thể hiện giá trị thời gian trung bình của RC4 và Chacha20 43](#_Toc29299969)

[Hình 3.8: Biểu đồ thể hiện giá trị throughput trung bình của RC4 và Chacha20 44](#_Toc29299970)

**DANH MỤC BẢNG**

[Bảng 3.1: Thời gian thực thi trung bình của RC4 và Chacha20 42](#_Toc29299971)

[Bảng 3.2: Throughput trung bình của RC4 và Chacha20 44](#_Toc29299972)

1. KHÁI QUÁT VỀ MÃ HÓA DÒNG
   1. Dữ liệu
      1. Dữ liệu dòng

Dữ liệu dòng là một chuỗi các kỹ thuật số mã hóa chặt chẽ tín hiệu (gói của dữ liệu hoặc các gói dữ liệu) được sử dụng để truyền hoặc nhận thông tin trong quá trình được truyền đi. Dữ liệu dòng là một tập hợp các thông tin được trích xuất từ ​​nhà cung cấp dữ liệu. Nó chứa dữ liệu thô được thu thập từ hành vi trình duyệt của người dùng từ các trang web, nơi đặt một pixel chuyên dụng. Dữ liệu dòng rất hữu ích cho các nhà khoa học dữ liệu về dữ liệu lớn và thuật toán.

* + 1. Truyền dữ liệu

Truyền dữ liệu là dữ liệu được tạo ra liên tục bởi hàng ngàn nguồn dữ liệu khác nhau, thường gửi đồng thời các bản ghi dữ liệu và ở kích thước nhỏ (thứ tự Kilobytes). Truyền dữ liệu được xử lý tăng dần bằng cách sử dụng các kỹ thuật xử lý luồng mà không có quyền truy cập vào tất cả dữ liệu.

* + 1. Thuận lợi, khó khăn

Thuận lợi

Truyền dữ liệu ngày càng hữu ích và được áp dụng trong nhiều lĩnh vực.

Áp dụng cho hầu hết các công nghiệp và các lĩnh vực sử dụng dữ liệu lớn. Bắt đầu với các ứng dụng đơn giản như thu thập nhật ký hệ thống và xử lý thô sơ như tính toán tối thiểu. Sau đó, các ứng dụng này phát triển để xử lý gần thời gian thực tinh vi hơn, phân tích dữ liệu phức tạp hơn, áp dụng thuật toán học máy và rút ra những hiểu biết sâu sắc hơn từ dữ liệu.

Khó khăn

Truyền xử lý dữ liệu đòi hỏi hai lớp: Lớp lưu trữ và lớp xử lý.

*Lớp lưu trữ:* Cần hỗ trợ thứ tự bản ghi và tính nhất quán mạnh mẽ để cho phép đọc và ghi nhanh, rẻ tiền và có thể phát lại các luồng dữ liệu lớn.

*Lớp xử lý:* Chịu trách nhiệm tiêu thụ dữ liệu từ lớp lưu trữ, chạy các tính toán trên dữ liệu đó và sau đó thông báo cho lớp lưu trữ để xóa dữ liệu không còn cần thiết.

Phải lập kế hoạch cho khả năng mở rộng, độ bền dữ liệu và khả năng chịu lỗi trong cả hai lớp lưu trữ và xử lý. Danh sách sau đây cho thấy một số điều cần lập kế hoạch khi truyền dữ liệu:

Lập kế hoạch cho khả năng mở rộng.

Lập kế hoạch cho độ bền dữ liệu.

Kết hợp khả năng chịu lỗi trong cả hai lớp lưu trữ và xử lý.

* + 1. Ứng dụng

Truyền dữ liệu đang trở nên hữu ích và cần thiết hơn trong thế giới ngày nay và đang được áp dụng trong một loạt các ngành công nghiệp, một số trong đó đã được đề cập trong các ví dụ như ngành y tế hoặc vận tải. Các ví dụ khác về các ngành hoặc thị trường, nơi truyền dữ liệu được áp dụng, là:

*Tài chính:* Cho phép theo dõi các thay đổi trên thị trường chứng khoán theo thời gian thực, tính toán rủi ro giá trị và tự động cân bằng lại danh mục đầu tư dựa trên biến động giá cổ phiếu.

*Bất động sản:* Các trang web có thể theo dõi một tập hợp dữ liệu từ các thiết bị di động của người tiêu dùng và đưa ra các đề xuất về tài sản theo thời gian thực để truy cập dựa trên vị trí địa lý của họ (Ví dụ Amazon).

*Chơi trò chơi:* Một công ty trò chơi trực tuyến có thể thu thập dữ liệu phát trực tuyến về các tương tác giữa người chơi và đưa dữ liệu vào nền tảng trò chơi của mình (Amazon).

*Thương mại điện tử / Tiếp thị:* Truyền dữ liệu có thể cung cấp tất cả các bản ghi nhấp chuột từ các thuộc tính trực tuyến của nó và tổng hợp và làm phong phú dữ liệu với thông tin nhân khẩu học của người dùng và tối ưu hóa vị trí nội dung trên trang web của mình, mang lại sự phù hợp và trải nghiệm tốt hơn cho khách hàng.

Truyền dữ liệu có ý nghĩa lớn nhất đối với ngành công nghiệp âm thanh, video và viễn thông vì việc tạo ra các dịch vụ truyền phát.

* 1. Bảo mật dữ liệu dòng với mã hóa dòng

Dữ liệu dòng là một trong những loại dữ liệu của tương lai. Khả năng ứng dụng của nó vào cuộc sống và công nghệ hiện tại đã góp phần đưa ra nhiều ý tưởng mới cho ngành công nghệ thông tin nói riêng và cuộc sống con người nói chung. Như mọi loại như dữ liệu khác, dữ liệu dòng là một mục tiêu lớn cho những hacker nhằm khai thác, đánh cắp thông tin được truyền đi trong dữ liệu dòng. Cùng với công nghệ tân tiến hiện nay, các cách tấn công và công cụ của các hackers đã ngày càng đa dạng và đặc biệt hơn nhiều khiến cho việc bảo mật dữ liệu dòng trở nên hết sức cần thiết. Để giúp bảo mật cho dữ liệu, có rất nhiều loại mã hóa có thể áp dụng lên dữ liệu để khiến chúng an toàn hơn và một trong số chúng là mã hóa dòng hay còn được gọi là stream cipher.

* + 1. Mã hóa dòng

Mã hóa dòng (Stream cipher) là 1 loại mã hóa bất đối xứng mà các chữ số của plaintext được kết hợp với các chữ số của dòng chữ số mã hóa định danh (pseudorandom cipher digit stream hay keystream) thường là thông qua phương pháp XOR.

A close up of a device

Description automatically generated

* + - * 1. Plaintext XOR với keystream để tạo ciphertext

Nguồn: cryptosmith.com

Stream cipher được xem như là một bản khác của một loại mã hóa không thể nào phá được, one – time pad (OTP). Cơ chế hoạt động của nó gần như là của stream cipher. OTP là một cơ chế mã hóa, theo đó toàn bộ plaintext XOR với một chuỗi số ngẫu nhiên để tạo ra một bản mã ngẫu nhiên. OTP đã được đánh giá là không thể nào phá được và được xem như có thể mang đến “bảo mật toàn vẹn” nhưng để làm điều đó thì nó phải đảm bảo các yêu cầu sau:

* Chuỗi khóa phải hoàn toàn ngẫu nhiên.
* Chuỗi khóa phải dài bằng hoặc hơn plaintext.
* Chuỗi khóa không được sử dụng lặp lại.
* Chuỗi khóa phải được giữ hoàn toàn bí mật.

A screenshot of a cell phone

Description automatically generated

* + - * 1. Ví dụ về OTP

Nguồn: tutorialspoint.com

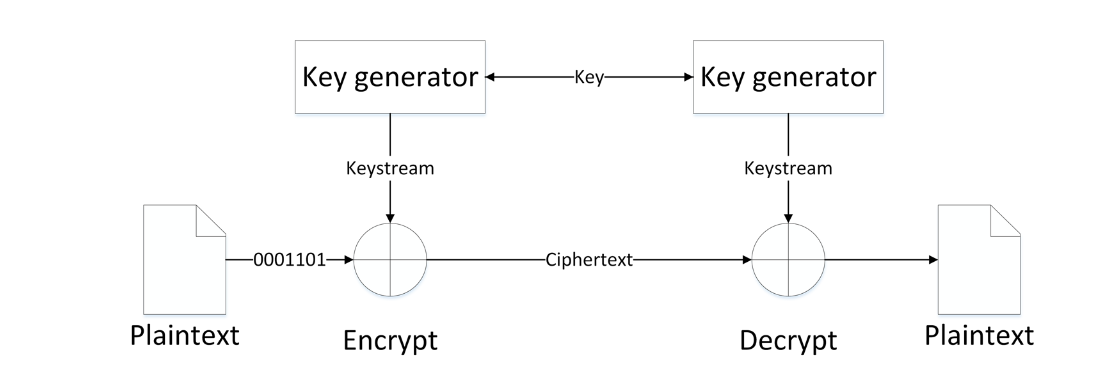
Điều này làm cho hệ thống trở nên cồng kềnh khi thực hiện trong nhiều ứng dụng thực tế và kết quả là OTP không được sử dụng rộng rãi, ngoại trừ các ứng dụng quan trọng nhất. Tạo khóa, phân phối và quản lý là rất quan trọng đối với các ứng dụng đó.

Stream cipher giải quyết các vấn đề này bằng cách hy sinh một mức độ tính bảo mật: nó áp dụng một khóa bí mật, được sử dụng để tạo ra một khóa giả ngẫu nhiên. Khóa bí mật là khóa đối xứng và khóa mã giả được tạo ra được sử dụng theo cách tương tự như chuỗi khóa trong OTPs. Tuy nhiên, thực tế là dòng khóa không hoàn toàn ngẫu nhiên nên gây ra những lo ngại về bảo mật. Luồng khóa phải đủ ngẫu nhiên để đảm bảo rằng nếu kẻ tấn công biết được chuỗi khóa, hắn không thể khôi phục khóa bí mật hoặc lấy được trạng thái nội bộ của mật mã.

* + 1. Phân loại mã hóa dòng

Stream cipher tạo ra chuỗi khóa dựa trên trạng thái nội bộ (Internal state) của nó. Việc này chia stream ciphers ra làm 2 loại tùy vào cách cập nhật trạng thái bên trong:

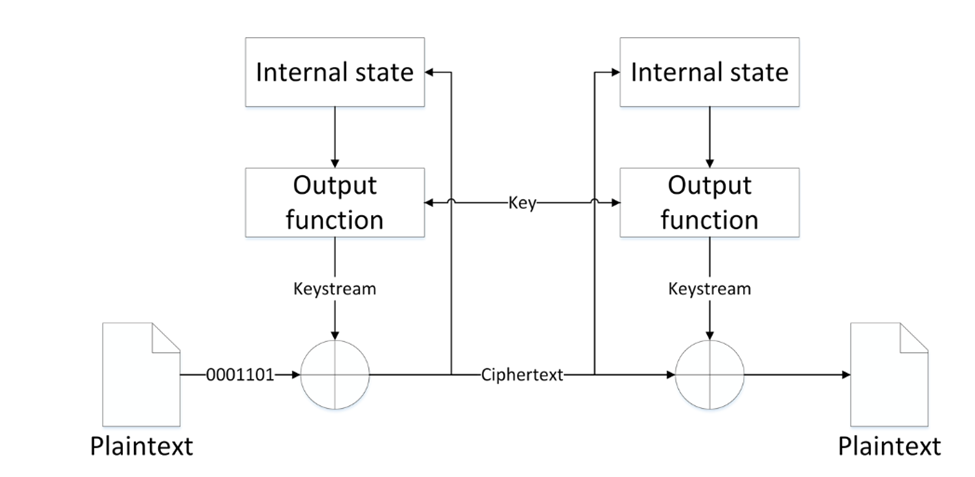
*Stream cipher đồng bộ (Synchronous stream ciphers):* cập nhật trạng thái nội bộ một cách tự nhiên từ dữ liệu plaintext hoặc ciphertext. Người gửi và người nhận phải được đồng bộ hóa trước khi mã hóa / giải mã diễn ra và nếu một phần của thông báo được thêm hoặc xóa, đồng bộ hóa này sẽ bị mất. Mặt khác, nếu một chút của bản mã bị thay đổi, chỉ một bit plaintext duy nhất bị ảnh hưởng, lỗi sẽ không được truyền đến các thông điệp còn lại. Đặc điểm này rất quan trọng trong các ứng dụng như truyền thông không dây, trong đó tỷ lệ lỗi là cao, nhưng nó cũng gây khó khăn cho việc phát hiện lỗi.



* + - * 1. Stream cipher đồng bộ

Nguồn: A survey of lightweight stream ciphers forembedded systems

*Stream cipher tự đồng bộ (Self – synchronizing stream cipher):* stream cipher tự đồng bộ cập nhật trạng thái nội bộ dựa trên N bit mã hóa trước đó: stream cipher tự đồng bộ, người nhận có thể tự động đồng bộ hóa với người gửi sau khi nhận được N bit ciphertext. Điều này làm cho việc khôi phục tin nhắn dễ dàng hơn ngay cả khi một số bit bị chặn hoặc loại bỏ. Nếu một bit bị thay đổi, sẽ chỉ có N bit văn bản sẽ bị ảnh hưởng. Tuy nhiên, trong bối cảnh mật mã, đây cũng là một bất lợi, vì tối đa bit sẽ được giải mã không chính xác cho mỗi bit bị thay đổi. Khóa được cập nhật dựa trên các bit trước đó, các thông số thống kê trong plaintext được tiết lộ trong chuỗi khóa. Mật mã tự đồng bộ hóa chủ yếu dễ bị tổn thương trước các cuộc tấn công mật mã, và kẻ tấn công giải mã bất kỳ phần nào của tin nhắn, hắn có thể giải mã được nhiều hơn.



* + - * 1. Stream cipher tự đồng bộ

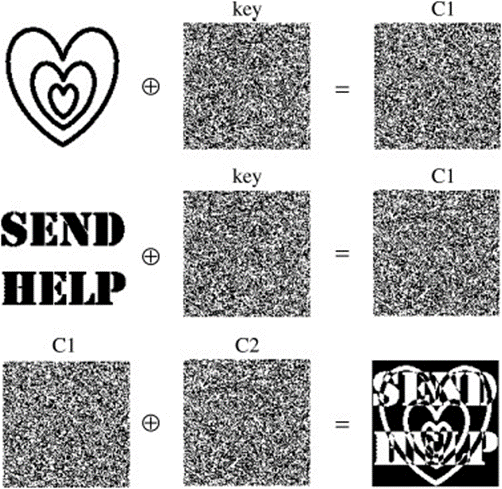
Nguồn: A survey of lightweight stream ciphers forembedded systems

* + 1. Điểm mạnh và điểm yếu của mã hóa dòng

Stream cipher tốt trong việc triển khai nhanh với mức tiêu thụ tài nguyên thấp. Stream cipher hoạt động tốt cho các khối dữ liệu lớn hoặc nhỏ. Chúng phù hợp với kích thước dữ liệu nhỏ hơn vì không cần kích thước khối. Stream cipher cũng hữu ích cho việc mã hóa tín hiệu không dây, phù hợp với mô hình truyền phát hơn là truyền dữ liệu trong các khối lớn hơn, có kích thước cố định. Ví dụ, Stream cipher A5 / 1 được sử dụng trong điện thoại GSM Stream cipher RC4 đã được sử dụng trong hệ thống bảo mật cho mạng cục bộ không dây.

Tuy vậy, Stream cipher vẫn có thể gặp những lỗi an ninh nguy hiểm nếu như không triển khai đúng cách:

*Sử dụng lại key:* Nguyên lý chính của việc sử dụng mật mã luồng một cách an toàn là KHÔNG BAO GIỜ lặp lại việc sử dụng khóa vì thuộc tính cộng đồng của XOR. Nếu C1 và C2 là XOR'd với khóa K, thì việc truy xuất khóa K đó là không đáng kể vì C1 XOR C2 = P1 XOR P2. Kết hợp hai ciphertext được mã hóa với cùng một khóa sẽ loại bỏ khóa, vì nó sẽ XOR với chính nó. Trong trường hợp này, khi khóa được loại bỏ, hai plaintext vẫn còn. XOR dựa trên ngôn ngữ tiếng Anh có nghĩa là các công cụ phân tích mật mã như phân tích tần số ký tự sẽ hoạt động tốt do entropy thấp của ngôn ngữ tiếng Anh.



* + - * 1. Miêu tả điểm yếu của việc sử dụng lại key của sream cipher

Nguồn: sciencedirect.com

*Tấn công lật bit (Bit flipping attack):* Một nguyên lý quan trọng khác của việc sử dụng stream cipher một cách an toàn là xem xét rằng chỉ vì một tin nhắn đã được giải mã, điều đó không có nghĩa là tin nhắn chưa bị giả mạo. Vì việc giải mã dựa trên trạng thái, nếu kẻ tấn công biết cách bố trí của plaintext, một cuộc tấn công Man in the Middle (MITM) có thể lật một chút trong quá trình thay đổi ciphertext bên dưới. Nếu một ciphertext giải mã thành 'Chuyển $ 1000', thì người trung gian có thể lật một bit để ciphertext hóa giải mã thành 'Chuyển $ 9000' vì việc thay đổi một ký tự trong ciphertext không ảnh hưởng đến trạng thái trong stream cipher đồng bộ.

* + 1. Thuật toán RC4

RC4 được thiết kế vào năm 1987 bởi Ron Rivest và là một trong những stream cipher phần mềm rộng rãi nhất và được sử dụng trong các giao thức phổ biến, như SSL (bảo vệ lưu lượng truy cập Internet), WEP (mạng không dây bảo mật). Nó được coi là nhanh và đơn giản về phần mềm. RC4 là một thuật toán mã hóa dòng khóa chung, yêu cầu trao đổi an toàn một khóa được chia sẻ. Thuật toán khóa đối xứng được sử dụng để mã hóa và giải mã sao cho luồng dữ liệu chỉ đơn giản là XOR với chuỗi khóa được tạo. Thuật toán mã hóa RC4 được sử dụng theo tiêu chuẩn chẳng hạn như IEEE 802.11 trong WEP (Giao thức mã hóa không dây) sử dụng các khóa 40 và 128 bit.

Thuật toán sử dụng khóa có độ dài thay đổi từ 1 đến 256 byte để khởi tạo bảng trạng thái 256 byte. Bảng trạng thái được sử dụng để tạo các byte giả ngẫu nhiên tiếp theo và sau đó để tạo ra một luồng giả ngẫu nhiên được XOR với plaintext để cung cấp ciphertext. Mỗi phần tử trong bảng trạng thái được hoán đổi ít nhất một lần.

Thuật toán hoạt động theo hai giai đoạn, thiết lập khóa và mã hóa.

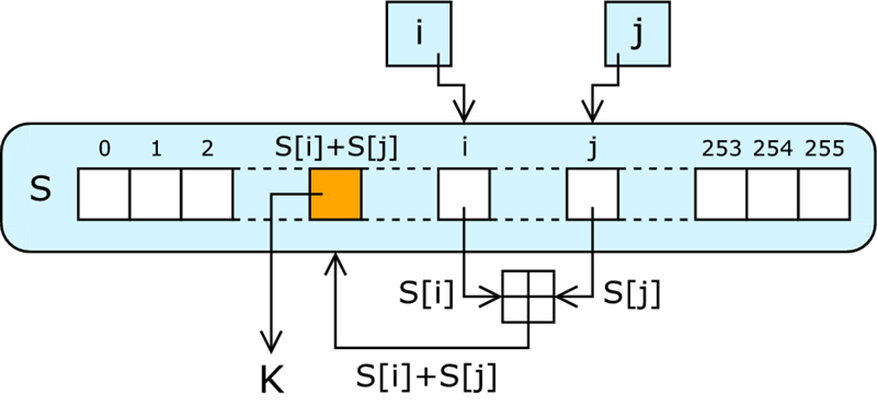
*Thiết lập khóa*: Thiết lập khóa là giai đoạn đầu tiên và khó khăn nhất của thuật toán mã hóa này. Trong quá trình thiết lập khóa N – bit (N là độ dài khóa), khóa mã hóa được sử dụng để tạo biến mã hóa bằng hai mảng, trạng thái của khóa và N số các hoạt động trộn. Các hoạt động trộn này bao gồm các byte hoán đổi, các hoạt động modulo và các công thức khác. Một phép toán modulo là quá trình thu được phần còn lại từ phép chia. Chẳng hạn, 11/4 là 2 phần còn lại 3; do đó mười một mod bốn sẽ bằng ba.

Bảng T dài 256 byte và được tạo dựa trên khóa bí mật. Nó được tạo ra như một bước đầu tiên của cả mã hóa và giải mã. Các hoạt động sau đây phải được thực hiện để tạo bảng:

* Mỗi ô trong bảng được điền một số bằng vị trí của nó. Các vị trí của bảng được đánh số từ 0 đến 255.
* Một biến tạm thời mới được tạo và được đặt thành 0.
* Đối với mỗi phần tử trong mảng, hai thao tác sau được thực hiện:
* Giá trị của biến tạm thời được cập nhật.
* Số trong mảng tại vị trí hiện tại được hoán đổi với số trong mảng tại vị trí được xác định bởi biến tạm thời.

*Giai đoạn mã hóa*: Trong quá trình mã hóa và giải mã, các byte trong keystream được tạo ra liên tục. Chúng được thêm XOR vào byte của thông điệp. Các byte keystream được tạo ra dựa trên bảng T. Các bước sau đây được thực hiện:

* Hai biến trợ giúp p1 và p2 được tạo và đặt thành 0.
* Biến p1 được tăng thêm 1 và kết quả là modulo chia cho 256.
* Biến p2 được tăng bởi giá trị trong mảng T tại vị trí được xác định bởi biến tạm thời p1 (T[p1]). Sau đó, kết quả được chia modulo cho 256.
* Giá trị trong mảng tại vị trí p1 được hoán đổi với giá trị trong mảng tại vị trí p2.
* Giá trị trong mảng tại vị trí p1 được thêm vào giá trị trong mảng tại vị trí p2. Sau đó, kết quả là modulo chia cho 256 và được gán cho biến helper mới p3.
* Giá trị trong mảng tại vị trí p3 là một byte dòng khóa mới.
* Nếu cần nhiều byte dòng chính hơn, tất cả các bước từ điểm 2 trở đi sẽ được lặp lại.



* + - * 1. Hoạt động của RC4

Nguồn: wikipedia

Mặc dù là một trong những Stream cipher phổ biến nhất nhưng RC4 đã được tạo ra cách đây khá lâu và nó có một số điểm yếu đã được cải thiện trong Stream cipher hiện đại.

Có lẽ điểm yếu quan trọng nhất của mật mã RC4 là lịch trình khóa không đủ. Do vấn đề đó, có thể có được một số thông tin về khóa bí mật dựa trên các byte đầu tiên của keystream. Bạn chỉ nên loại bỏ một số byte đầu tiên của dòng khóa. Cải tiến này được gọi là RC4 – dropN, trong đó N thường là bội số của 256.

RC4 không có một nonce (một số tùy ý có thể được sử dụng chỉ một lần trong giao tiếp mật mã) riêng biệt bên cạnh khóa cho mỗi mã hóa, để lại giao thức truyền thông để xác định cách kết hợp hai tham số này và tạo ra keystream. Do đó, hệ thống mật mã phải chăm sóc các giá trị độc nhất của keystream và chỉ định cách kết hợp nonce với khóa bí mật ban đầu. Ý tưởng tốt nhất sẽ là hash nonce và khóa cùng nhau để tạo cơ sở để tạo keystream RC4. Thật không may, nhiều ứng dụng chỉ đơn giản là ghép khóa và nonce, khiến chúng dễ bị tấn công.

* + 1. Thuật toán Salsa20 (Chacha20)

Salsa20 là mật mã dòng được phát triển bởi Daniel J. Bernstein vào năm 2005, sau đó được gửi tới eSTREAM (the ECRYPT Stream Cipher Project.). Salsa20 được xây dựng trên hàm giả ngẫu nhiên dựa trên các hoạt động add – rotation – xor (ARX) – bổ sung 32 bit, thêm bit (XOR) và thao tác xoay (round function). Cách thức chính salsa20 là ánh xạ khóa 256 bit, nonce 64 bit và bộ đếm 64 bit thành khối 512 bit của keystream (cũng tồn tại phiên bản Salsa có khóa 128 bit). Điều này mang lại cho Salsa20 lợi thế khác thường mà người dùng có thể tìm kiếm một cách hiệu quả ở bất kỳ vị trí nào trong keystream trong thời gian liên tục và tránh các timing attack. Mỗi khối đầu ra như vậy là một sự kết hợp độc lập của khóa, nonce và bộ đếm và do không có sự kết nối giữa các khối, hoạt động của Salsa20 giống như hoạt động của mật mã khối trong chế độ bộ đếm. Do đó, Salsa20 chia sẻ những lợi thế triển khai rất giống nhau, đặc biệt là khả năng tạo các khối đầu ra theo bất kỳ thứ tự nào và song song.

*Mã hóa:*

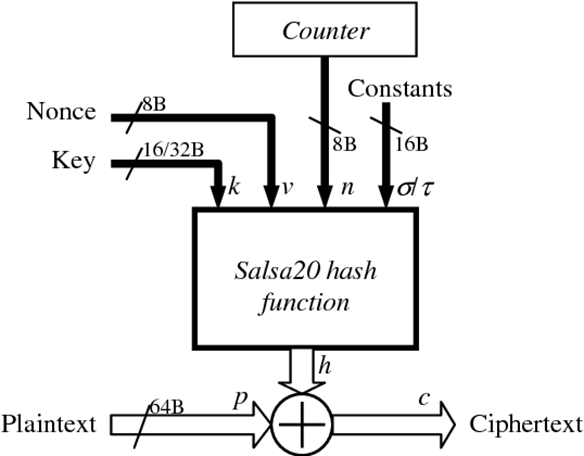
Đối với mỗi khối dữ liệu 64 byte, thuật toán sử dụng hàm mở rộng Salsa20. Đầu vào của hàm là khóa bí mật (có thể có 32 hoặc 16 byte) và một chuỗi nonce dài 8 byte được nối với một số khối, giá trị này thay đổi từ 0 đến 264 - 1 (nó cũng được lưu trữ trên 8 byte). Mỗi cuộc gọi đến hàm mở rộng sẽ tăng số khối lên một.

Cốt lõi của thuật toán mã hóa Salsa20 là hàm băm nhận dữ liệu đầu vào dài 64 byte từ hàm mở rộng Salsa20, trộn nó và cuối cùng trả về đầu ra dài 64 byte. Hàm băm Salsa20 hoạt động trên chuỗi byte nhận được, bao gồm:

* Khóa bí mật.
* Nonce với số khối.
* Bốn vectơ không đổi nhận được từ hàm mở rộng, giá trị này phụ thuộc vào kích thước của khóa bí mật.

Hàm băm hoạt động trên dữ liệu được chia thành các từ. Mỗi từ chứa 4 byte và có thể có các giá trị từ 0 đến 232 – 1. Do đó, dữ liệu đầu vào dài 16 từ, một khóa chứa 8 hoặc 4 từ và nonce có 2 từ.

Đầu ra từ chức năng mở rộng Salsa20 được thêm XOR vào khối dữ liệu 64 byte. Kết quả là một khối ciphertext 64 byte.



* + - * 1. Cơ chế mã hóa của Salsa20

Nguồn: semanticscholar

*Giải mã:*

Thuật toán tương tự nên được sử dụng trong quá trình giải mã. Dữ liệu nên được chia thành các phần có cùng kích thước.

Đầu ra từ chức năng mở rộng Salsa20 nên được thêm XOR vào khối ciphertext 64 byte. Kết quả là một khối plaintext 64 byte.

*Khả năng bảo mật:*

Salsa20 là một trong những stream cipher có tốc độ cao, tùy theo phiên bản mà Salsa20 thích hợp cho cả phần cứng lẫn phần mềm. Cũng như là một trong những stream cipher được đánh giá cao trong thời điểm hiện tại khi mà RC4 đã được phát hiện có quá nhiều lỗ hổng trong an ninh. Phiên bản Salsa20/12 của Salsa20 là một trong 4 stream cipher đã được lựa chọn trong giai đoạn 3 của profile 1 (phần mềm) của eSTREAM project cùng với HC – 128, Rabbit, SOSEMANUK. Trong những năm sau khi xuất bản, Salsa20 đã trải qua các cuộc phân tích mật mã quan trọng. Mặc dù một số cuộc tấn công đã được tìm thấy trên các phiên bản rút gọn của mật mã, không có cuộc tấn công nào tốt hơn ngoài cách tấn công tìm kiếm khóa đầy đủ trên Salsa20 / 12 hoặc Salsa20 / 20.

* 1. Thách thức đối với mã hóa dữ liệu
     1. Tấn công Fluhrer, Mantin và Shamir

Năm 2001, một phát hiện mới và ngạc nhiên đã được Fluhrer, Mantin và Shamir thực hiện. Trên tất cả các khóa RC4, các số liệu thống kê cho một vài byte đầu tiên của dòng khóa đầu ra là thông tin rò rỉ không ngẫu nhiên về khóa. Nếu khóa nonce và khóa dài hạn được ghép đơn giản để tạo khóa RC4, khóa dài hạn này có thể được phát hiện bằng cách phân tích một số lượng lớn tin nhắn được mã hóa bằng khóa này. Điều này và các hiệu ứng liên quan sau đó đã được sử dụng để phá vỡ mã hóa WEP được sử dụng với các mạng không dây 802.11.

Các giao thức có thể bảo vệ chống lại cuộc tấn công này bằng cách loại bỏ phần ban đầu của dòng chính. Một thuật toán được sửa đổi như vậy thường được gọi là "RC4 – drop [n]", trong đó n là số byte dòng khóa ban đầu được loại bỏ. Mặc định SCAN là n = 768 byte, nhưng giá trị bảo toàn sẽ là n = 3072 byte.

Cuộc tấn công Fluhrer, Mantin và Shamir không áp dụng cho SSL dựa trên RC4, vì SSL tạo ra các khóa mã hóa mà nó sử dụng cho RC4 bằng cách băm, có nghĩa là các phiên SSL khác nhau có các khóa không liên quan.

* + 1. Tấn công Nomore

Vào năm 2015, các nhà nghiên cứu bảo mật từ KU Leuven đã trình bày các cuộc tấn công mới chống lại RC4 trong cả TLS và WPA – TKIP. Được mệnh danh là cuộc tấn công khai thác và phục hồi khai thác (Nomore) nhiều lần, đây là cuộc tấn công đầu tiên thuộc loại này đã được chứng minh trong thực tế. Cuộc tấn công của họ chống lại TLS có thể giải mã cookie HTTP an toàn trong vòng 75 giờ. Cuộc tấn công chống lại WPA – TKIP có thể được hoàn thành trong vòng một giờ và cho phép kẻ tấn công giải mã và tiêm các gói tùy ý.

* + 1. Tấn công Chosen – IV

Mật mã dòng kết hợp một khóa bí mật với một vectơ khởi tạo đã được thống nhất (IV) để tạo ra một chuỗi giả ngẫu nhiên mà theo thời gian được đồng bộ hóa lại. Một cuộc tấn công "Chosen IV" phụ thuộc vào việc tìm kiếm các IV cụ thể được kết hợp với nhau có thể sẽ tiết lộ thông tin về khóa bí mật. Thông thường, nhiều cặp IV được chọn và sự khác biệt trong các luồng khóa được tạo sau đó được phân tích thống kê cho mối tương quan tuyến tính và / hoặc mối quan hệ boolean đại số. Nếu việc chọn các giá trị cụ thể của vectơ khởi tạo sẽ làm lộ ra một mẫu không ngẫu nhiên trong chuỗi được tạo, thì cuộc tấn công này sẽ tính toán một số bit và do đó rút ngắn độ dài khóa hiệu quả. Một triệu chứng của cuộc tấn công sẽ được đồng bộ hóa lại thường xuyên. Mật mã dòng hiện đại bao gồm các bước để trộn đầy đủ khóa bí mật với một vectơ khởi tạo, thường bằng cách thực hiện nhiều vòng ban đầu.

1. XÂY DỰNG HỆ THỐNG MÃ HÓA DÒNG

Các ứng dụng hiện nay của dữ liệu dòng vào hệ thống công nghệ hiện đại có mặt khắp mọi nơi từ xem video, theo dõi thiết bị RFID cho tới các nền tảng video streaming. Lấy nền tảng từ các ứng dụng đó, chúng em định hướng để làm cài đặt một hệ thống call video trực tuyến nhiều người dùng trong nội bộ mạng LAN.

Khi nói đến call video trực tuyến, nhiều người sẽ nghĩ ngay đến ứng dụng các ứng dụng như Skype hay Microsoft Team. Chúng đều là những ứng dụng call video trực tuyến nổi tiếng được sử dụng trong đời sống hàng ngày cũng như trong công việc. Nhưng việc xây dụng nên một hệ thống như thế là một việc quá tốn kém, tiêu chí xây dựng hệ thống call video của em là một hệ thống nhẹ, đơn giản, có thể xây dựng được trong mạng nội bộ một cách dễ dàng để có thể dễ dàng triển khai trong các môi trường như trường học, nơi cư trú, chung cư. Bởi vì thế chúng em tìm đến hệ thống call video mã nguồn mở Jitsi.

* 1. Khái niệm Jitsi

Jitsi là một tập hợp các ứng dụng thoại đa nền tảng miễn phí và mã nguồn mở bao gồm các chức năng như nói qua ip, video call và ứng dụng nhắn tin tức thời cho nền tảng web, Windows, Linux, Mac OS X và Android. Dự án Jitsi bắt đầu với Máy tính để bàn Jitsi (trước đây gọi là SIP Communicator). Với sự phát triển của công nghệ WebRTC, nhóm dự án đã chuyển sang jitsi video bridge để cho phép gọi call video dựa trên web. Sau đó, nhóm đã thêm Jitsi Meet, một ứng dụng gọi hội nghị video đầy đủ bao gồm các ứng dụng web, Android và iOS. Giống như các dự án nguồn mở khác, với tư cách là một mô hình kinh doanh, Jitsi về cơ bản là một công ty tư vấn, công nghệ có sẵn miễn phí và bất kỳ ai cũng có thể sử dụng công nghệ và cải tiến.

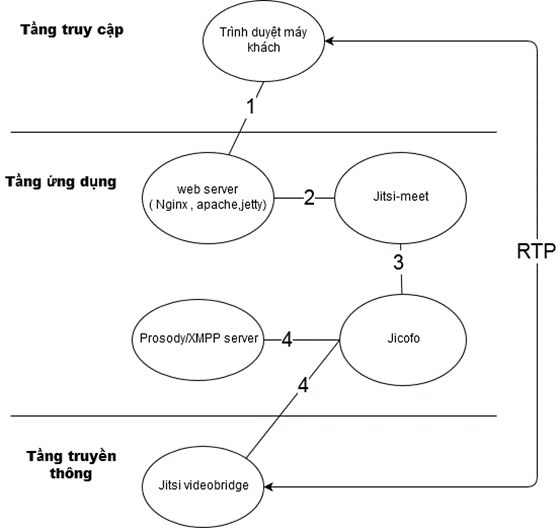
* 1. Tính năng của Jitsi

Một số tính năng của Jitsi như sau:

* Tin nhắn tức thời giữa các user trong cuộc hội thoại (trên nền tảng web).
* Khóa phòng hội thảo với mật khẩu.
* Chia sẻ màn hình.
* Stream cuộc hội thảo đến youtube.
* Share một video Youtube trong cuộc hội thoại.
* Khả năng tích hợp với các website / ứng dụng khác thông qua API.

Jitsi được cấu thành từ nhiều thành phần mã nguồn mở khác nhau:

* Một web server miễn phí và mã nguồn mở để quản lí các yêu cầu của client đến jitsi – meet. Các web server mà jitsi hỗ trợ là nginx, apache và jetty.
* Prosody: một máy chủ nhắn tin tức thời XMPP đa nền tảng miễn phí được viết bằng ngôn ngữ của Lua. Tính năng của nó là sử dụng tài nguyên thấp, dễ sử dụng và mô – đun hóa.
* Jitsi – Meet: Ứng dụng JavaScript (phía máy khách) tích hợp API WebRTC cũng như tất cả các giao diện đồ họa của dịch vụ.
* Jitsi – videobridge: Trong cơ sở hạ tầng Jitsi, Jitsi – videobridge hoạt động như một máy chủ phương tiện, cho phép giao tiếp âm thanh / video nhiều người dùng. Nó là một đơn vị chuyển tiếp chọn lọc, tương tự như cấu trúc liên kết mạng sao, còn được gọi là Hub và Spoke, đây là cấu trúc liên kết phổ biến nhất hiện nay. Nó cũng rất linh hoạt về mặt quản lý và xử lý sự cố mạng: sự cố của một nút không làm xáo trộn chức năng chung của mạng. Mặt khác, thiết bị trung tâm kết nối tất cả các nút tạo thành một điểm thất bại duy nhất: lỗi ở cấp độ này làm cho mạng hoàn toàn không thể sử dụng được. Mỗi người tham gia nhận được các luồng phương tiện truyền thông độc lập.
* Jicofo: Một máy chủ báo hiệu và phiên được sử dụng để tạo một phiên bản mới của một hội nghị và để duy trì các phiên khác nhau giữa mỗi người tham gia và Jitsi – videobridge.
  1. Hoạt động của Jitsi

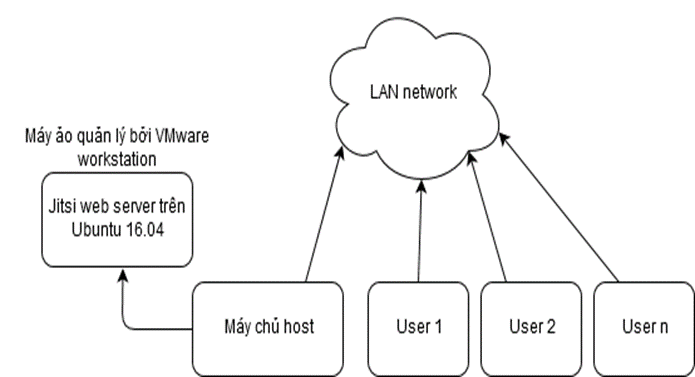


* + - * 1. Sơ đồ hoạt động của Jitsi

Để triển khai hệ thống Jitsi, chúng em sử dụng phần mềm quản lý máy ảo Vmware Workstation. VMware Workstation là một trình ảo hóa được lưu trữ chạy trên các phiên bản x64 của hệ điều hành Windows và Linux (phiên bản x86 của các bản phát hành trước đó đã có sẵn). Nó cho phép người dùng thiết lập máy ảo (VM) trên một máy vật lý duy nhất và sử dụng chúng đồng thời cùng với máy thật. Mỗi máy ảo có thể thực thi hệ điều hành riêng của mình, bao gồm các phiên bản Microsoft Windows, Linux, BSD và MS – DOS.

Thay vì phải cài đặt lên một máy tính cụ thể trong mạng nội bộ, Vmware workstation đóng vai trò tạo máy ảo quản lý môi trường để cài đặt hệ thống Jitsi. Việc này giúp cho việc quản lý Jitsi dễ dàng hơn vì với phần mềm quản lý máy ảo chúng em có thể sử dụng chức năng snapshot để lưu lại trạng thái của máy ảo, giúp cho việc sửa lỗi, quản lí cấu hình, thử nghiệm hệ thống được lưu loát hơn là cài đặt môi trường lên một máy tính.

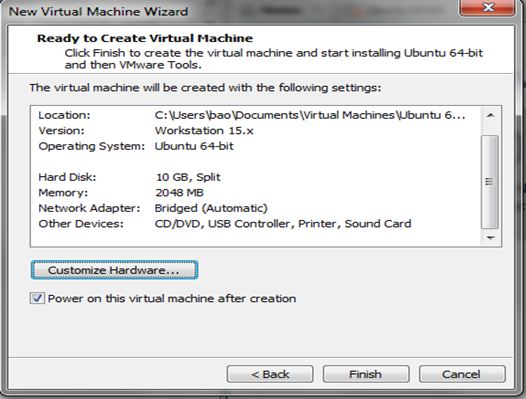
Để triển khai server, môi trường thích hợp nhất chính là môi trường hệ điều hành Linux và hệ điều hành chúng em sử dụng để cài đặt hệ thống Jitsi là Ubuntu 16.04 LTS. Hệ điều hành Linux nổi tiếng phục vụ cho các server vì cấu hình nhẹ, miễn phí và mã nguồn mở, linh hoạt và bảo mật tốt. Trong đó, Ubuntu là một trong những hệ điều hành linux nổi bật bởi vì không chỉ hỗ trợ người dùng với các câu lệnh command line thuần của linux, Ubuntu còn có giao diện đồ họa tích hợp sẵn để người dùng có thể dễ dàng quản lý, thao tác lên hệ điều hành.



* + - * 1. Kiến trúc xây dựng hệ thống Jitsi trên mạng LAN

Để tiến hành xây dựng Jisti, trước tiên chúng ta cần phải chuẩn bị máy ảo và môi trường hệ điều hành Ubuntu 16.04. Vì là một hệ điều hành mã nguồn mở, image của Ubuntu 16.04 có thể được tải về từ nhiều nơi khác nhau trên mạng nhưng một trong những nguồn đáng tin cậy nhất đó là từ chính website của Ubuntu – releases.ubuntu.com. Phiên bản Ubuntu 16.04 chúng em chọn sẽ là bản 64 bit.

Sau khi chuẩn bị image của Ubuntu 16.04, chúng ta tiến hành dùng nó để tạo máy ảo thông qua Vmware workstation. Sử dụng chức năng Create a New Virtual Machine để tiến hành tạo một máy ảo có cấu hình như sau:

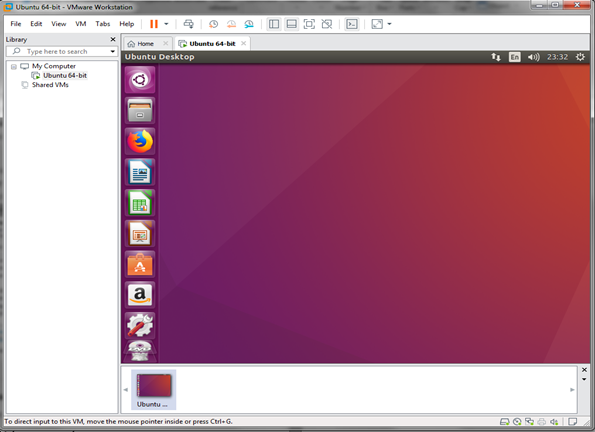


* + - * 1. Cấu hình máy ảo cho môi trường Ubuntu 16.04

Trong đó, có một số cấu hình quan trọng như:

* Operating System: Đây là hệ điều hành của máy ảo. Chúng ta có thể tự chọn hoặc là khi đưa image vào thì Vmware station sẽ tự động lựa chọn sẵn.
* Hard disk: Đây là dung lượng ổ đĩa cứng của máy ảo được lưu trữ trên máy host. Tính năng Split là dùng để chia nhỏ ổ đĩa máy ảo ra làm nhiều mảnh để dễ dàng di chuyển ổ đĩa nếu cần.
* Memory: Số lượng RAM của máy ảo khi chạy trên máy host.
* Network adapter: Bridged networking sẽ kết nối máy ảo với mạng internet thông qua bộ chuyển đổi internet của máy ảo.

Khi Vmware workstation hoàn thành việc chuẩn bị máy ảo, nó sẽ tự động khởi động máy ảo lên. Hệ điều hành Ubuntu 16.04 sẽ khởi động và yêu cầu người dùng cài đặt Ubuntu. Chúng em tiến hành cài đặt Ubuntu như bình thường mà không thay đổi hay thêm cấu hình gì.

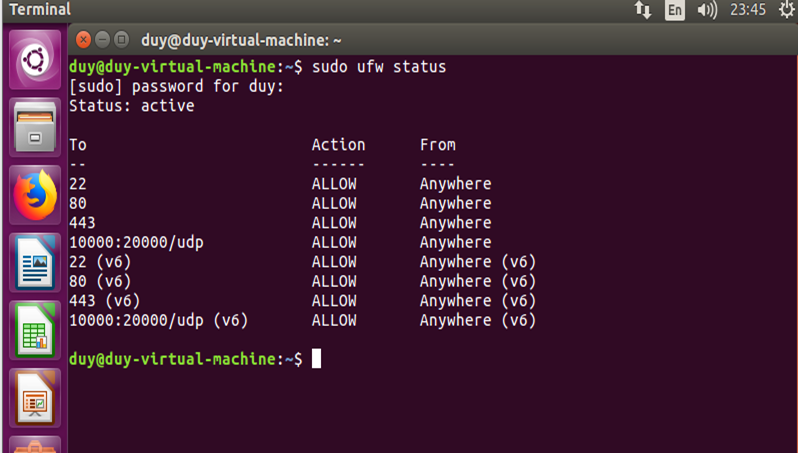


* + - * 1. Hoàn thành cài đặt Ubuntu 16.04 trên máy ảo

Một trong những việc trước tiên đó là chúng em sẽ thiết lập tường lửa cho máy ảo cũng như thiết lập các port cần thiết để web server giao tiếp lên Network (22, 443). Ở Ubuntu, tường lửa được mặc định tắt, sử dụng câu lệnh sudo ufw enable để tiến hành khởi động tường lửa. Sau đó, chúng em thêm từng port cần thiết vào tường lửa. Các port chúng em thêm vào là:

* Port 22 cho việc SSL.
* Port 80 cho HTTP.
* Port 443 cho HTTPS.
* Port 10000 – 20000 cho việc “voice – over – ip” của Jitsi.

Việc thêm các cổng này là cần thiết cho web server cũng như Jitsi có thể đi ra ngoài Network và kết nối đến với người dùng đầu cuối. Sử dụng câu lệnh sudo ufw status để xem trạng thái của tường lửa, nó sẽ như hình dưới sau khi đã thêm các port cần thiết:



* + - * 1. Tường lửa Ubuntu sau khi them các cổng

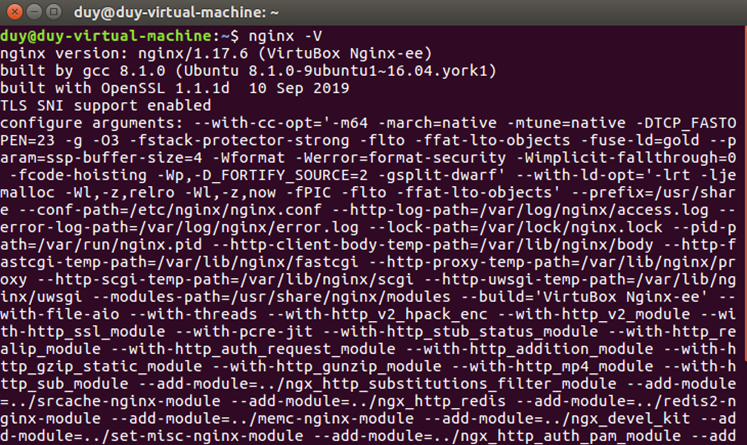
Một trong những cách để trao đổi dữ liệu trên mạng Internet an toàn là sử dụng các giao thức mã hóa. Đối với nền tảng web, giao thức quan trọng và được tin cậy nhất đó là TLS (Transport layer protocol). Tuy vậy, TLS chỉ là các giao thức trừu tượng hoặc áp dụng các hàm liên quan tới bảo mật, áp dụng các phương pháp mã hóa vì thế nên việc tích hợp TLS vào trong việc xây dựng các hệ thống web không phải là một chuyện dễ dàng. Vì thế một dự án được triển khai để tạo một thư viện có thể dễ dàng áp dụng TLS vào các hệ thống web, dự án đó có tên là OpenSSL. OpenSSL là một thư viện phần mềm dành cho các ứng dụng bảo mật liên lạc qua mạng máy tính chống nghe lén hoặc cần xác định bên ở đầu kia. Nó được sử dụng rộng rãi bởi các máy chủ Internet, bao gồm phần lớn các trang web HTTPS. Bởi vì tính phổ biến cũng như tầm quan trọng của OpenSSL và TLS trên hệ thống web, chúng em thấy rằng nó phù hợp với việc xây dựng hệ thống Jitsi.

OpenSSL được áp dụng và quản lý bởi web server của hệ thống Jitsi. Các dữ liệu từ Jitsi trước khi đi ra mạng ngoài được áp dụng các phương thức bảo mật được cấu hình trên web server dựa trên thư viện OpenSSL. Với web server nginx, nó đã được tích hợp sẵn OpenSSL ở nguồn. Tuy nhiên, phiên bản OpenSSL được tích hợp sẵn của nginx là 1.0.2 – một phiên bản quá cũ để sử dụng cho các việc bảo mật HTTPS hiện tại vì chúng không có các phiên bản mã hóa mới, chưa được tối ưu. Để có thể tích hợp một phiên bản mới OpenSSL mới hơn vào nginx, chúng em phải biên dịch lại nginx từ nguồn với một phiên bản OpenSSL mới và sau đó cài đặt nó lên Ubuntu.

Việc biên dịch lại nginx rất cầu kỳ và tốn thời gian, thay vì phải biên dịch nginx bằng tay, chúng em sử dụng các script để có thể tự động biên dịch nginx cùng với các module cần thiết như OpenSSL. Một trong những script như vậy được gọi là nginx-ee được làm ra bởi người dùng VirtuBox cùng với centminmod, hakase, Karl Chen. Nginx-ee là một script mã nguồn mở, mã nguồn của nó có thể được tìm thấy trên Github.

Để tiến hành sử dụng nginx-ee, chúng ta chỉ cần sử dụng một câu lệnh script đơn giản bash <(wget -O - vtb.cx/nginx-ee || curl -sL vtb.cx/nginx-ee). Câu lệnh của nginx-ee sẽ cài đặt phiên bản mainline mới nhất của nginx (1.17.6) được tích hợp với OpenSSL 1.11.1d. Câu lệnh trên có thể tùy chỉnh để cài đặt các module khác nhau, thông tin về các module được hỗ trợ có thể được tìm thấy ở trang Github của nginx – ee.

Sau khi nginx – ee hoàn thành cài đặt, chúng em gõ lệnh nginx –V để kiểm tra phiên bản của nginx và OpenSSL được tích hợp với nó:



* + - * 1. Nginx 1.17.6 được tích hợp OpenSSL 1.1.1d

Khi việc chuẩn bị web server và OpenSSL hoàn tất, chúng em tiến hành cài đặt Jitsi. Trước tiên, thêm kho lưu trữ của Jitsi bằng câu lệnh sau

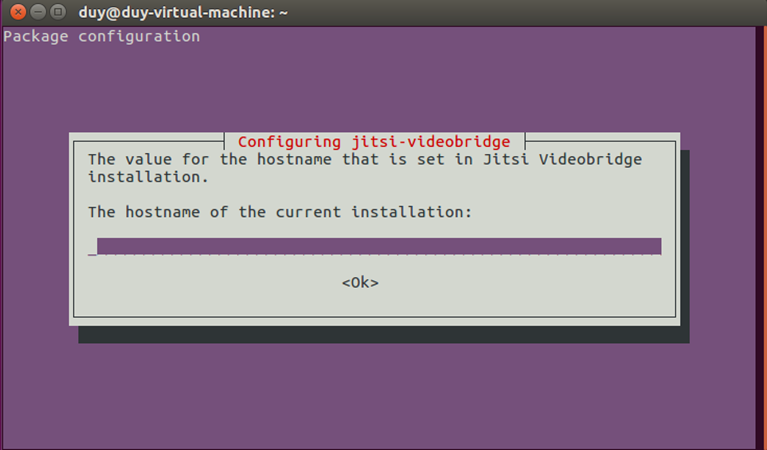
echo 'deb https://download.jitsi.org stable/' >> /etc/apt/sources.list.d/jitsi-stable.listwget -qO - https://download.jitsi.org/jitsi-key.gpg.key | apt-key add -

Tiến hành update và cài đặt Jitsi:

apt-get update

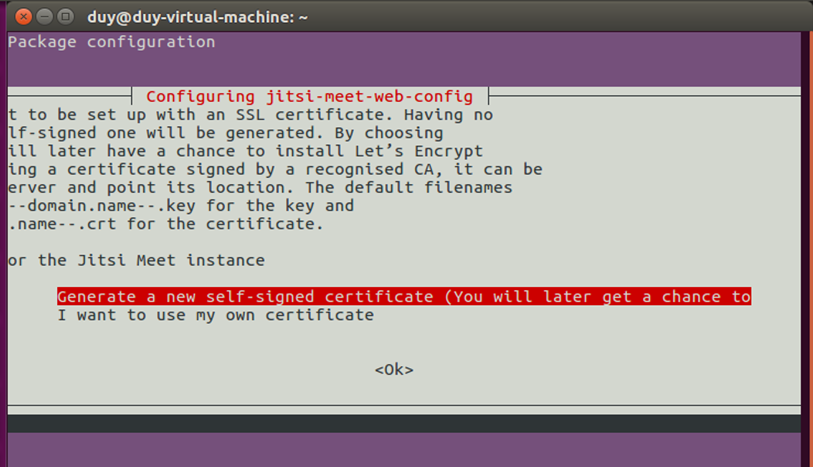
apt-get -y install jitsi-meet

Trong lúc cài đặt Jitsi, Jitsi sẽ hiện thông báo cho phép chúng em cấu hình tên miền của Jitsi-bridge hoặc là địa chỉ public ip của máy chủ hệ thống Jitsi. Chúng em sẽ sử dụng public ip của máy chủ vì để sử dụng tên miền, nó phải được mua thông các nhà cung cấp tên miền trong khi public ip đã được cấp sẵn khi tạo máy ảo.



* + - * 1. Bảng thông báo đặt tên miền / public ip

Sau đó, Jitsi sẽ cho phép chúng em lựa chọn chứng chỉ khóa công khai như sau:



* + - * 1. Lựa chọn cách thiết lập chứng chỉ

Chứng chỉ khóa công khai (Public key certificate) là một tài liệu điện tử được sử dụng để chứng minh quyền sở hữu của khóa công khai. Các loại chứng chỉ này được cung cấp bởi các nhà cơ quan cung cấp chứng chỉ như Let’s Encrypt. Jitsi có ba hướng để cấu hình chứng chỉ cho người dùng trong lúc cài đặt:

* Nếu như tên miền đã được cấp chứng chỉ, lựa chọn phương án thứ hai sẽ cho phép sử dụng chứng chỉ đã được cung cấp.
* Nếu như đã có tên miền nhưng chưa được cấp chứng chỉ, người dùng có thể thông qua Let’s Encrypt để tạo một chứng chỉ cho mình rồi sau đó cấu hình cho Jitsi.
* Người dùng có thể “tự ký” chứng chỉ cho tên miền của mình trong quá trình cài đặt.

Chúng em sẽ sử dụng lựa chọn “tự ký” chứng chỉ vì hai phương án kia đòi hỏi phải có tên miền hợp lệ. Sau khi lựa chọn xong, việc cài đặt Jitsi sẽ được tiếp tục.

Giai đoạn cuối cùng là việc cấu hình Jitsi hoạt động với web server nginx. Để nginx có thể quản lý được Jitsi, nó cần một tập tin cấu hình của Jitsi. Sử dụng lệnh sudo gedit, chúng em sử dụng trình hỗ trợ văn bản của Ubuntu để tạo ra một tập tin có nội dung như sau:

server\_names\_hash\_bucket\_size 64;

server {

listen 80;

server\_name public-ip;

return 301 https://$host$request\_uri;

}

server {

listen 443 ssl;

server\_name public-ip;

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_prefer\_server\_ciphers on;

ssl\_ciphers "ECDHE-RSA-CHACHA20-POLY1305";

add\_header Strict-Transport-Security "max-age=31536000";

ssl\_certificate /etc/jitsi/meet/jitsi-meet.example.com.crt;

ssl\_certificate\_key /etc/jitsi/meet/jitsi-meet.example.com.key;

root /usr/share/jitsi-meet;

# ssi on with javascript for multidomain variables in config.js

ssi on;

ssi\_types application/x-javascript application/javascript;

index index.html index.htm;

error\_page 404 /static/404.html;

location = /config.js {

alias /etc/jitsi/meet/public-ip.com-config.js;

}

location = /external\_api.js {

alias /usr/share/jitsi-meet/libs/external\_api.min.js;

}

#ensure all static content can always be found first

location ~ ^/(libs|css|static|images|fonts|lang|sounds|connection\_optimization|.well-known)/(.\*)$

{

add\_header 'Access-Control-Allow-Origin' '\*';

alias /usr/share/jitsi-meet/$1/$2;

}

# BOSH

location = /http-bind {

proxy\_pass http://localhost:5280/http-bind;

proxy\_set\_header X-Forwarded-For $remote\_addr;

proxy\_set\_header Host $http\_host;

}

location ~ ^/([^/?&:'"]+)$ {

try\_files $uri @root\_path;

}

location @root\_path {

rewrite ^/(.\*)$ / break;

}

location ~ ^/([^/?&:'"]+)/config.js$

{

set $subdomain "$1.";

set $subdir "$1/";

alias /etc/jitsi/meet/public-ip.com-config.js;

}

#Anything that didn't match above, and isn't a real file, assume it's a room name and redirect to /

location ~ ^/([^/?&:'"]+)/(.\*)$ {

set $subdomain "$1.";

set $subdir "$1/";

rewrite ^/([^/?&:'"]+)/(.\*)$ /$2;

}

# BOSH for subdomains

location ~ ^/([^/?&:'"]+)/http-bind {

set $subdomain "$1.";

set $subdir "$1/";

set $prefix "$1";

rewrite ^/(.\*)$ /http-bind;

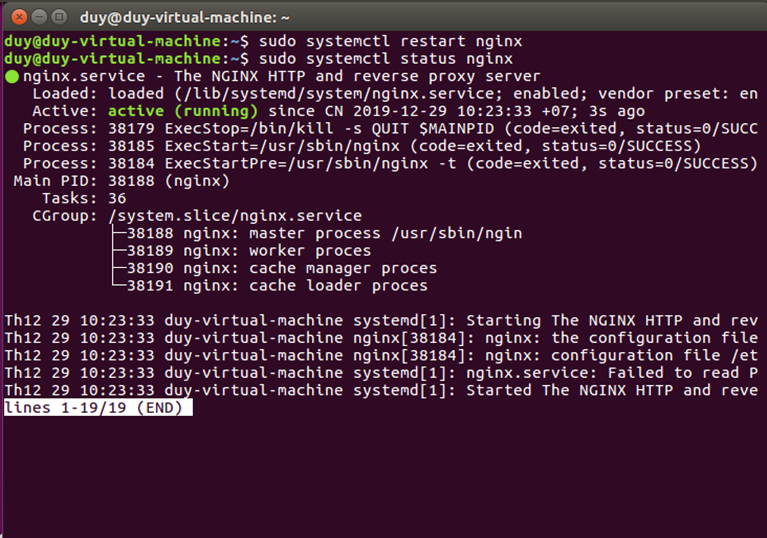
}

}

Trong đó có một số trường quan trọng như:

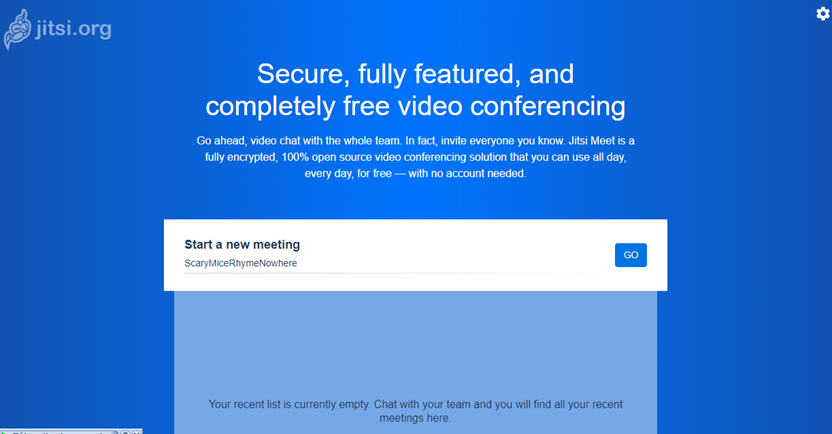
* listen 443 ssl: trường này sẽ cấu hình nginx chạy Jitsi trên HTTPS.
* server\_name public-ip: đây là trường để tên miền mà Jitsi sẽ dùng, vì chúng em sử dụng public ip nên chúng em sẽ để public ip của máy ảo tại đây.
* ssl\_protocols: phương thức SSL sẽ cấu hình loại phương thức SSL nào để nginx áp dụng lên Jitsi. Tùy vào phương thức SSL được cấu hình mà nó sẽ hộ trợ các cipher suite khác nhau. Chúng em chọn TLSv1.2 và TLSv1.3 cho hệ thống Jitsi.
* ssl\_ciphers: Trường này sẽ qui định các cipher suite để áp dụng vào việc mã hóa dữ liệu đi ra từ server Jitsi. Trong đây, chúng em chọn cipher suite “ECDHE-RSA-CHACHA20-POLY1305” để làm cipher suite chính cho việc mã hóa Jitsi vì thuật toán mã hóa của nó là CHACHA20.

Sau khi hoàn tất chỉnh sửa tập tin cấu hình, chúng em lưu nó tại đường dẫn “/etc/nginx/sites-enabled” và “/etc/nginx/sites-available”. Đây là nơi lưu trữ các tập tin cấu hình của nginx. Jitsi sẽ được quản lý bởi nginx sau khi nginx được khởi động lại nhưng trước khi khởi động lại, chúng em dừng các quá trình đang chạy trên cổng 80 và cổng 443 để tránh bị trùng lặp với nginx bằng hai câu lệnh sudo kill -9 `sudo lsof -t -i:80` và sudo kill -9 `sudo lsof -t -i:443`. Khởi động lại nginx với lệnh sudo systemctl restart nginx sau đó kiểm tra xem trạng thái của nginx với lệnh sudo systemctl status nginx.



* + - * 1. Nginx được khởi động lại thành công

Việc triển khai hệ thống Jitsi đến đây là hoàn tất, để kiểm tra hệ thống chúng em sử dụng một máy tính khác để vào mạng LAN và kết nối tới hệ thống thông qua public ip như đã cấu hình ở trên.



* + - * 1. Giao diện trang chủ hệ thống Jitsi
  1. Hệ thống Jitsi với mã hóa RC4

Thay đổi lớn nhất trong việc cài đặt hệ thống Jitsi với mã hóa RC4 đó là việc sử dụng web server nginx tại nguồn thay vì việc sử dụng phiên bản biên dịch lại của nginx.

Sau khi thiết lập tường lửa, chúng em cập nhật hệ thống và cài đặt nginx:

sudo apt-get update

sudo apt-get install nginx

Câu lệnh trên sẽ cài đặt phiên bản nginx 1.10.1 cùng với OpenSSL 1.0.2g được tích hợp sẵn. Sau khi cài đặt Jitsi lên máy ảo, chúng em chỉnh sửa phần ssl\_ciphers của tập tin cấu hình như sau:

ssl\_protocols TLSv1.2 TLSv1.3;

ssl\_prefer\_server\_ciphers on;

ssl\_ciphers " ECDH-RSA-RC4-SHA";

Nginx sẽ sử dụng cipher suite “ECDH-RSA-RC4-SHA” chứa RC4 để mã hóa các dữ liệu đi ra từ Jitsi.

* 1. Kiểm tra sự kháng chịu với các tấn công mã hóa khác nhau

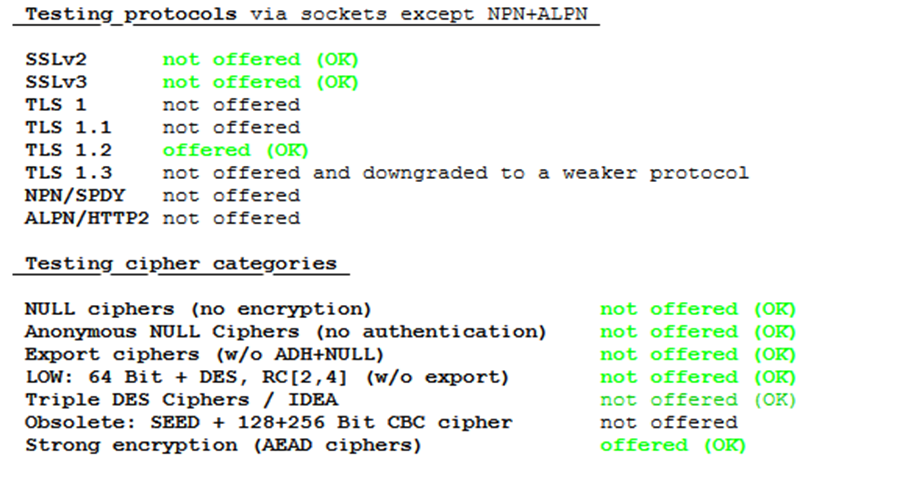
Để kiểm tra được khả năng kháng chịu của hệ thống với hai mã hóa RC4 và CHACHA20, chúng sử dụng một công cụ dòng lệnh gọi là testssl.sh. testssl.sh là một công cụ dòng lệnh giàu tính năng và miễn phí được sử dụng để kiểm tra các dịch vụ kích hoạt mã hóa TLS / SSL cho các mật mã, giao thức và một số lỗ hổng mã hóa được hỗ trợ, trên các máy chủ Linux / BSD.

Một số tính năng chủ yếu của testssl.sh như sau:

* Kết quả rõ ràng: kết quả có thể dễ dàng giúp người dùng biết liệu mọi thứ là tốt hay xấu.
* Dễ cài đặt: Nó hoạt động cho Linux, Mac OSX, FreeBSD và WSL / MSYS2 / Cygwin.
* Tính linh hoạt: Bạn có thể kiểm tra mọi dịch vụ SSL / TLS được kích hoạt và STARTTLS, không chỉ các máy chủ web tại cổng 443.
* Độ tin cậy: các tính năng được kiểm tra kỹ lưỡng.
* Quyền riêng tư: Chỉ có bạn là người nhìn thấy kết quả chứ không phải bên thứ ba.

Sau khi cài đặt testssl.sh, chúng em chạy lệnh sudo./testssl.sh --html public-ip để tiến hành kiểm tra hệ thống thông qua public ip của hệ thống. Sau khi testssl.sh chạy thử kiểm tra hệ thống, nó sẽ in ra một file log html như trên câu lệnh. Một số kiểm tra của testsslsh như sau:

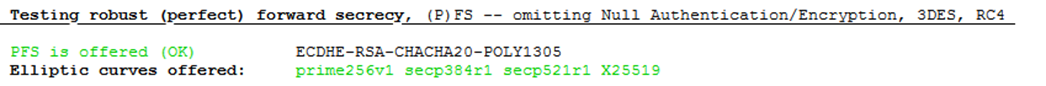
* + 1. Kiểm tra trên hệ thống sử dụng cipher Chacha20



* + - * 1. Thử nghiệm giao thức và các cipher

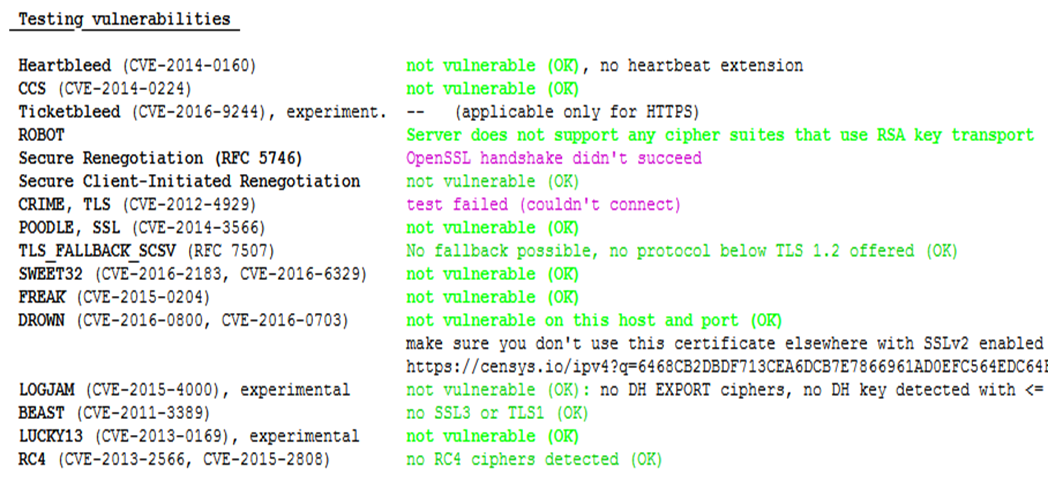
Testssl.sh thử nghiệm các giao thức thông qua các socket và cho thấy rằng hệ thống có hỗ trợ các giao thức bảo mật (TLSv1.2) và không hỗ trợ các giao thức bảo mật đã cũ và không có đạt được độ tin cậy (SSLv2 và SSLv3). Điều này khiến cho hệ thống đạt được độ bảo mật cao hơn.

Việc thử nghiệm các cipher cho thấy các kết quả khả quăng trong việc lựa chọn cipher suit. Dựa trên bảng thử nghiệm, hệ thống có sử dụng thuật toán mã hóa (CHACHA20), hệ thống có sử dụng thuật toán xác thực (POLY1305) và không sử dụng sử dụng các thuật toán có thể dễ dàng bị bẻ khóa như các Export ciphers, DES, RC2, RC4, 3DES. Không chỉ vậy, hệ thống sử dụng loại mã hóa xác thực với dữ liệu liên quan (CHACHA20 – POLY1305) khiến cho dữ liệu truyền đi từ hệ thống vừa được bảo mật bởi mã hóa vừa có tính xác thực cao.



* + - * 1. Thử nghiệm chuyển tiếp bí mật

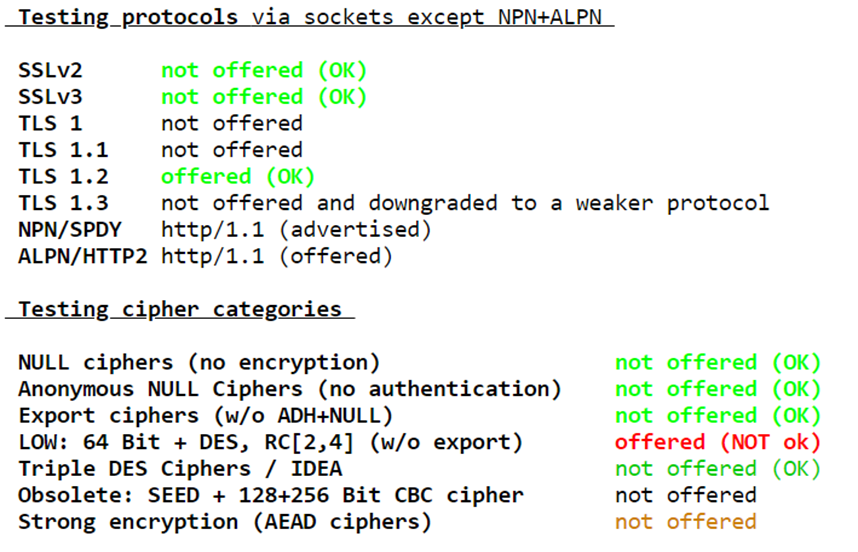
Thông qua cipher suite “ECDHE-RSA-CHACHA20-POLY1305”, hệ thống cũng đã đạt được chuyển tiếp bí mật (forward secrecy) và có hỗ trợ các loại thuật toán mật mã đường cong elip: prime256v1 secp384r1 secp521r1 X25519.



* + - * 1. Thử nghiệm các điểm yếu

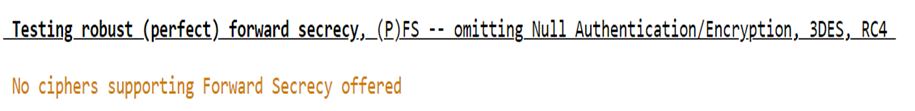
Dựa qua các thử nghiệm của testssl.sh, hệ thống đạt được độ bảo mật cao vì nó không có điểm yếu với các cuộc tấn công thường thấy (Heartbleed, BEAST, FREAK, LOGJAM).

* + 1. Kiểm tra trên hệ thống sử dụng cipher RC4



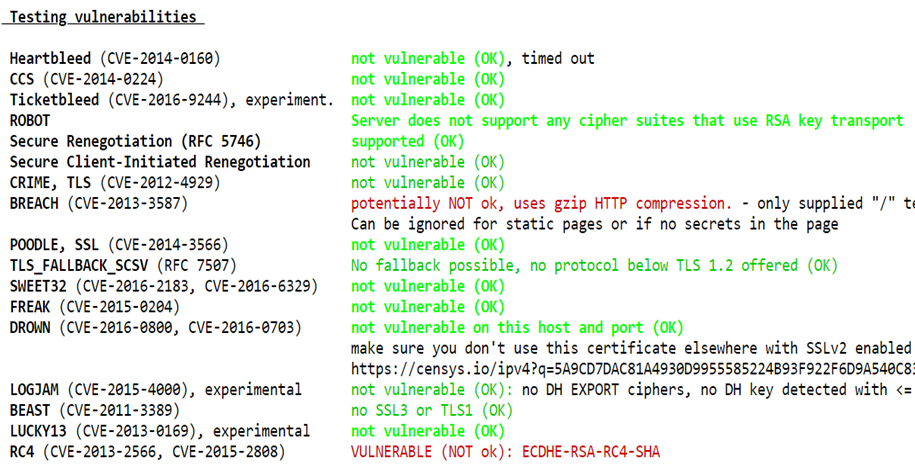
* + - * 1. Thử nghiệm giao thức và các cipher trên RC4

Với thử nghiệm giao thức và cipher, điều khác biệt nhất đó là hệ thống sử dụng cipher để mã hóa dữ liệu, điều này khiến cho đánh giá cipher của testssl.sh không chấp nhận. Hơn nữa, cipher RC4 không được công nhận là một AEAD cipher nên nó khiến cho việc bảo mật của hệ thống bị giảm xuống.



* + - * 1. Thử nghiệm chuyển tiếp bí mật của hệ thống RC4

Ngoài việc không được công nhận là một AEAD cipher, RC4 cũng không đạt được chuẩn chuyển tiếp bí mật được đặt ra cũng như không hỗ trợ các loại thuật toán mật mã đường cong elip nào.



* + - * 1. Thử nghiệm các điểm yếu của hệ thống RC4

Mặc dù hệ thống có khả năng bảo mật tốt trước các điểm yếu nhưng nó vẫn gặp điểm yếu như là tấn công BREACH do sử dụng HTTP compression và điểm yếu lớn nhất của hệ thống là sử dụng phương thức mã hóa dữ liệu RC4 bởi vì RC4 có thể dễ dạng bị các bên thứ ba khai thác để tấn công vào hệ thống.

1. KẾT QUẢ THỰC NGHIỆM

Ở phần này, chúng em sẽ trình bày sự tấn công của Fluhrer, Mantin và Shamir trên mạng WEP bằng ngôn ngữ lập trình python của người dùng jackieden26 trên Github.

* 1. Bảo mật bằng WEP

WEP (Wired Equivalent Privacy) là một thuật toán nhằm bảo vệ sự trao đổi thông tin chống lại sự nghe trộm, chống lại những nối kết mạng không được cho phép cũng như chống lại việc thay đổi hoặc làm nhiễu thông tin truyền. WEP sử dụng stream cipher RC4 cùng với một mã 40 bit và một số ngẫu nhiên 24 bit (initialization vector – IV) hoặc một mã 104 bit và 1 số ngẫu nhiên 24 bit để mã hóa thông tin. Thông tin mã hóa được tạo ra bằng cách thực hiện operation XOR giữa keystream và plain text. Thông tin mã hóa và IV sẽ được gửi đến người nhận. Người nhận sẽ giải mã thông tin dựa vào IV và khóa WEP đã biết trước.

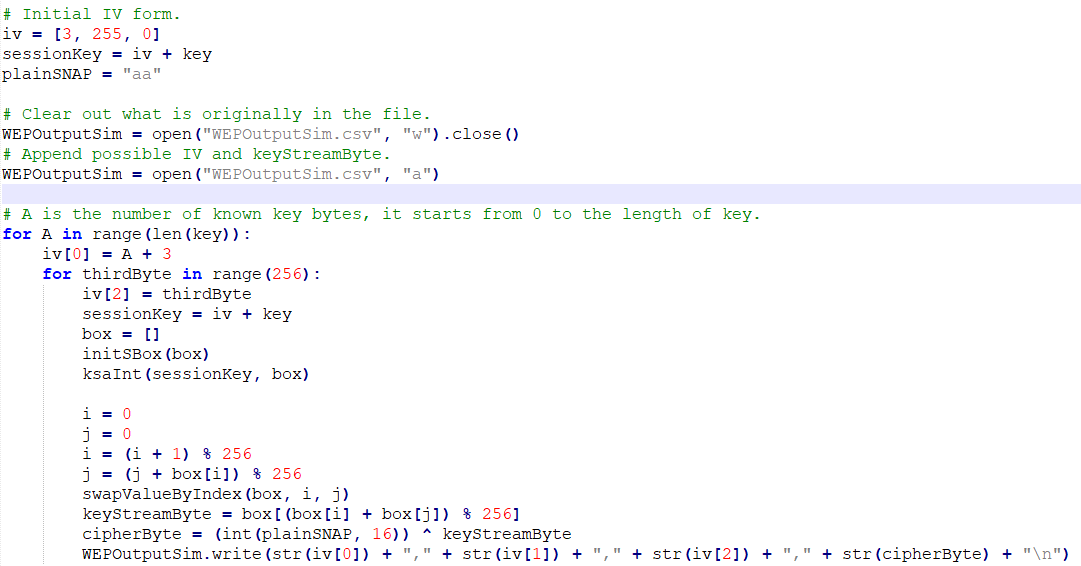
* 1. Tấn công WEP

Trong WEP, IV luôn được thêm vào khóa đầu vào của người dùng. Do khóa đầu vào của người dùng rất có thể vẫn giữ nguyên cho cùng một mạng, kẻ tấn công có thể nghe lén rất nhiều gói và dựa trên tiêu đề IV và SNAP đã biết, khóa bí mật có thể được phục hồi. Do đó, không sử dụng mã hóa WEP nữa.

Cấu trúc của IV yếu nhất có dạng (A + 3, N – 1, V) trong đó A là chỉ số của byte khóa chia sẻ và V có thể là bất kỳ số nào. Khi IV được sử dụng, có thể khôi phục đầu vào của RC4, biết đầu ra của byte đầu tiên.

Vì RC4 là một mật mã luồng, kẻ tấn công có thể sửa đổi các bit bản rõ qua bản mã và sửa tổng kiểm tra CRC tương ứng. Trong đó một phần của bản rõ có thể dự đoán được, kẻ tấn công xem trộm được gói và thay đổi địa chỉ IP của nó sang máy của người đó từ bản mã. Do đó, kẻ tấn công có được văn bản được giải mã mà không phá vỡ mã hóa.

Chúng em sử dụng hai file WEPOutput.py và keyRecover.py để tấn công WEP khi sử dụng thuật toán RC4.



* + - * 1. Khởi tạo giá trị IV

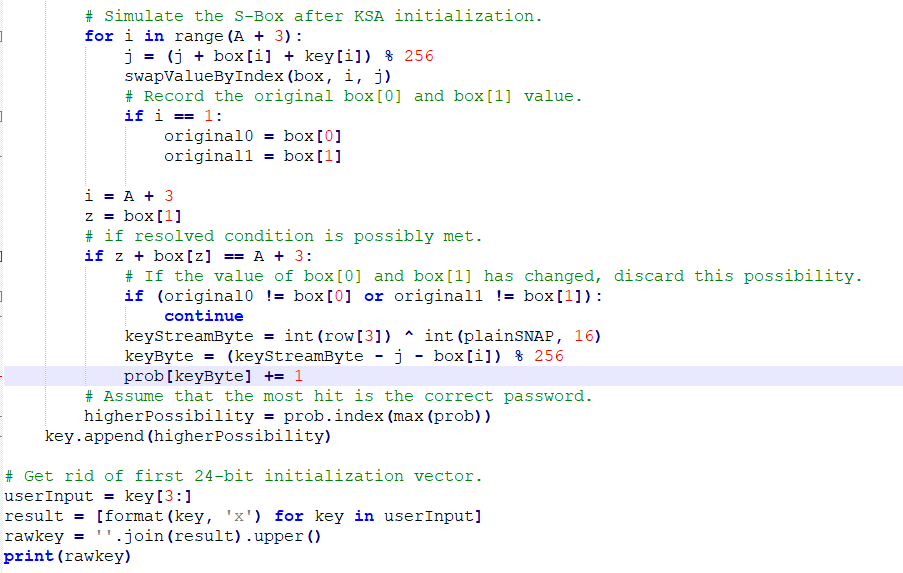
Trong môi trường thực tế, các nhà tấn công sẽ theo dõi và lấy các gói tin để thu thập các thông tin như sau: IV, dòng dữ liệu đã được mã hóa, byte đầu tiên của SNAP header của WEP.

Trong phần demo, chúng em giả định rằng những thông tin này có sẵn cho người tấn công. Ở file WEPOutput.py để tạo key WEP do người dùng nhập và khởi tạo IV. Ngoài ra file WEPOutput còn sinh ra file WEPOutputSim.csv, file này lưu trữ thông tin bao gồm IV ở dạng: (A + 3, 255, V) và byte đầu tiên của SNAP header của WEP. Nó sẽ được người tấn công sử dụng để tìm ra key gốc do người dùng nhập vào.



* + - * 1. Nhập key cho WEP và xuất file WEPOutputSim.csv

Sau đó, sử dụng keyRecovery.py cùng với file WEPOutput.py chúng em lấy được bộ IV với giá trị key gốc.



* + - * 1. Thuật toán tìm key gốc

Khi loại bỏ giá trị IV dư thừa thu được key gốc do người dung nhập vào như trong file WEPOutput.py.



* + - * 1. Key gốc được tìm bởi người tấn công
  1. Kiểm tra hiệu suất của thuật toán

Trong phần này, để kiểm tra hiệu suất thuật toán chúng em tham khảo code thuật toán RC4 của Thimo Kraemer và Chacha20 của Péter Szabó dựa vào ngôn ngữ python. Mã nguồn chúng em lấy trên Github của hai tác giả.

Việc kiểm tra hiệu suất của thuật toán chúng em sẽ chạy trên máy có các thông số sau:

Operating system: Windows 10 Pro 64 – bit (10.0, Build 18362).

Processor: Intel Core(TM) i7 – 6500U CPU @ 2.50GHz (4 CPUs), ~2.6GHz

Memory: 8GB

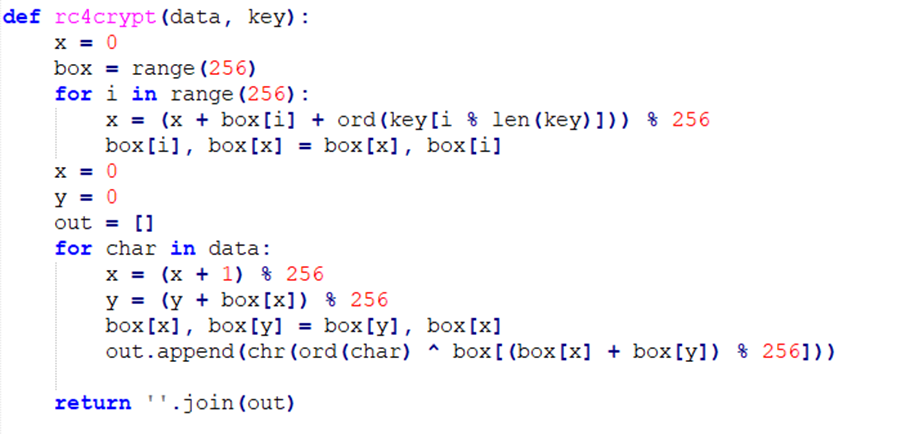
Đầu tiên, chúng em sẽ tính throughput của quá trình mã hóa của 10 dữ liệu khác nhau.

Công thức tính throughput:

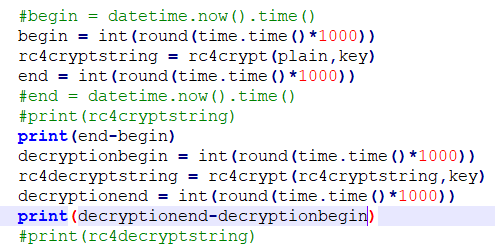
Throughput = size (plaint.txt) / (Thời gian kết thúc – thời gian bắt đầu).

Lấy kết quả tính throughput của 10 dữ liệu sau đó chúng em tính giá trị thoughput trung bình của dữ liệu đó. (Công thức tính throughput của quá trình mã hóa và giải mã giống nhau).

Trong file rc4.py, chúng em xây dựng hàm có tên rc4crypt có input đầu vào là biến data và biến key. Hàm này là chuỗi plaintext cùng với secret key người dùng tự đặt. Với 2 input đó, sau đó sử dụng rc4 để áp dụng vào việc mã hóa, đầu ra của thuật toán là chuỗi ciphertext.



* + - * 1. Áp dụng thuật toán RC4

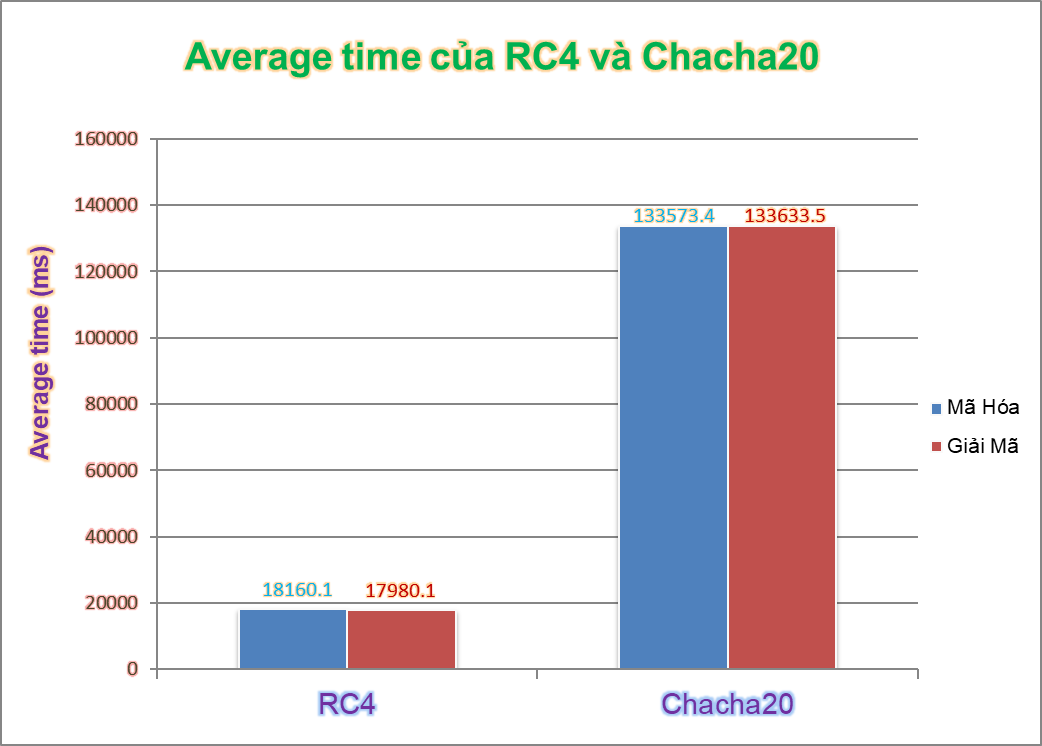


* + - * 1. Hàm tính thời gian bắt đầu và kết thúc mã hóa, giải mã

Dưới đây là kết quả thời gian thực thi trung bình của chiều dài khóa 256 bit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Text file size in Kbytes | RC4 | | Chacha20 | |
|  | Mã hóa | Giải mã | Mã hóa | Giải mã |
| 7724 | 4244 | 4288 | 30951 | 31920 |
| 14181 | 7627 | 7556 | 56862 | 59755 |
| 14216 | 7647 | 7626 | 56920 | 55271 |
| 23172 | 12457 | 12403 | 113619 | 99936 |
| 28361 | 15346 | 15255 | 113991 | 113778 |
| 28370 | 15149 | 15304 | 112129 | 111139 |
| 38620 | 20821 | 21306 | 149962 | 149585 |
| 42584 | 23141 | 22937 | 165711 | 163470 |
| 46341 | 25462 | 25033 | 181916 | 192418 |
| 88927 | 49707 | 48093 | 353673 | 359063 |
| Average time (ms) | 18160,1 | 17980,1 | 133573,4 | 133633,5 |

Thời gian thực thi trung bình của RC4 và Chacha20

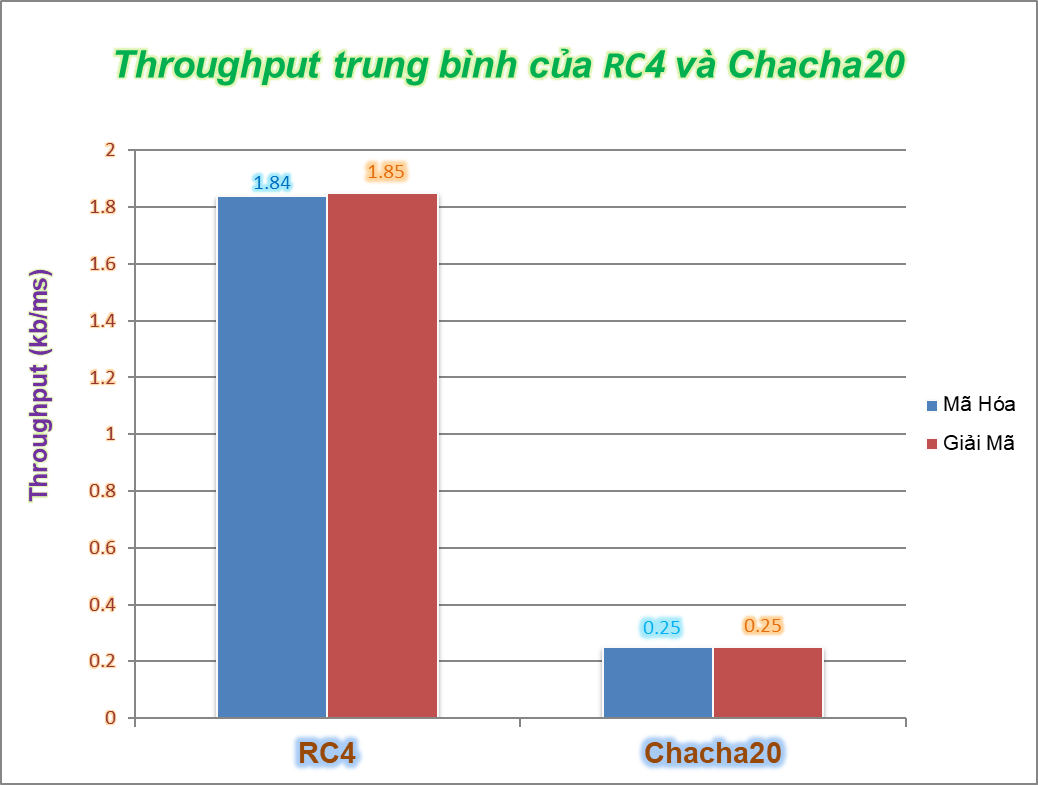


* + - * 1. Biểu đồ thể hiện giá trị thời gian trung bình của RC4 và Chacha20

Giá trị throughput trung bình của chiều dài khóa 256 bit.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Text file size in Kbytes | RC4 | | Chacha20 | |
|  | Mã hóa | Giải mã | Mã hóa | Giải mã |
| 7724 | 1,82 | 1,80 | 0,25 | 0,24 |
| 14181 | 1,86 | 1,88 | 0,25 | 0,24 |
| 14216 | 1,86 | 1,86 | 0,25 | 0,26 |
| 23172 | 1,86 | 1,87 | 0,2 | 0,23 |
| 28361 | 1,85 | 1,86 | 0,25 | 0,25 |
| 28370 | 1,87 | 1,85 | 0,25 | 0,26 |
| 38620 | 1,85 | 1,81 | 0,26 | 0,26 |
| 42584 | 1,84 | 1,86 | 0,26 | 0,26 |
| 46341 | 1,82 | 1,85 | 0,25 | 0,24 |
| 88927 | 1,79 | 1,85 | 0,25 | 0,25 |
| Throughput (Kb/ms) | 1,84 | 1,85 | 0,25 | 0,25 |

Throughput trung bình của RC4 và Chacha20



* + - * 1. Biểu đồ thể hiện giá trị throughput trung bình của RC4 và Chacha20

Qua bảng tính giá trị throughput trung bình và biểu đồ đã vẽ ở trên thì chúng em thấy tốc độ throughput của thuật toán Chacha20 nhanh hơn thuật toán RC4 và giá trị test của quá trình mã hóa và giải mã trong từng thuật toán cũng xấp xỉ nhau. (Có thể do tốc độ xử lý của máy).

1. KẾT LUẬN

Sau khi tìm hiểu và xây dựng hệ thống mã hóa dòng, chúng em thấy một trong những cách phổ biến nhất để áp dụng dữ liệu dòng đó là thông qua streaming video call trực tuyến. Hệ streaming video call như hệ thống Jitsi có thể dễ dàng thiết lập và cài đặt trong các môi trường cộng đồng có thể giúp kết nối, giao tiếp và hỗ trợ các cá nhân trong môi trường đó. Không chỉ vậy, vì hệ thống Jitsi được tạo thành từ các phần mềm mã nguồn mở nên nó có thể được tích hợp vào các hệ thống hoặc công nghệ khác để có thể nâng cao tính phục vụ cho người dùng. Nhưng việc xây dựng và áp dụng hệ thống phải đi đôi với việc bảo mật chúng. Việc mã hóa dòng RC4 trở thành vô hiệu hóa sau quá nhiều lỗ hổng an ninh đã khiến cho các nhà phát triển phải tìm đến các phương thức mã hóa khác để bảo mật cho dữ liệu của mình. Salsa20/CHACHA20 là một thuật toán mã hóa dòng chỉ mới xuất hiện trong thời gian gần đây tuy nhiên Salsa20/CHACHA20 là một trong những ứng cử viên của dự án eSTREAM và vượt qua nhiều kiểm tra bảo mật của các nhà mật mã học, chúng em tin rằng nó có thể trở thành một khuôn mẫu cho các loại mã hóa dòng trong tương lai sắp tới. Ngoài ra, chúng em còn đánh giá được độ an toàn và test những số liệu thực tế để đo hiệu suất của hệ thống khi sử dụng các thuật toán.

Bên cạnh những việc làm được chúng em còn gặp những hạn chế trong phần cài đặt và thực nghiệm như:

Hệ thống Jitsi vẫn chưa được tùy chỉnh hoàn thiện về hình thức và chức năng, chưa hỗ trợ trên nền tảng di động và chưa áp dụng được các phương thức state – of – the – art. Môi trường thực nghiệm và code áp dụng thuật toán vẫn chưa được tối ưu.

TÀI LIỆU THAM KHẢO

1. S.Fluthrer, I. Mantin, A. Shamir, “Weaknesses in the Key Scheduling Algorithm of RC4”, SAC2001 (S. Vaudenay, A.Youssef,eds.), col. 2259 of LNCS, pp. 1-24, springerVerlag, 2001. Adam Stubblefield, John Ioannidis, Aviel D. Rubin “A Key Recovery Attack on the 802.11b Wired Equivalent Privacy Protocol (WEP)”
2. T.D.B Weerasinghe, Member IEEE, “An effective RC4 stream cipher”
3. Maytham M. Hammood, Kenji Yoshigoe and Ali M. Sagheer, RC4 Stream Cipher with Two State Tables
4. Eli Biham, Orr Dunkelman, Differential Cryptanalysis in Stream Ciphers, Computer Science Department, Technion, Isarel.
5. Manju Kumari, Vipin Pawar and Pawan Kumar, A Novel Image Encryption Scheme With Huffman Encoding And Steganography Technique
6. Vincent Rijmen (2010), Stream Ciphers and the eSTREAM Project?, Graz University of Technology, Austria
7. Roel Verdult (2015), Introduction to Cryptanalysis: Attacking Stream Ciphers, Radboud University Nijmegen, The Netherlands.
8. Rajeev Sobti, Geetha Ganesan (2016), Analysis of Quarter Rounds of Salsa and ChaCha Core and Proposal of an Alternative Design to Maximize Diffusion, lovely Professional University, Phagwara, India.
9. Rick Wash, Lecture Notes on Stream Ciphers and RC4.
10. Charalampos Manifavas, George Hatzivasilis, Konstantinos Fysarakis, Yannis Papaefstathiou (2015), A survey of lightweight stream ciphers for embedded systems, George Hatzivasilis, Dept. of Electronic and Computer Engineering, Technical University of Crete, Akrotiri Campus, 73100 Chania, Crete, Greece.
11. William Stallings (2005), The RC4 Stream Encryption Algorithm.
12. Daniel J.Bernstein (2007), The Salsa20 family of stream ciphers, The University of Illinois, Chicago.
13. Daniel J. Bernstein (2008), ChaCha, a variant of Salsa20, The University of Illinois, Chicago.
14. Scott Fluhrer, Itsik Mantin, Adi Shamir (2001), Weaknesses in the Key Scheduling Algorithm of RC4, the Weizmann Institute, Isarel.
15. jitsi-meet, <https://github.com/jitsi/jitsi-meet>
16. John Carl Villanueva (2016), An Introduction To Cipher Suites | SSL/TSL Cipher Suites Explained, <https://www.jscape.com/blog/cipher-suites>
17. Ivan Ristić (2016), OpenSSL Cookbook, feistyduck.
18. Remy van Elst (2019), Strong SSL Security on nginx, <https://raymii.org/s/tutorials/Strong_SSL_Security_On_nginx.html#toc_14>
19. <https://www.revolvy.com/page/Bit%252Dflipping-attack>
20. <https://www.revolvy.com/page/RC4?stype=videos&cmd=list>