ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

LÊ THỊ LEN

MẬT MÃ DÒNG TRONG MẬT MÃ NHẸ VÀ TRIỂN VỌNG TRONG IoT

LUẬN VĂN THẠC SĨ

Ngành: Hệ thống thông tin

HÀ NỘI - 2017

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

LÊ THỊ LEN

MẬT MÃ DÒNG TRONG MẬT MÃ NHẸ VÀ TRIỂN VỌNG TRONG IoT

Ngành: Hệ thống thông tin

Chuyên ngành: Hệ thống thông tin

Mã số: 60480104

LUẬN VĂN THẠC SĨ
Ngành: Hệ thống thông tin

NGƯỜI HƯỚNG DẪN KHOA HỌC

TS. Lê Phê Đô
TS. Phùng Văn Ổn

HÀ NỘI - 2017

TÓM TẮT

Tóm tắt: Cùng với sự phát triển của tính toán khắp nơi, các hệ thống vạn vật kết nối (Internet of Things – IoT) ngày càng thu hút được sự quan tâm của các chuyên gia cũng như các nhà ứng dụng. Vấn đề an ninh an toàn thông tin trong các hệ thống IoT với các thiết bị nhỏ gọn, năng lực tính toán thấp, trở thành một chủ đề nóng hiện nay. Với khả năng tính toán nhanh, an toàn và chi phí thực hiện thấp, mật mã nhẹ, tiêu biểu là mật mã dòng, là sự lựa chọn tối ưu cho những thiết bị chuyên dụng của IoT. Luận văn nghiên cứu khả năng ứng dụng, điều kiện áp dụng cũng như yêu cầu của một số giải thuật mật mã nhẹ, đề xuất phương án sử dụng mật mã dòng trong mật mã nhẹ phù hợp với từng lớp bài toán cụ thể. Đi vào thực nghiệm luận văn đề xuất sử dụng mã hóa đầu cuối với mật mã dòng Grain trong mật mã nhẹ và mã xác thực thông báo với hàm băm nhẹ Keccak trên thiết bị Raspberry Pi để thu thập, điều khiển nhiệt độ, độ ẩm, cửa ra vào trong một ngôi nhà – tiền đề cho những nghiên cứu về bảo mật trong mô hình smart home nói riêng và các mô hình IoT nói chung.

Từ khóa: *vạn vật kết nối, mật mã dòng, Grain, mật mã nhẹ, mã xác thực thông báo, mã hóa đầu cuối, hàm băm Keccak.*

LỜI CẢM ƠN

Tôi xin chân thành cảm ơn Khoa Công nghệ thông tin, Trường Đại học Công nghệ đã tạo điều kiện, môi trường thuận lợi cho học viên trong quá trình học tập, nghiên cứu và hoàn thành luận văn thạc sĩ.

Với lòng biết ơn sâu sắc nhất, tôi xin gửi đến tiến sĩ Lê Phê Đô, Trường Đại học Công nghệ, ĐHQG Hà Nội, cùng tiến sĩ Phùng Văn Ổn, Văn phòng Chính phủ đã tận tâm hướng dẫn tôi qua từng buổi học trên lớp cũng như các buổi nói chuyện, thảo luận về đề tài nghiên cứu. Nếu không có sự định hướng, những lời dạy bảo của thầy thì luận văn này của tôi rất khó có thể hoàn thiện được. Một lần nữa, tôi xin chân thành cảm ơn thầy.

Tôi xin bày tỏ lòng kính trọng và biết ơn sâu sắc tới gia đình và các thầy, cô trong Trường Đại học Công nghệ, ĐHQG Hà Nội, các anh chị trong nhóm Nghiên cứu KH & CN Mật Mã UET-CRYPT, Trường Đại học Công nghệ - ĐHQG HN và bạn Đỗ Công Thành – học viên cao học Trường Đại học Công nghệ khóa K23, những người đã dạy dỗ, giúp đỡ tôi trong suốt quá trình học tập và thực hiện luận văn này.

Bước đầu đi vào thực tế, tìm hiểu về lĩnh vực chuyên sâu trong An toàn thông tin, kiến thức của tôi còn hạn chế và còn nhiều bỡ ngỡ. Do vậy, không tránh khỏi những thiếu sót trong luận văn. Tôi rất mong nhận được những ý kiến đóng góp quý báu của thầy cô và các bạn để hoàn thiện luận văn hơn nữa.

Tôi xin chân thành cảm ơn.

LỜI CAM ĐOAN

Tôi xin cam đoan các kết quả đạt được trong luận văn này do tôi thực hiện dưới sự hướng dẫn của Tiến sĩ Lê Phê Đô và Tiến sĩ Phùng Văn Ổn.

Tất cả các tham khảo từ những nghiên cứu liên quan đều được trích dẫn nguồn gốc một cách rõ ràng từ danh mục tài liệu tham khảo trong luận văn. Luận văn không sao chép tài liệu, công trình nghiên cứu của người khác mà không chỉ rõ về mặt tài liệu tham khảo. Các kết quả thực tế của luận văn đều được tiến hành thực nghiệm.

Nếu phát hiện có bất kỳ sự gian lận nào, tôi xin hoàn toàn chịu trách nhiệm trước hội đồng, cũng như kết quả luận văn tốt nghiệp của mình.

Hà Nội, ngày tháng năm ,

Học viên thực hiện

Lê Thị Len

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC TỪ VIẾT TẮT

Ký hiệu	Dạng đầy đủ
IoT	Internet of Things – Vạn vật kết nối – Một kịch bản của thế giới, khi mà mỗi đồ vật, con người được cung cấp một định danh của riêng mình, và tất cả có khả năng truyền tải, trao đổi thông tin, dữ liệu qua một mạng duy nhất mà không cần đến sự tương tác trực tiếp giữa người với người, hay người với máy tính
NIST	(National Institute of Standards and Technology) Viện tiêu chuẩn và Công nghệ của Mỹ
RFID	Radio Frequency Identification – Hệ thống nhận dạng bằng tần số của sóng vô tuyến
SHT11	Cảm biến đo nhiệt độ và độ ẩm. Đây là dòng cảm biến (SHT10, SHT11, SHT15, SHT75) chuyên dùng, có độ chính xác cao. Nó được sử dụng rộng dãi trong công nghiệp và dân dụng.
HTML5	Một chuẩn mới và là thế hệ tiếp theo của ngôn ngữ đánh dấu siêu văn bản - HyperText Markup Language explained (gọi tắt là HTML)
NFSR	Nonlinear feedback shift register – Thanh ghi dịch hồi phi tuyến – một thanh ghi dịch hồi với bit đầu vào là một hàm phi tuyến tính của trạng thái trước đó của nó. $r_{i+1}(b_0, b_1, b_2, \dots, b_{n-1}) = r_i(b_1, b_2, \dots, f(b_0, b_1, b_2, \dots, b_{n-1}))$
LFSR	Linear feedback shift register – Thanh ghi dịch hồi tuyến tính – một thanh ghi dịch hồi với bit đầu vào là một hàm tuyến tính của trạng thái trước đó của nó. Hàm tuyến tính được dùng phổ biến nhất là XOR.
MAC	(Message authentication code) Mã xác thực thông báo
HMAC	(Hash Messasge Authentication Code) Mã xác thực thông báo sử dụng hàm băm
CMAC	(Cipher Message Authentication Code) Mã xác thực thông báo dựa trên mã hóa
GE	(Gate equivalence) Cổng tương đương - một đơn vị đo lường cho phép xác định độ phức tạp độc lập về công nghệ sản xuất của các mạch điện tử kỹ thuật số

MỞ ĐẦU

1. Cơ sở khoa học và thực tiễn của đề tài:

Cùng với sự phát triển theo hàm số mũ của cuộc cách mạng công nghiệp lần thứ 4, vấn đề an ninh an toàn thông tin ngày càng được quan tâm và trở thành một trong những ưu tiên hàng đầu trong nghiên cứu cũng như ứng dụng. Để đảm bảo sự an toàn, bí mật cho thông tin, đa số các thiết bị thông tin hiện nay đã trang bị những phương pháp phòng ngừa, bảo vệ hiệu quả. Tiêu biểu như các phương pháp mã hóa truyền thống được triển khai phổ biến trên các thiết bị có cấu hình cao. Tuy nhiên, với những thiết bị nhỏ, kích thước, năng lượng và khả năng tính toán hạn chế, việc sử dụng các mã hóa truyền thống là không khả thi, cần có một giải thuật khác phù hợp hơn. Đây chính là vùng đất vàng cho mật mã nhẹ phát triển.

Tùy từng yêu cầu bảo mật cụ thể của từng thiết bị mà chúng ta có thể áp dụng các giải thuật mã hóa khác nhau để có thể cân đối giữa ba tiêu chí quan trọng của mật mã nhẹ: độ an toàn, hiệu suất và giá thành. Luận văn nghiên cứu khả năng ứng dụng, điều kiện áp dụng cũng như yêu cầu của một số giải thuật mã hóa nhẹ cụ thể, đề xuất phương án sử dụng mật mã nhẹ, tiêu biểu là mật mã dòng phù hợp cho những thiết bị nhỏ gọn, năng lực tính toán thấp, nhất là trong môi trường Internet of Thing (IoT).

Một trong những thiết bị nhỏ gọn có khả năng tính toán thấp nhưng lại có vai trò quan trọng trong nhiều hệ thống IoT đó là Raspberry Pi [Phụ lục A]. Raspberry Pi là một vi máy tính được phát triển bởi Raspberry Pi Foundation tại Anh. Phần cứng Raspberry Pi có nhiều cấu hình khác nhau, dung lượng bộ nhớ và các thiết bị ngoại vi khác nhau tùy từng phiên bản. Hiện nay, Raspberry Pi đang được sử dụng một cách rộng rãi trong các ứng dụng IoT. Theo đánh giá từ các nhà nghiên cứu, Raspberry Pi là một nền tảng phần cứng quan trọng để thực nghiệm cũng như ứng dụng cho các dự án IoT.

Với các thiết bị IoT này, có một vài mối nguy hiểm cần được quan tâm như vấn đề bảo mật. Vấn đề bảo mật trong một hệ thống IoT tiêu biểu hiện nay như smart home cũng có nhiều tầng, nhiều mức độ. Kẻ tấn công có thể đánh cắp thông tin cũng như chiếm quyền điều khiển đối với các sensor bằng việc can thiệp trực tiếp vào đường truyền vật lý giữa các sensor với các thiết bị thu nhận, phân tích, điều khiển dữ liệu như Raspberry Pi. Một cách tấn công khác, kẻ tấn công có thể tìm cách truy cập trực tiếp vào thiết bị máy chủ, ra lệnh giả điều khiển các thiết bị cảm biến, các đồ dùng, vật dụng trong nhà. Việc giao tiếp giữa các thiết bị giữa Raspberry và client side (điện thoại di động, máy tính bảng, laptop...) sẽ không còn là an toàn khi một ai đó xâm nhập vào hệ thống và lấy được các dữ liệu trên đường truyền dùng cho mục đích xấu. Ở mức cao hơn một chút, kẻ tấn công có thể chiếm quyền kiểm soát client side để trực tiếp điều khiển hệ thống smart home. Trong phạm vi nghiên cứu, luận văn chỉ tập trung vào bài toán an toàn thông tin trong quá trình giao tiếp giữa thiết bị Raspberry Pi

với các client side. Luận văn nghiên cứu và đề xuất sử dụng mã hóa đầu cuối với mật mã dòng trong mật mã nhẹ và mã xác thực thông báo trên thiết bị Raspberry Pi để thu thập, điều khiển nhiệt độ, độ ẩm, cửa ra vào trong một ngôi nhà – tiền đề cho những nghiên cứu về bảo mật trong mô hình smart home nói riêng và các mô hình IoT nói chung.

2. Nội dung của đề tài và các vấn đề cần giải quyết

2.1. Hướng nghiên cứu:

- Nghiên cứu mật mã nhẹ, mật mã dòng trong mật mã nhẹ.
- Khả năng ứng dụng mật mã dòng trong mật mã nhẹ trong IoT.
- Đề xuất xây dựng kênh truyền tin an toàn bằng phương pháp mã hóa đầu cuối sử dụng kỹ thuật mã hóa dòng Grain và xác thực thông báo với hàm băm Keccak trên Raspberry PI để điều khiển nhiệt độ, độ ẩm, cửa ra vào trong một ngôi nhà.
- Đánh giá hiệu quả của việc sử dụng mật mã dòng trong mật mã nhẹ trên Raspberry so với một số giải thuật mã hóa khác.

2.2. Nội dung nghiên cứu:

Ngoài phần mở đầu và kết luận, nội dung của luận văn được trình bày trong ba chương:

Chương 1: Giới thiệu tổng quan về mật mã nhẹ, mật mã dòng trong mật mã nhẹ, một số khái niệm quan trọng, tìm hiểu một số hệ mật mã dòng trong mật mã nhẹ phổ biến hiện nay và lợi ích cũng như vấn đề gặp phải khi ứng dụng mật mã dòng trong mật mã nhẹ trong thực tế mà tiêu biểu IoT.

Chương 2: Nghiên cứu và đánh giá về một họ hệ mật mã dòng tiêu biểu trong mật mã nhẹ – Grain và khả năng ứng dụng mật mã dòng nhẹ Grain trong IoT.

Chương 3: Thực nghiệm áp dụng mã hóa đầu cuối với mật mã Grain và mã xác thực thông báo với hàm băm nhẹ Keccak trong việc sử dụng thiết bị Rasberry để thu thập dữ liệu từ cảm biến SHT11 dùng để đo nhiệt độ, độ ẩm của phòng làm việc; qua đó trả lại thông tin cho người dùng thông qua giao diện Web HTML5. Người dùng có thể điều khiển các thiết bị trong phòng để thay đổi nhiệt độ, độ ẩm. Đồng thời đánh giá hiệu quả của việc sử dụng mật mã dòng trong mật mã nhẹ trên Raspberry so với một số giải thuật mã hóa khác.

3. Kết quả đạt được

Sau 6 tháng nghiên cứu, về lý thuyết, luận văn đã nghiên cứu, đánh giá được độ an toàn, hiệu suất sử dụng của hệ mật mã Grain nói riêng và hệ mật mã dòng trong mật mã nhẹ nói chung trong môi trường IoT. Về thực nghiệm, luận văn đã xây dựng thành công kênh truyền tin an toàn bằng phương pháp mã hóa đầu cuối sử dụng kỹ thuật mã hóa dòng Grain và xác thực thông báo với hàm băm Keccak trên Raspberry PI để điều khiển nhiệt độ, độ ẩm, cửa ra vào trong một ngôi nhà. Từ đó có những số liệu thực tế đánh giá hiệu quả, độ an toàn của mật mã dòng nhẹ Grain so với một số hệ mật mã nhẹ khác.

Chương 1. MẬT MÃ DÒNG TRONG MẬT MÃ NHẸ

1.1. Tổng quan về mật mã nhẹ

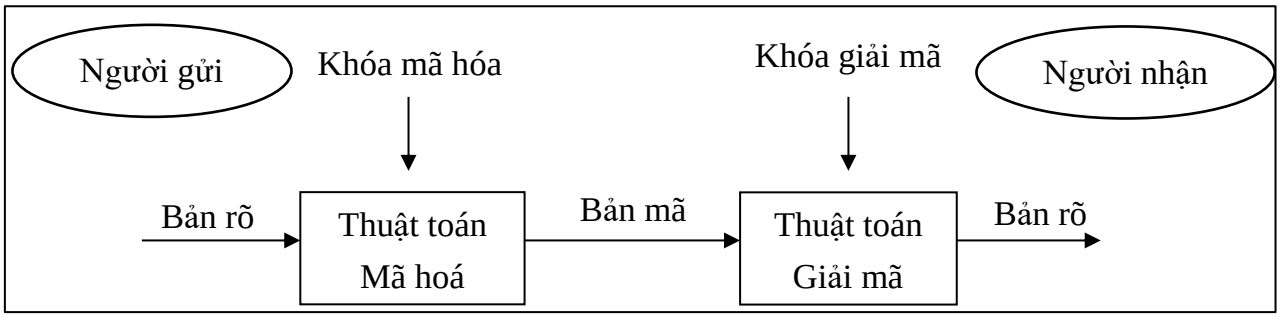
1.1.1. Một số khái niệm cơ bản

1.1.1.1. Hệ mật mã

Hệ mật mã [17]: là một hệ bao gồm 5 thành phần (P, C, K, E, D) thoả mãn các tính chất sau:

- P (Plaintext) là tập hợp hữu hạn các bản rõ có thể.
- C (Ciphertext) là tập hợp hữu hạn các bản mã có thể.
- K (Key) là tập hợp các bản khoá có thể.
- E (Encrytion) là tập hợp các qui tắc mã hoá có thể.
- D (Decrytion) là tập hợp các qui tắc giải mã có thể.

$$\mathbf{E_K(P) = C \text{ và } D_K(C) = P}$$



Hình 1-1: Sơ đồ hệ mật mã

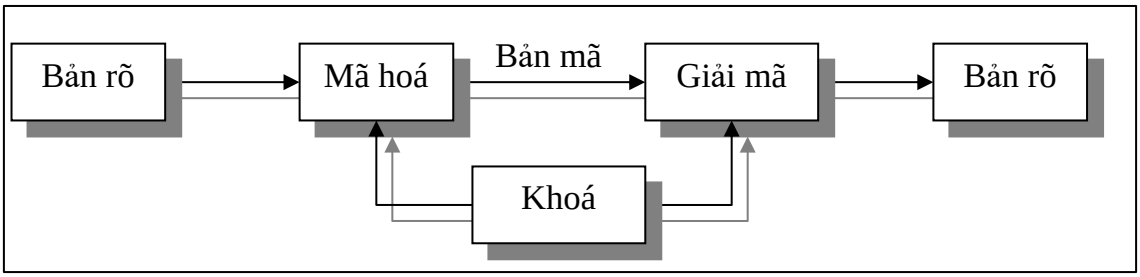
Những yêu cầu đối với hệ mật mã [18]

- ***Độ tin cậy***: bằng việc sử dụng các kỹ thuật mã hóa khác nhau, hệ mật giúp che giấu thông tin, đảm bảo sự bí mật cho các thông báo và dữ liệu được lưu trữ.
- ***Tính toàn vẹn***: cung cấp cơ chế đảm bảo thông báo không bị thay đổi trong quá trình truyền nhận.
- ***Tính không thể chối bỏ***: có thể cung cấp một cách xác nhận rằng tài liệu đã đến từ ai đó ngay cả khi họ cố gắng từ chối nó.
- ***Tính xác thực***: xác thực nguồn gốc của một thông báo và của người đang đăng nhập một hệ thống.

Ta có thể phân hệ mật mã thành hai loại dựa vào khóa:

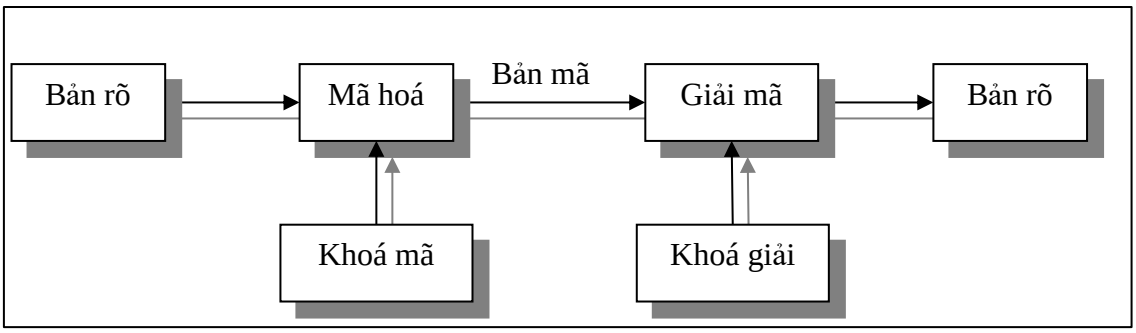
- Hệ mật mã khóa đối xứng (Symmetric cryptosystem): là một hệ mật mã khá lâu đời với *khoá mã hoá* có thể tính ra được *khoá giải mã*, hoặc khoá mã hoá và khoá giải mã là *giống nhau*. Với hệ mật mã khóa đối xứng, người gửi và người nhận phải thỏa thuận khóa bí mật trước khi thực hiện mã hóa. Độ an toàn của

thuật toán này phụ thuộc vào khoá. Nếu khóa bí mật bị tiết lộ thì bất kỳ ai cũng có thể giải mã thông báo trong hệ thống mã hoá.



Hình 1-2: Mã hoá với hệ mật mã khóa đối xứng

- Hệ mật mã bất đối xứng (Asymmetric cryptosystem): Diffie và Hellman đã phát minh ra *hệ mã hoá công khai* hay *hệ mã hoá phi đối xứng* từ những năm 1970. Hệ mã hoá công khai khác biệt so với hệ mật mã đối xứng ở khóa mã hóa và khóa giải mã. *Khoá mã hoá (khóa công khai)* hoàn toàn khác với *khoá giải mã (khóa bí mật)*. Đặc biệt, kẻ tấn công không thể tính toán được khóa giải mã kể cả khi biết khóa mã hóa. Khóa mã hóa có thể công khai. Bất kỳ ai cũng có thể sử dụng khoá công khai để mã hoá thông báo, nhưng chỉ người có khoá bí mật thì mới có khả năng giải mã được thông điệp. Khoá công khai và bản mã đều có thể gửi trên một kênh truyền không an toàn mà không làm mất độ an toàn của hệ mật.



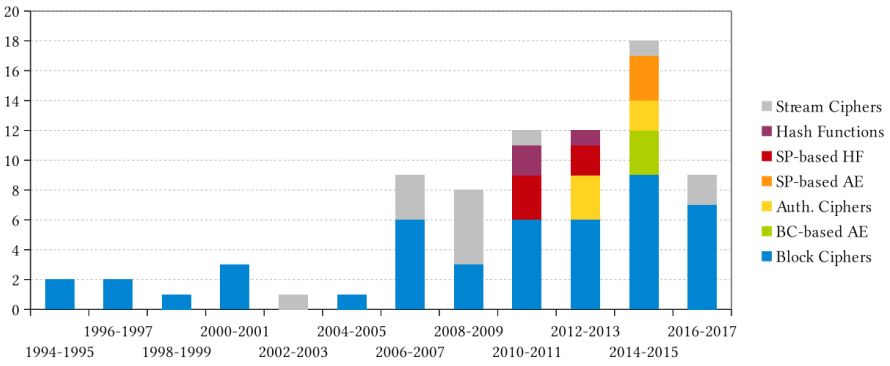
Hình 1-3: Mã hoá với hệ mật mã bất đối xứng

1.1.1.2. Khái niệm mật mã nhẹ

Hiện nay, chưa có một tổ chức nào đưa ra khái niệm chính xác hay định lượng cụ thể về mật mã nhẹ. Vì vậy có rất nhiều phiên bản để định nghĩa mật mã nhẹ. Một trong số đó là tiêu chuẩn ISO/IEC 29192-1 [1] đã đưa ra khái niệm cơ bản về mật mã nhẹ trong phần tổng quan của tiêu chuẩn. Mật mã nhẹ là mật mã được dùng cho mục đích bảo mật, xác thực, nhận dạng và trao đổi khóa; phù hợp cài đặt cho những môi trường tài nguyên hạn chế. Trong ISO / IEC 29192, tính chất nhẹ được mô tả dựa trên nền tảng cài đặt. Trong triển khai phần cứng, diện tích chip và năng lượng tiêu thụ là những biện pháp quan trọng để đánh giá tính nhẹ của hệ mật. Trong triển khai phần mềm thì kích thước mã nguồn, kích thước RAM lại là tiêu chí cho một hệ mật được coi là nhẹ.

1.1.2. Quá trình hình thành và phát triển của mật mã nhẹ

Mật mã nhẹ đã được nhiều nhà nghiên cứu tìm hiểu từ rất lâu, nhưng mãi đến cách đây 13 năm mới có sự ra đời và áp dụng chính thức của những giải thuật mật mã nhẹ đầu tiên: Grain và Trivium (2005), Present, DESL, DESXL (2007), KATAN (2009) và Sprout (2015) ... Ngày càng nhiều thuật toán mã hóa nhẹ được ra đời với nhiều ứng dụng hữu ích [30].

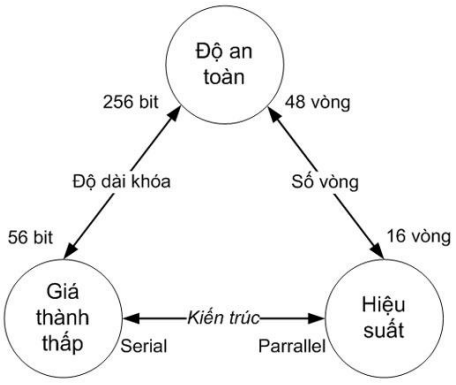


Hình 1-4: Số lượng mật mã nhẹ được phát triển bởi các nhà khoa học

Bên cạnh đó mạng lưới vạn vật kết nối cũng chứa đựng nhiều yếu điểm – cơ hội cho những kẻ tấn công thực hiện những hành động xấu. Nhất là trong những ứng dụng yêu cầu độ an toàn cao như các ứng dụng quân sự, ngân hàng hay tự động hóa. Ngoài những tấn công vào đường truyền vật lý, tính toán khắp nơi còn bị đe dọa bởi những cuộc tấn công chiếm quyền kiểm soát, tấn công lấy dữ liệu trên đường truyền, … Chính vì thế trong hệ thống tính toán khắp nơi, độ an toàn của hệ mật cần được quan tâm xem xét.

1.1.3. Nguyên lý thiết kế thuật toán mật mã nhẹ

Nguyên lý thiết kế các thuật toán mật mã nhẹ là một bài toán chưa có lời giải chính xác cho các thiết bị có tài nguyên hạn chế. Mật mã nhẹ cần đáp ứng được yêu cầu "nhẹ" trong cài đặt nhưng mặt khác nó vẫn phải đảm bảo mức độ an toàn cần thiết cho ứng dụng/phần cứng. Người thiết kế mật mã nhẹ phải thỏa hiệp, cân đối giữa ba tiêu chí: độ an toàn, hiệu suất và chi phí cài đặt (Hình 1-5).



Hình 1-5: Ba nguyên lý thiết kế thuật toán mật mã nhẹ

Độ an toàn: Khi thiết kế bất kỳ một hệ mật nào, điều đầu tiên người thiết kế cần quan tâm là độ an toàn của hệ mật. Độ an toàn có thể coi là một yếu tố sống còn của một hệ mật. Với mật mã nhẹ, người thiết kế cần thiết kế một hệ mật "đủ an toàn" trong điều kiện cho phép về chi phí và hiệu quả cài đặt. Dĩ nhiên độ an toàn của mật mã nhẹ chỉ đạt đến một ngưỡng an toàn chấp nhận được nào đó trong một điều kiện cụ thể.

Hiệu quả cài đặt, thường được đánh giá qua các độ đo tài nguyên được sử dụng bởi thuật toán như: diện tích bề mặt, số chu kỳ xung nhịp, thời gian thực thi, thông lượng, nguồn cung cấp, năng lượng, điện tích… Yêu cầu này liên quan mật thiết đến chi phí cài đặt, hiệu suất và khả năng tính toán trên đường truyền. Độ đo cho tính hiệu quả của phần cứng chính bằng tỷ lệ thông lượng và điện tích sử dụng của hệ mật mã đó (Bảng 1-1)

Bảng 1-1: Hiệu quả phần cứng của một số giải thuật mật mã nhẹ

Mã pháp	Số bits khóa	Số bits khối	Chu kỳ xung nhịp trên một khối	Thông lượng ở 100MHz (Kbps)	Xử lý logic (μm)	Điện tích (GEs)
Mã khối						
Present	80	64	32	200	0.18	1570
Hight	128	64	34	188	0.18	3048
mCrypton	96	64	13	492	0.13	2681
Mã dòng						
Trivium	80	1	1	100	0.13	2599
Grain	80	1	1	100	0.13	1294

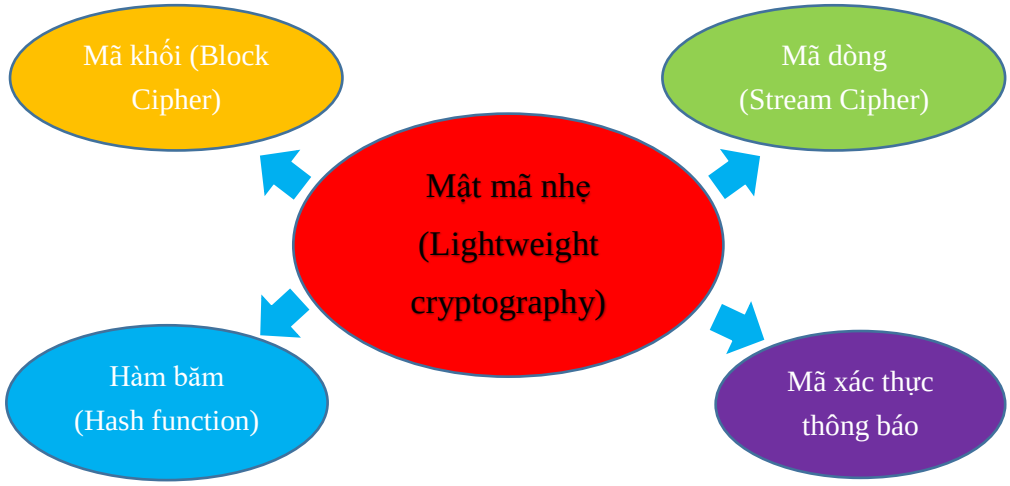
Giá thành của thuật toán: Thông thường các hệ mật mã nhẹ thường được áp dụng trên một số lượng lớn các thiết bị như hệ thống IoT. Chính vì vậy giá thành của thuật toán cũng đóng ý nghĩa quan trọng trong việc triển khai.

Một hệ mật tốt cần phải cân bằng giữa giá thành, hiệu suất và độ an toàn. Tuy nhiên việc cân bằng cả 3 yếu tố này là một bài toán khó. Tùy từng điều kiện, yêu cầu cụ thể, người thiết kế có thể cân đối nên ưu tiên khía cạnh nào hơn. Ví dụ như khi thực hiện cài đặt bằng phần cứng có hiệu suất cao thì thường dẫn tới các yêu cầu cao về điện tích, giá thành cao. Mặt khác, khi thiết kế các hệ mã ưu tiên độ an toàn trên một thiết bị có phần cứng thấp thì hiệu suất có thể sẽ rất thấp.

1.1.4. Các mật mã nhẹ nguyên thủy

Theo nghiên cứu của ECRYPT[1], mật mã nhẹ cũng có 4 loại mật mã nguyên thủy tương tự với 4 loại của mật mã truyền thống. Đó là mã khối, mã dòng, mã xác thực thông báo và hàm băm. Qua các hội nghị ECRYPT đã đề cập đến nhiều hệ mật như:

- Mã khối: HIGHT, KATAN/KTANTAN, DESL/DESX/DESXL, PRESENT, PRINTCIHER, SEA, XTEA, LBlock, …
- Mã dòng: Grain, MICKER, TRIVIUM, F-FCSR-H, WG-7
- Mã xác thực thông báo: SQUASH
- Hàm băm: MAME, H-PRESENT / DM-PRESENT, Keccak, PHOTON, QUARK hay Spongent …



Hình 1-6: Các nguyên thủy mật mã nhẹ

Bảng dưới mô tả một số thông số/ đặc điểm cũng như ưu, nhược điểm và ứng dụng của một số hệ mật mã nhẹ nguyên thủy đã được ECRYPT đề cập.

[1] ERCYPT là một mạng lưới nghiên cứu về mật mã nổi tiếng ở Châu Âu và được IST (Information Societies Technology) tài trợ từ năm 2004. Mục tiêu của nó là tăng cường sự hợp tác của các nhà nghiên cứu Châu Âu về an ninh thông tin, đặc biệt là trong kỹ thuật mật mã và kỹ thuật số.

Loại mật mã nhẹ	Hệ mật mã	Người thiết kế	Key (bits)	Block / IV (bits)	Đặc điểm	Ứng dụng
Mã khối	HIGHT		128	64	- 32 vòng lặp - Sử dụng phép toán đơn giản như XOR, mod 2^8 và dịch bits - Có thể thực hiện với 3048 cổng, công nghệ 0.25 μm	Triển khai trên các thiết bị hạn chế như RFID hay các thiết bị phổ biến khắp nơi
	KATAN / KTANTAN	Chrstophe de Canniere, Orr Dunkelman và Miroslav Knezevic	80	32 / 48 / 64	- Kiến trúc của KATAN / KTANTAN rất đơn giản. Bản rõ được lưu bởi 2 thanh ghi. Trong mỗi vòng, một số bit được lấy ra và đưa vào hàm phi tuyến Boolean, và LFRS 8 bits để mã hóa.	
	DES, DESL, DESX and DESXL		56 / 184	64	- 16 vòng lặp - DES sử dụng lặp lại một S-box (6*4 bits) 8 lần	
	PRESENT		80 / 128	64	- Cấu trúc SPN với 31 vòng - Mỗi vòng thực hiện phép cộng XOR để đưa vào khóa vòng - Tầng phi tuyến sử dụng một S-box 4 bits duy nhất được áp dụng 16 lần song song trong mỗi vòng	

	PRINTCIPHER		48 / 96		- PRINTCIPHER 48 sử dụng 48 bits khóa bí mật và cộng thêm 32 bits được sinh ra từ thuật toán mã hóa sử dụng 16 S-box 3 bits	Sử dụng trong mạch tích hợp in ấn (Integrated circuit – IC-printing)
	SEA – Scalable Encryption Algorithm	F.-X. Standaert, G. Piret, N. Gershenfeld, J.-J. Quisquater	8	48 / 96 / 144	- Thiết kế của SEA dựa trên một số phép toán cơ bản: XOR, thay thế, dịch trái, đảo bit, cộng mod 2^b	phần mềm trong bộ điều khiển, thẻ thông minh hoặc bộ vi xử lý
	XTEA	David Wheeler và Roger Needham	128	64	- Sử dụng 64 vòng lặp	
	LBlock	Wenling Wu and Lei Zhang	80	64	- Nó sử dụng một cấu trúc Feistel biến thể với 32 vòng lặp sử dụng 8 S-box 4 bits	Áp dụng trong các nền tảng phần mềm như vi điều khiển 8 bits
Mã dòng	Grain	Martin Hell, Thomas Johansson và Willi Meier năm 2004	64 / 80 / 128	64 / 96	- Mã dòng đồng bộ - Dựa trên LFSR và NFSR - Có thể triển khai song song - Ưu việt cho phần cứng nhẹ	Ứng dụng sử dụng WLAN, RFID/WSN
	MICKEY v2	Steve Babbage và Matthew Dodd năm 2005	80/ 128	0-80/ 0-128	- MICKEY 2.0 có kích thước mạch là 3,188 GE, hoạt động tối đa với tần số 454,5 MHz, và thông lượng 454,5 Mbps	Sử dụng cho nền tảng phần cứng với tài nguyên giới hạn
	Trivium	Christophe De Cannière	80	80	- Sử dụng 3 thanh ghi LFSR với	

		and Bart Preneel			thanh ghi đầu tiên sử dụng các "S-box" (1x1) để tạo ra các bit của keystream, sau đó ADD với hai LFSR còn lại	
	F-FCSR-H	Thierry Berger, François Arnault, and Cédric Lauradoux	80/ 128	80/ 128	- Chu kỳ của thuật toán là log(n) – n là tổng chiều dài thanh ghi	Mật mã dòng đầu tiên sử dụng các component FCSR
	WG-7	Y. Luo, Q. Chai, G. Gong và X. Lai năm 2010	80	81	- Thuật toán mã hóa dòng dựa trên WG Stream Cipher	Ứng dụng trong thẻ RFID và điện thoại di động
Hàm băm	MAME	Hirotaka Yoshida, Dai Watanabe, Katsuyuki Okeya, Jun Kitahara, Hongjun Wu, Ozgul Kucuk, Bart Preneel năm 2007	96	256	- Các thao tác lôgic đơn giản và S-box đã đem lại hiệu quả phần cứng cho MAME: chỉ cần 8,1 Kgates cho công nghệ 0,18 μm	Ứng dụng yêu cầu phần cứng hạn chế
	H-PRESENT / DM-PRESENT	Poschmann, Alex		80 / 128	- Hàm nén sử dụng mã khối PRESENT	Sử dụng trong những ứng dụng yêu cầu hàm một chiều và 64 bits bảo mật
	Keccak	Guido Bertoni, Joan Daemen, Michaël Peeters and Gilles Van Assche		256	- Với hiệu suất cao và sức đề kháng tốt, Keccak đã được Viện Tiêu chuẩn và Công nghệ (NIST) chọn	

					như một tiêu chuẩn mới của SHA-3 vào tháng 10/2012	
	PHOTON	Jian Guo, Thomas Peyrin, and Axel Poschmann			- Một cách hoán vị ngẫu nhiên dựa trên AES, sử dụng 12 lần lặp cho mỗi chuỗi sự biến đổi thực hiện trên một hình vuông Nibbles (4 bits)	
	QUARK	Jean-Philippe Aumasson, Luca Henzen, Willi Meier, Maria Naya-Plasencia		136 / 176 / 256	- Một hoán vị phần cứng P-Sponge sử dụng mã hóa KTANTAN và KATAN cùng với phần cứng theo định hướng của mã dòng Grain	Mã xác thực thông báo (MAC), sinh số giả ngẫu nhiên, mã hóa dòng...
	Spongent	Bogdanov, A., Knežević, M., Leander, G., Toz, D., Varıcı, K., & Verbauwhede		88 / 128 / 160 / 224 / 256	- Họ các hàm băm nhẹ - Linh hoạt về mức độ tuần tự và tốc độ, là một hàm băm với footprint nhỏ nhất trong phần cứng được công bố từ trước đến nay	
Mã xác thực thông báo	SQUASH	Adi Shamir (RFID Security Workshop 2007)			- Sử dụng hàm tuyến tính dựa trên thuật toán mã hóa khóa công khai Rabin	

Những báo cáo liên quan tới thuật toán mật mã đối xứng nhẹ cũng đã được trình bày trong hội nghị ECRYPT 2011 như: Mã khối (PUFFIN, PUFFIN2, LBlock, Piccolo, TWINE) và Hàm băm (SPONGENT).

1.1.5. Ứng dụng mật mã nhẹ trong IoT

Các thuật toán mật mã nhẹ ra đời và phát triển nhằm đáp ứng một nhu cầu cụ thể của một thiết bị hay một lớp bài toán nào đó. Chúng phù hợp với những ứng dụng, thiết bị có cấu hình nhỏ gọn, tốc độ xử lý nhanh và nhiều phép tính trong một khoảng thời gian cố định, yêu cầu bảo mật không quá cao. Bảng dưới mô tả một số thông số của một số hệ mật mã nhẹ trong quá trình triển khai.

Bảng 1-2: Một số thông số của một số hệ mật mã nhẹ trong triển khai

Hệ mật	GE	Throughput 100KHz (Kbit/s)	Logic (μm)
AES	2400	56.6	0.13
DES	2309	44.4	0.18
DESL	1848	44.4	0.18
DESXL	2168	44.4	0.18
PRESENT	1570	200	0.18
KATAN64	1054	25.1	0.13
Grain	1294	100	0.13
Trivium	2599	100	0.13
Sprout	813	100	0.18

Ta có thể thấy, hệ mật mã dòng trong mật mã nhẹ có thể thiết kế nhỏ gọn, yêu cầu phần cứng ít hơn và tốc độ nhanh hơn hệ mật mã khối.

Với những thiết kế riêng của mình, mật mã nhẹ đem lại nhiều lợi ích đặc trưng như:

- Yêu cầu nguồn tài nguyên thấp, năng lượng tiêu thụ nhỏ, phù hợp với những trang thiết bị cấu hình nhỏ. Vì các giải pháp trong mật mã nhẹ đều hướng đến việc cài đặt gọn nhẹ trên những thiết bị có năng lực tính toán thấp hay tài nguyên hạn chế.
- Giá thành rẻ. Mật mã nhẹ thường được ứng dụng trong những thiết bị có tính thâm nhập khắp nơi, cần phải triển khai hàng loạt trên hàng trăm, hàng nghìn, thậm chí hàng tỉ thiết bị. Chính vì thế việc giảm giá thành của công nghệ sử dụng được quan tâm hàng đầu.
- Hoạt động rất nhanh, thực hiện đầy đủ và hiệu quả chức năng của nó.

Bên cạnh những lợi ích rất thực tế, mật mã nhẹ còn chứa đựng nhiều hiểm họa và khó khăn trong quá trình áp dụng.

- Độ an toàn của mật mã nhẹ không cao như những mật mã thông thường khác, vì nó phải cân đối giữa hai yếu tố là tính an toàn và tính gọn nhẹ. Về cơ bản mật mã nhẹ không phù hợp với những ứng dụng yêu cầu độ an toàn cao như các ứng dụng quân sự, tài chính, ...

- Các mật mã nhẹ chỉ xử lý được một lượng thông tin nhỏ, không có băng thông cao. Bản thân mật mã nhẹ được thiết kế chủ yếu không phải cho phần mềm mà để áp dụng, nhúng vào các phần cứng. Vì lượng thông tin cần truyền trên đây không nhiều.
- Khó khăn trong việc tối ưu hóa các thuật toán mật mã nhẹ hiện có. Bản thân các hệ mật mã nhẹ đã mất rất nhiều sức đề kháng trong khi tài nguyên được sử dụng là hạn chế. Điều này làm chậm đáng kể việc phát triển một thuật toán trong mật mã nhẹ.

Tất cả những điều này làm cho việc sử dụng mật mã nhẹ trong thực hành có chuyên môn cao là khá khó khăn. Tùy từng nhu cầu của ứng dụng cũng như phần cứng mà quyết định có nên lựa chọn sử dụng mật mã nhẹ cho vấn đề đảm bảo an toàn của hệ thống hay không.

1.2. Mật mã dòng trong mật mã nhẹ

1.2.1. Khái niệm

Nếu xét theo số bit xử lý, mật mã khóa đối xứng có hai loại là mã dòng (stream cipher) và mã khối (block cipher). Mã khối làm việc bằng cách chia khối dữ liệu cần mã hóa thành những khối có độ dài nhất định và xử lý (mã hóa/giải mã) trên các khối dữ liệu này. Yêu cầu của mã khối là phải biết trước kích thước bản rõ. Tuy nhiên không phải dữ liệu cần mã hóa nào cũng rõ ràng, tường minh ngay từ đầu, mà thường không biết trước kích thước hoặc biến thiên theo thời gian. Mật mã dòng hoạt động với dữ liệu đầu vào được mã hóa từng bit một có thể đáp ứng được sự biến thiên theo thời gian trên những khối bản rõ (plaintext) riêng biệt.

Mật mã dòng thực sự phát triển từ những năm 1960 với rất nhiều tổ chức sử dụng như các quân đội, ngoại giao, các tổ chức gián điệp, các doanh nghiệp, viễn thông... Mã dòng ngày càng trở lên phổ biến nhất là khi những thiết bị mã hóa điện tử bán dẫn bắt đầu xuất hiện với dung lượng bộ nhớ thấp. Nhất là với tốc độ phát triển của IoT như ngày nay, theo dự đoán của SICCO thì đến năm 2020 có thể có đến 50 tỷ thiết bị tham gia vào Internet. Ngoài máy tính, các thiết bị có cấu hình cao thì còn có rất nhiều thiết bị chỉ có chip xử lý hạn chế như tủ lạnh, điều hòa, máy giặt...

So với các thuật toán mã hóa dòng khác, các thuật toán mã hóa dòng nhẹ dành được ưu thế về sự đơn giản trong triển khai. Theo thực nghiệm của Good, T., & Benaissa, M [17], RC4 cần 50000 GE và có thông lượng 10 Gbps; Snow 3G cần 11000 GE, 1.72 Gbps; A5/1 cần 700 GE; trong khi các thuật toán mã hóa nhẹ tiêu biểu như Grain 80 chỉ cần đến 1294 GE để đạt thông lượng 725 Mbps, hay Trivium x64 chỉ cần đến 4921 GE để có thông lượng 22300 Mbps.

Bảng 1-3: Kết quả triển khai các thuật toán mã hóa nhẹ trên phần cứng (1) [17]

Hệ mật	**Key**	**Interface**	**Load/ini cycles**	**Bits/Cycle**	**Max. Clock freq. MHz**	**Area NAND GE, gates**	**Leakage power, µW**	**Total Pow10 MHz, µW**
Grain80	80	1	312	1	724.6	1294	2.22	109.45
Grain80, x4	80	4	81	4	649.4	1678	3.24	126.59
Grain80, x8	80	8	41	8	632.9	2191	4.63	150.66
Grain80, x16	80	16	21	16	617.3	3239	7.40	200.53
Trivium	80	1	1333	1	358.4	2599	3.84	181.18
Trivium, x4	80	4	36	5	413.2	2660	4.04	184.83
Trivium, x8	80	8	170	8	359.7	2801	4.45	199.59
Trivium, x16	80	16	87	16	408.2	3185	5.84	231.23
Trivium, x32	80	32	45	32	350.9	3787	7.50	282.55
Trivium, x64	80	64	24	64	348.4	4921	10.68	374.19
F-FCSR-H	80	8	225	8	392.2	4760	7.97	269.27
Grain128	128	1	513	1	925.9	1857	2.70	167.73
Grain128, x4	128	4	129	4	584.8	2129	3.81	183.37
Grain128, x8	128	8	65	8	581.4	2489	4.90	205.12
Grain128, x16	128	16	33	16	540.5	3189	6.88	254.64
Grain128, x32	128	32	17	32	452.5	4617	11.44	344.74
Mickey128	128	1	417	1	413.2	5039	8.14	310.73
Phelix, ½ md	256	32	51	16	88.1	13159	23.90	928.85
Phelix, 1 md	256	32	34	32	63.0	15032	27.60	1432.37
Sosemaunk	256	32	255	32	188.3	18819	33.55	812.47
Salsa20, 1h	256	32	533	0.994	121.9	12126	19.36	708.46
Salsa20, 4h	256	32	100	5.28	155.0	12914	22.34	883.94
Salsa20, 16h	256	32	40	13.84	54.4	16394	31.43	2368.98
Salsa20, 32h	256	32	30	18.96	35.2	18626	35.06	3375.28
AES[7]*	128	32	50	2.37	131.2	5398	-	-
AES[17]*	128	8	1016	0.124	80.0	3400	-	-
Tốt			*Thấp*	*Thấp*	*Cao*	*Thấp*	*Thấp*	*Thấp*

Bảng 1-4: Kết quả triển khai các thuật toán mã hóa nhẹ trên phần cứng (2)

Hệ mật	**Thông lượng (Mbps)**	**Năng lượng tính toán (uW)**	**Năng lượng/ bit (pJ/bit)**	**Vùng thời gian sản phẩm (um^2– us)**	**Thông lượng / diện tích (kbps/um^2)**	**Năng lượng-Diện tích-Thời gian (nJ-um^2)**
Grain80	725	7,772	10.73	9.26	108.00	72.0
Grain80, x4	2,778	8,659	3.08	3.13	319.33	26.8
Grain80, x8	5,063	9,247	1.83	2.24	445.78	20.7
Grain80, x16	9,877	11,929	1.21	1.70	588.27	20.3
Trivium	355	6,360	17.74	37.58	26.61	239.1
Trivium, x4	1,653	7,475	4.52	8.34	119.88	62.3
Trivium, x8	2,878	7,024	2.44	5.05	198.16	35.4
Trivium, x16	6,531	9,205	1.41	2.53	395.57	23.3
Trivium, x32	11,228	9,658	0.86	1.75	571.88	16.9
Trivium, x64	23,300	12,677	0.57	1.14	874.14	14.5
F-FCSR-H	3,137	10,255	3.27	7.87	127.13	80.7
Grain128	926	15,283	16.51	10.39	96.20	158.9
Grain128, x4	2,339	10,505	4.49	4.72	211.98	49.6
Grain128, x8	4,651	11,646	2.50	2.77	360.52	32.3
Grain128, x16	8,648	13,399	1.55	1.91	523.10	25.6
Grain128, x32	14,480	15,093	1.04	1.65	604.92	24.9
Mickey128	413	12,512	30.28	63.21	15.82	790.9
Phelix, ½ md	1,410	7,997	5.67	48.39	20.66	387.0
Phelix, 1 md	2,016	8,879	4.40	38.65	25.88	343.2
Sosemaunk	6,026	14,702	2.44	16.19	61.77	238.0
Salsa20, 1h	121	8,423	69.47	518.50	1.93	4367.3
Salsa20, 4h	818	13,380	16.35	81.80	12.22	1094.6
Salsa20, 16h	753	12,756	16.93	112.82	8.86	1439.2
Salsa20, 32h	668	11,801	17.67	144.56	6.92	1705.9
AES[7]*	311	-	-	90.12	11.10	-
AES[17]*	10	-	-	1776.33	0.56	-
Tốt	*Cao*	*Thấp*	*Thấp*	*Thấp*	*Cao*	*Thấp*

1.2.2. Các thuật toán đặc trưng

1.2.2.1. Hệ mật ChaCha

ChaCha là một biến thể của hệ mật mã dòng Salsa20[2] – một hệ mật có tốc độ nhanh hơn cả AES. ChaCha được phát triển bởi Daniel J. Bernstein (University of Illinois at Chicago, USA) năm 2008 (SASC 2008). ChaCha20 với 20 vòng đã được chuẩn hóa trong IETF RFC 7539. ChaCha sử dụng 256 bits khóa, một bộ đếm khởi tạo 32 bits và vector khởi tạo 96 bits. Ngoài khóa bí mật và IV, nó còn nhận một hằng số 128 bits và bộ đếm khối 32 bits để xuất ra một số giả ngẫu nhiên 512 bits. Trong mỗi vòng lặp, ChaCha sử dụng 4 phép cộng, 4 phép XOR và 4 phép dịch bit để cập nhật trạng thái từ 32 bits khởi tạo.

$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d \lll= 16;$$
$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b \lll= 12;$$
$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d \lll= 8;$$
$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b \lll= 7;$$

Theo đánh giá tại eBASC[3] (tính đến tháng 6 năm 2016), thuật toán đạt được 1,2 chu kỳ / byte trên bộ vi xử lý Intel Core i5. Ngoài ra, theo đánh giá của Fair Evaluation of Lightweight Cryptographic Systems (FELICS)[4] (tính đến tháng 6 năm 2016), thuật toán yêu cầu 144 chu kỳ để khởi tạo bộ vi xử lý ARM Cortex-M3 và có thông lượng là 54,3 chu kỳ / byte. Tính đến năm 2016, chưa có kết quả thỏa hiệp an ninh nào của ChaCha20 được tìm ra. Aumasson và cộng sự đã chứng minh được rằng 7 trong số 20 vòng của ChaCha20 có thể bị tấn công bằng tấn công khác biệt hiệu quả hơn là tìm kiếm toàn diện. Các phân tích khác bởi Shi và Maitra cũng nhằm cải thiện hiệu quả cảu Aumasson nhưng số lượng các vòng có thể tấn công cũng chỉ giới hạn ở mức 7.

Mật mã này chủ yếu được sử dụng kết hợp với một mã xác thực tin nhắn, Poly-1305, để phục vụ như là một mã hóa xác thực. Mã hóa ChaCha được sử dụng để bảo vệ các kênh truyền thông (https) cho các dịch vụ được cung cấp bởi Google [3].

1.2.2.2. Hệ mật E0

E0 là một hệ mật mã dòng nhẹ được sử dụng rộng rãi, nhất là trong các giao thức Bluetooth. Nó tạo ra một chuỗi số giả ngẫu nhiên để XOR với dữ liệu tạo ra bản mã. Độ dài khóa của các phiên bản E0 có thể khác nhau nhưng luôn luôn là bội số của 2, thường là 128 bit. Tại mỗi lần lặp, E0 tạo ra một bit bằng cách sử dụng 4 thanh ghi dịch chuyển có độ dài khác nhau (25, 31, 33, 39 bits) và hai trạng thái khởi tạo độ dài

[2] Salsa20 là một họ các hệ mật mã dòng được mô tả chi tiết trong "*The Salsa20 family of stream ciphers" của* Bernstein, D.J (2005). eSTREAM submission

[3] ebacs: Ecrypt benchmarking of cryptographic systems. http://bench.cr.yp.to/ results-stream.html.

[4] Felics stream ciphers brief results. https://www.cryptolux.org/index.php/FELICS_ Stream_Ciphers_Brief_Results

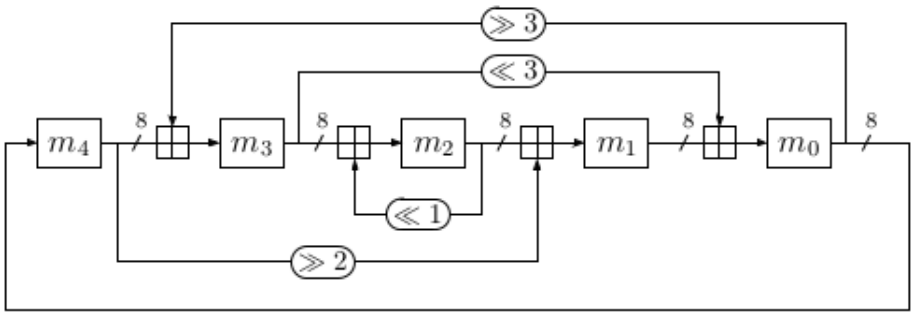
2 bits. Tại mỗi tick, các thanh ghi được dịch chuyển và hai trạng thái được cập nhật với trạng thái hiện tại, trạng thái trước làm các giá trị trong thanh ghi thay đổi. Bốn bits được tách ra từ các thanh ghi, cộng lại với nhau. Thuật toán XOR tổng hợp giá trị trong thanh ghi 2 bit. Bit đầu tiên của kết quả là đầu ra cho việc mã hóa.

Đã có những cuộc tấn công và E0 tuy nhiên hiệu quả không cao và vẫn chưa thực hiện được trong thực tế vì cần một số lượng lớn keystream.

1.2.2.3. Hệ mật FCSR-based Stream-Ciphers

Mật mã dòng đầu tiên sử dụng các component FCSR (Feedback with Carry Shift Register) được giới thiệu trong eSTREAM tháng 01/2009 [8]. Tuy nhiên sau sự phá mã của Hell và Johansson [13], các nhà thiết kế sau đó đã tái xuất bản thuật toán này với sự thay đổi đáng kể cấu trúc của FCSR. Hiện tại FCSR là một thanh ghi dịch chuyển phản hồi (Feedback Shift Register) có độ dài b bits. Trong mỗi một vòng, mỗi ô có thể thay đổi giá trị theo hai cách:

- Nếu nó không nhận được phản hồi (m_4 trên hình bên phải) thì nội dung của nó tại thời điểm t + 1 là nội dung của ô trước đó (m_0) trong thời điểm t.
- Nếu nó nhận được thông tin phản hồi, nó sẽ được "cộng" thêm b bits. Bit có trọng số nhỏ là giá trị mới của ô đó tại thời điểm t + 1, bit có trọng số cao được lưu trữ trong ô mang thông tin phản hồi.



Hình 1-7: Ví dụ về FCSR

Trong [13] các tác giả đã chứng minh chu kỳ của thuật toán này là log(n), trong đó n là tổng chiều dài của thanh ghi. Điều đó có nghĩa là FCSR giống như áp dụng một hoán vị trạng thái nội bộ của nó, do đó không có entropy bị mất. Tuy nhiên, có thể xảy ra va chạm đáng kể trong chu kỳ đầu của giải thuật. Đó cũng là tiền đề cho cuộc tấn công GLUON.

1.2.2.4. Hệ mật F-FCSR-H v3

F-FCSR-H v3 cũng là hệ mã hóa dòng giống FCSR-based Stream-Ciphers. Tuy nhiên các FCSR được sử dụng trong F-FCSR-H v3 có 1 bit trong mỗi ô, 82 phản hồi và 8 bits được lọc tại mỗi lần. Hệ mật này yêu cầu 80 bits khóa và 80 bits IV.

1.2.2.5. Hệ mật F-FCSR-16 v3

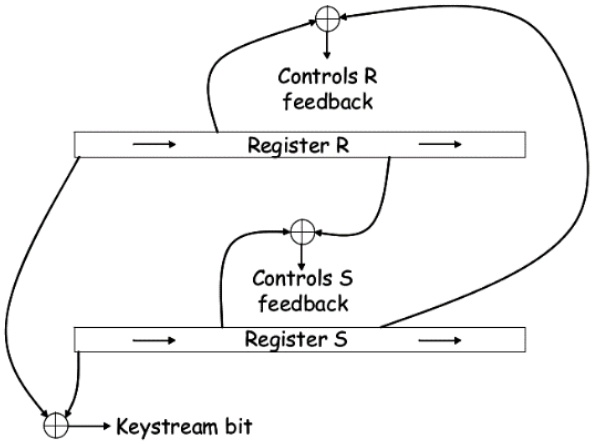
Tương tự như F-FCSR-H v3, các FCSR được sử dụng trong F-FCSR-16 v3 có 1 bits trong mỗi ô, 130 phản hồi và 16 bits được lọc tại mỗi lần. Hệ mật này yêu cầu 128 bits khóa và 128 bits IV.

1.2.2.6. Hệ mật Grain

Grain có mặt trong danh mục của eSTREAM, dựa trên hai FSR khác nhau có ảnh hưởng xung nhịp theo cách không tuyến tính và một hàm kết hợp phi tuyến để tạo ra keystream từ nội dung của FSR. Grain phù hợp cho việc triển khai với phần cứng nhẹ. Nó có thể đạt được mức độ song song nhất định, và triển khai cho các ứng dụng thực tế. Nội dung cụ thể của Grain sẽ được trình bày trong phần tiếp theo.

1.2.2.7. Hệ mật MICKEY v2

MICKEY (Mutual Irregular Clocking KEYstream generator) được xây dựng dựa trên 2 thanh ghi LFSR không đồng bộ bởi Steve Babbage và Matthew Dodd. MICKEY được công bố lần đầu tiên tại eSTREAM năm 2005. Thuật toán được thiết kế sử dụng cho các nền tảng phần cứng với tài nguyên giới hạn và là một trong ba hệ mật được chấp nhận trong eSTREAM.

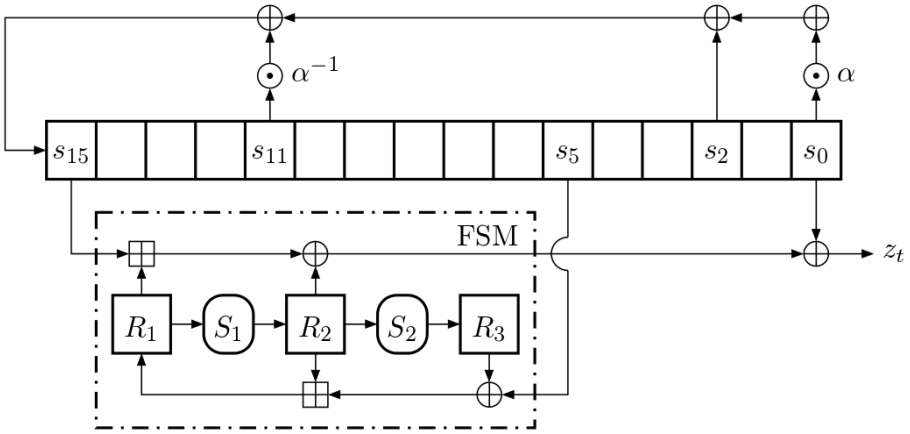


Hình 1-8: Kiến trúc của MICKEY

MICKEY là một giải thuật hoàn toàn miễn phí với người sử dụng. Nó cần một khóa 80 bits, một vector khởi tạo độ dài từ 0 đến 80 bits và một keystream có chiều dài tối đa là 2^{40} bit. MICKEY 2.0 có kích thước mạch là 3,188 GE, hoạt động tối đa với tần số 454,5 MHz, và thông lượng 454,5 Mbps.

1.2.2.8. Hệ mật SNOW 3G

SNOW 3G là một hệ mật mã dòng nhẹ sử dụng 128 bits khóa, 128 bits IV, là một bản cập nhật của SNOW và SNOW 2.0. SNOW 3G được dựa trên thanh ghi LFSR với 16 ô, mỗi ô có chiều dài 8 bits và một máy trạng thái hữu hạn (Finite State Machine - FSM). Các LFSR được cập nhật bằng cách sử dụng số học hữu hạn để tính toán các ô phản hồi.

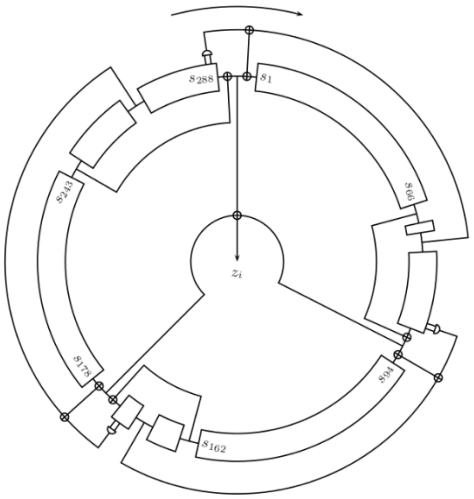


Hình 1-9: Kiến trúc của SNOW 3G

SNOW 3G là một thuật toán nền tảng trong 3GPP, ứng dụng rộng rãi cho việc bảo mật dữ liệu trên điện thoại di động. Đặc biệt SNOW 3G còn góp mặt trong việc đảm bảo an toàn dữ liệu của công nghệ 4G/LTE hiện nay.

1.2.2.9. Hệ mật Trivium

Để thu hẹp khoảng cách về sự hiểu biết lý thuyết giữa mật mã khối và mật mã dòng, các tác giả (Christophe De Canni`ere, Bart Preneel (Katholieke Universiteit Leuven)) đã xây dựng một hệ mật có sự kết hợp giữa mật mã khối và mật mã dòng, đó là Trivium. Trivium sử dụng 3 thanh ghi LFSR với thanh ghi đầu tiên sử dụng các "S-box" (1x1) để tạo ra các bit của keystream, sau đó ADD với hai LFSR còn lại. Thuật toán này có kích thước khóa và kích thước IV là 80 bits, kích thước trạng thái khởi tạo là 288 bits. Nhưng các tác giả cũng cung cấp một phiên bản yếu hơn chỉ sử dụng 2 thanh ghi LFSR là Bivium.



Hình 1-10: Thiết kế của Trivium

Trivium có một thiết kế độc đáo với ba thanh ghi dịch hồi phi tuyến có chiều dài khác nhau. Dựa trên các hoạt động bitwise, có mức độ song song cao, hệ mật mã dòng này có tính năng thực thi trên phần cứng nhẹ và tốc độ cao trong việc triển khai phần mềm. Tuy nhiên, nó không thích hợp cho xử lý dữ liệu ngắn vì thời gian khởi tạo dài.

1.2.3. Triển vọng của mật mã dòng trong mật mã nhẹ trong IoT

Ngày nay, mật mã dòng trong mật mã nhẹ đã có mặt trên các thiết bị điện thoại di động, các thiết bị nhúng hay trên các ứng dụng yêu cầu phần cứng nhỏ gọn, năng lượng ít. Ví dụ như trong các giao thức Bluetooth, truyền thông 3GPP, 4G/LTE hay các kênh truyền thông https cho các dịch vụ được cung cấp bởi Google.

Các kỹ thuật mã hóa dòng nhẹ nên được sử dụng trong IoT vì 2 lý do sau đây [27]:

- Hiệu quả của việc trao đổi thông tin. Để có thể sử dụng mã hóa đầu cuối trong IoT, mỗi node đều cần phải thực hiện một thuật toán mã hóa khóa đối xứng. Đối với các thiết bị nhỏ, việc hạn chế tiêu thụ năng lượng cho hoạt động mã hóa cần được đảm bảo, việc ứng dụng mã hóa dòng trong mật mã nhẹ cho phép tiêu thụ năng lượng thấp ở các thiết bị đầu cuối.
- Khả năng ứng dụng cho các thiết bị công suất thấp. Việc thực thi các hệ mật mã dòng nhẹ là đơn giản so với các mật mã thông thường và nó mở ra khả năng có thể kết nối với các thiết bị mạng công suất nhỏ.

Những thuật toán khóa công khai đã và đang được ứng dụng rất nhiều trong thực tế nhưng để áp dụng nó trong những thiết bị nhúng, thiết bị hoạt động với công suất nhỏ... thì hoàn toàn không phù hợp. Những trường hợp này cần áp dụng mật mã nhẹ, nhất là mật mã dòng để đem lại hiệu quả cao về năng lượng tiêu thụ, công suất và độ an toàn.

Chương 2. HỌ GRAIN

2.1. Lịch sử

Grain là hệ mật mã dòng được đăng trên eSTREAM[5] bởi Martin Hell, Thomas Johansson và Willi Meier năm 2004 với phiên bản đầu tiên Grain v0 [21]. Sau đó hệ mật này tiếp tục được phát triển thành Grain v1 [12] – là một trong bảy dự án được eSTREAM đưa vào các danh mục đầu tư từ 09/09/2008. Cùng với Grain v1 là một phiên bản mật mã với khóa bí mật 128 bits – Grain-128 [12] cũng được áp dụng rộng rãi hiện nay. Bài nghiên cứu này chủ yếu tập trung phân tích hệ mã nguyên thủy Grain v0, Grain v1, Grain-128 và một phiên bản xác thực Grain-128a cùng những cuộc tấn công đã được thực hiện trên các hệ mã này.

2.2. Mô tả Grain

2.2.1. Grain V0

Thuật toán nguyên thủy Grain đề xuất một định hướng mật mã dòng bit đồng bộ. Trong một hệ mã hóa dòng đồng bộ, các khóa dòng sẽ được tạo một cách độc lập từ bản rõ. Thiết kế của thuật toán này được dựa trên hai thanh ghi dịch chuyển, một thanh ghi dịch hồi tuyến tính (LFSR - linear feedback shift register) và một thanh ghi phản hồi phi tuyến (NFSR - nonlinear feedback shift register). Hai thanh ghi này cùng với một hàm đầu ra tạo ra ba khối chính cho thuật toán mã hóa Grain. LFSR đảm bảo chu kỳ tối thiểu cho keystream và cũng tạo ra sự cân bằng cho đầu ra. NFSR cùng với chức năng phi tuyến cũng đem lại sự phi tuyến cho mật mã. Đầu vào cho NFSR được kết hợp cùng đầu ra của LFSR để cân bằng trạng thái của NFSR. Vì vậy chúng ta sử dụng ký hiệu NFSR mặc dù bản chất của điều này thực ra là một bộ lọc. Cả hai thanh ghi dịch chuyển đều có kích thước 80 bit. Kích thước khóa là 80 bits và kích thước IV được xác định là 64 bits. Mật mã được thiết kế sao cho không có cuộc tấn công nào nhanh hơn việc tìm kiếm vét cạn, vì vậy cuộc tấn công tốt nhất cần phải có độ phức tạp tính toán không thấp hơn đáng kể so với 2^{80}.

Nội dung của LFSR được biểu diễn bằng $s_i, s_{i+1}, \dots, s_{i+79}$ và nội dung của NFSR được mô tả bằng $b_i, b_{i+1}, \dots, b_{i+79}$.

Đa thức nguyên thủy bậc 80 của bộ ghi dịch hồi tuyến tính LFSR, f(x) được định nghĩa là:

$$f_0(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}$$

Loại bỏ những phần có thể mơ hồ, chúng ta sử dụng một phiên bản cập nhật của LFSR như sau:

$$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i$$

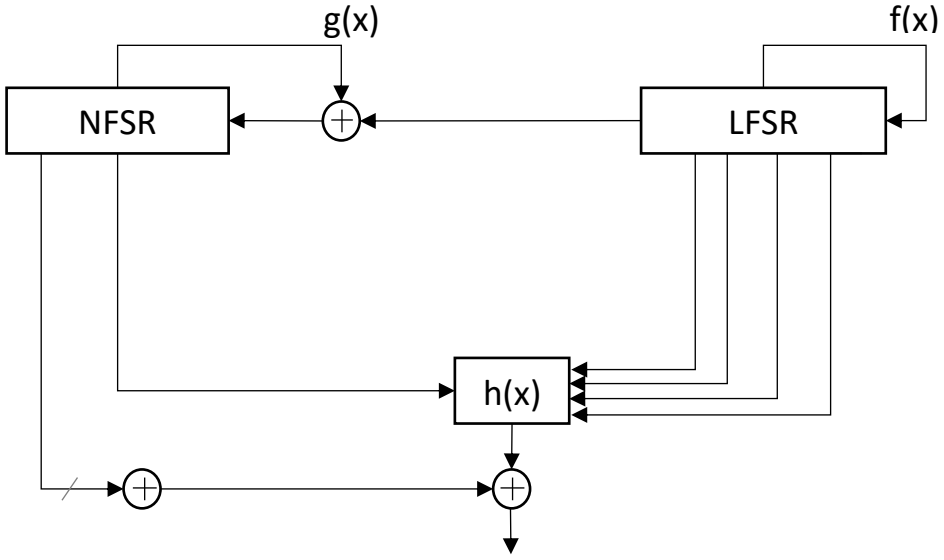
[5] eSTREAM là một dự án để "xác định thuật toán mã hóa dòng mới phù hợp để áp dụng rộng rãi", được tổ chức bởi EU ECRYPT (European Network of Excellence in Cryptology).

Hàm của bộ ghi dịch hồi phi tuyến (NFSR) được định nghĩa như sau:

$$\begin{aligned} g_0(x) = 1 &+ x^{18} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{66} + x^{71} + x^{80} + x^{17}x^{20} \\ &+ x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} \\ &+ x^{20}x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} \\ &+ x^{28}x^{35}x^{43}x^{47}x^{52}x^{59} \end{aligned}$$

Một lần nữa, chúng ta tiếp tục loại bỏ những mơ hồ để được hàm cập nhật như sau:

$$\begin{aligned} b_{i+80} = s_i &+ b_{i+62} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + b_{i+14} + b_{i+9} \\ &+ b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + b_{i+15}b_{i+9} + b_{i+60}b_{i+52}b_{i+45} \\ &+ b_{i+33}b_{i+28}b_{i+21} + b_{i+63}b_{i+45}b_{i+28}b_{i+9} + b_{i+60}b_{i+52}b_{i+37}b_{i+33} \\ &+ b_{i+63}b_{i+60}b_{i+21}b_{i+15} + b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} \\ &+ b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21} \end{aligned}$$



Hình 2-1: Kiến trúc của Grain

Nội dung của hai thanh ghi được thay đổi trạng thái của mã hóa. Từ 5 biến đầu vào, qua hàm logic h(x) được cân bằng với một đầu ra của hàm phi tuyến NFSR.

$$h_0(x) = x_1 + x_4 + x_0x_3 + x_2x_3 + x_3x_4 + x_0x_1x_2 + x_0x_2x_3 + x_0x_2x_4 + x_1x_2x_4 + x_2x_3x_4$$

Trong đó x_0, x_1, x_2, x_3, x_4 tương ứng với các vị trí $s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63}$. Đầu ra của hàm này sẽ là

$$z^0{}_i = b_i + h_0(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63})$$

Tấn công Grain V0

Một cuộc tấn công khác biệt với Grain V0 [22] đã được Khazaei, Hasanzadeh và Kiaei sử dụng các khái niệm xấp xỉ mạch tuần tự tuyến tính – một phương phán được đưa ra bởi Golic. Cuộc tấn công này đòi hỏi một giai đoạn tiền xử lý để tính các bội số của một đa thức nguyên thủy bậc 80 và yêu cầu thời gian và độ phức tạp tính toán là $O(2^{40})$. Cuộc tấn công phân biệt này có thể phân biệt chuỗi đầu ra của Grain từ một

dạng ngẫu nhiên hoàn toàn với độ phức tạp $O(2^{61.4})$. Cuộc tấn công thứ hai [20] được trình bày bởi Barbein, Gilbert và Maximov, đó là cuộc tấn công phục hồi chống lại Grain V0. Trong cuộc tấn công này, trước tiên phương pháp xấp xỉ tuyến tính được sử dụng để lấy được các bit LFSR và sử dụng các bit này để khôi phục lại trạng thái ban đầu của NFSR và hiểu biết về khóa. Cuộc tấn công này đòi hỏi 238 khóa dòng và độ phức tạp tính toán là $O(2^{43})$ để phục hồi chìa khóa. Để ngăn chặn các cuộc tấn công và tăng cường an ninh, các nhà thiết kế của Grain đã đề xuất một thiết kế mới Grain V1 và gửi nó cho eSTREAM.

2.2.2. Grain V1

Tương tự như Grain v0, Grain v1 cũng sử dụng $k = 80$ và số bits của đầu ra là $l = 64$. Thiết kế của Grain V1 cũng tương tự như Grain V0 với hai thanh ghi LFSR và NFSR 80 bits và trạng thái khởi tạo 160 bits. Thông tin phản hồi tuyến tính của LFSR được giữ lại giống như Grain V0 nhưng thông tin phản hồi phi tuyến và cập nhật của NFSR có sửa đổi một chút để khắc phục điểm yếu của Grain V0.

$$\begin{aligned}\mathrm{g}_1(\mathrm{x}) = 1 &+ x^{18} + x^{20} + x^{28} + x^{35} + x^{43} + x^{47} + x^{52} + x^{59} + x^{65} + x^{71} + x^{80} + x^{17}x^{20} \\ &+ x^{43}x^{47} + x^{65}x^{71} + x^{20}x^{28}x^{35} + x^{47}x^{52}x^{59} + x^{17}x^{35}x^{52}x^{71} \\ &+ x^{20}\,x^{28}x^{43}x^{47} + x^{17}x^{20}x^{59}x^{65} + x^{17}x^{20}x^{28}x^{35}x^{43} + x^{47}x^{52}x^{59}x^{65}x^{71} \\ &+ x^{28}x^{35}x^{43}x^{52}x^{59}\end{aligned}$$

Hàm cập nhật của NFSR được định nghĩa bởi

$$\begin{aligned}b_{i+80} = s_i &+ b_i + b_{i+9} + b_{i+14} + b_{i+21} + b_{i+28} + b_{i+33} + b_{i+37} + b_{i+45} + b_{i+52} \\ &+ b_{i+60} + b_{i+62} + b_{i+9}b_{i+15} + b_{i+33}b_{i+37} + b_{i+60}b_{i+63} \\ &+ b_{i+21}b_{i+28}b_{i+33} + b_{i+45}b_{i+52}b_{i+60} + b_{i+15}b_{i+21}b_{i+60}b_{i+63} \\ &+ b_{i+33}b_{i+37}b_{i+52}b_{i+60} + b_{i+9}b_{i+28}b_{i+45}b_{i+63} \\ &+ b_{i+9}b_{i+15}b_{i+21}b_{i+28}b_{i+33} + b_{i+37}b_{i+45}b_{i+52}b_{i+60}b_{i+63} \\ &+ b_{i+21}b_{i+28}b_{i+33}b_{i+37}b_{i+45}b_{i+52}\end{aligned}$$

Các bit đầu ra của Grain v1 được định nghĩa khác với Grain v0:

$$z^1{}_i = \sum_{i \in A_1} b_{k+i} + \mathrm{h}_1(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63})$$

Trong đó $A_1 = \{1, 2, 4, 10, 31, 43, 56\}$.

Tấn công Grain V1

Canniere, Kucuk và Preneel [14] đã tấn công thành công vào Grain V1 bằng cách sử dụng điểm yếu của thuật toán khởi tạo. Cuộc tấn công này khai thác tính trượt của Grain V1 đem lại sự giống nhau trong khởi tạo và quy trình tạo ra khóa dòng. Những kẻ tấn công đã tuyên bố có thể giảm độ phức tạp tấn công bằng một nửa cuộc tấn công tìm kiếm vét cạn. Lee và cộng sự [15] đã mở rộng và đề xuất một cuộc tấn công tinh vi bằng cách khai thác cùng một điểm yếu liên quan đến Grain V1. Cuộc tấn công này là

cuộc tấn công khôi phục khóa, có thể khôi phục khóa với 222,59 IV được lựa chọn, 226,29 bits khóa và 222,90 phép tính.

Grain V1 vẫn còn nhiều lỗ hổng như cuộc tấn công Dynamic Cube có thể khôi phục 80 bits khóa nếu số vòng khởi tạo giảm xuống 100 với độ phức tạp tính toán là 248. Chính vì vậy các nhà thiết kế Grain đã tiếp tục nghiên cứu và đưa ra một phiên bản mạnh mẽ, an toàn hơn – Grain-128.

2.2.3. Grain 128

Thuật toán Grain-128 có đầu vào $k = 128$ và đầu ra $l = 96$. Hai thanh ghi đều có độ dài 128 bits. Hàm của LFSR được định nghĩa như sau:

$$\mathrm{f}_{128}(\mathrm{x}) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}$$

Đầu ra của hàm LFSR sẽ là:

$$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}$$

Hàm của NFSR được định nghĩa như sau:

$$\mathrm{g}_{128}(\mathrm{x}) = 1 + x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} + x^{69}x^{101} + x^{88}x^{80} + x^{110}x^{111} + x^{115}x^{117}$$

Đầu ra của hàm NFSR sẽ là

$$b_{i+128} = s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84}$$

Bộ lọc: $\mathrm{h}_{128}(\mathrm{x}) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$

Với các giá trị $x_0, \dots, x_8$ tương ứng với các vị trí $b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}, s_{i+95}$.

Đầu ra của Grain-128 sẽ là**:**

$$z^{128}{}_i = \sum_{i \in A_{128}} b_{k+i} + s_{93+i} + \mathrm{h}_{128}(b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}, s_{i+95})$$

Trong đó $A_{128} = \{2, 15, 36, 45, 64, 73, 89\}$.

Tấn công Grain-128

Với thiết kế khá tương tự Grain V1, Grain-128 cũng có thể bị tấn công bởi những phương pháp tương tự Grain V1. Phương pháp Dynamic Cube attack có thể khôi phục đầy đủ khóa trong thời gian thực khi số vòng khởi tạo giảm xuống còn 207. Nếu số vòng được giảm đến 250, thì độ phức tạp tính toán của cuộc tấn công này sẽ giảm còn 228 so với tấn công tìm kiếm vét cạn.

2.2.4. Grain-128a

Trong nghiên cứu mới đây [29], Martin Agren, Martin Hell, Thomas Johansson, và Willi Meier đã đề xuất một phiên bản mới của Grain-128. Đó là Grain-128a một phiên bản được bổ sung thêm chức năng xác thực và sự cải thiện về hiệu năng một cách rõ rệt. Đồng thời Grain-128a có sử dụng các hàm phi tuyến khác nhau để tăng cường sự chống lại các cuộc tấn công đã biết trước đây lên Grain-128.

Grain-128a bao gồm cơ chế tạo ra luồng đầu ra trước và một cơ chế phụ, tùy chọn để xác thực. Về cơ bản Grain-128a khá giống với Grain-128, sử dụng chung hàm LFSR, khác nhau ở hàm NFSR.

$$\begin{aligned} \mathrm{g}_{128}(\mathrm{x}) = 1 &+ x^{32} + x^{37} + x^{72} + x^{102} + x^{128} + x^{44}x^{60} + x^{61}x^{125} + x^{63}x^{67} \\ &+ x^{69}x^{101} + x^{88}x^{80} + x^{110}x^{111} + x^{115}x^{117} + x^{46}x^{50}x^{58} \\ &+ x^{103}x^{104}x^{106} + x^{33}x^{35}x^{36}x^{40} \end{aligned}$$

Đầu ra của hàm NFSR này sẽ là

$$\begin{aligned} b_{i+128} = {} & s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} \\ & + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84} \\ & + b_{i+88}b_{i+92}b_{i+93}b_{i+95} + b_{i+22}b_{i+24}b_{i+25} + b_{i+70}b_{i+78}b_{i+82} \end{aligned}$$

Bộ lọc: $\mathrm{h}_{128}(\mathrm{x}) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$

Tiền đầu ra:

$$y_i = \sum_{i \in A_{128}} b_{k+i} + s_{93+i} + \mathrm{h}_{128}(x)$$

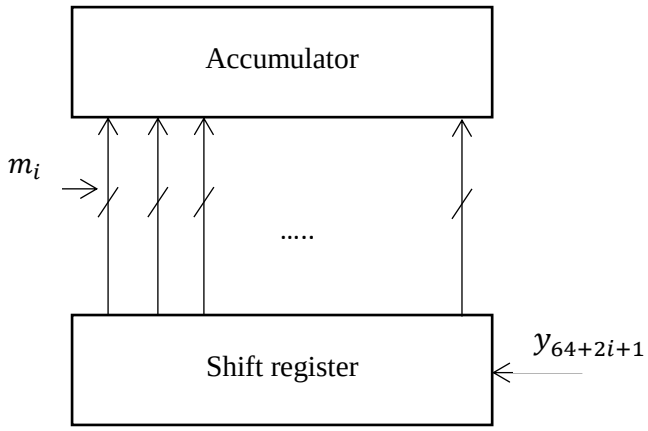
Trong đó $A_{128} = \{2, 15, 36, 45, 64, 73, 89\}$.

Khi đó, đầu ra của Grain-128a sẽ là $z_i = \mathrm{y}_{64+2\mathrm{i}}$

Chúng ta chọn đầu ra là bit thứ hai của mật mã sau khi bỏ qua 64 bits đầu tiên. 64 bits đầu tiên cùng phần còn lại sẽ được sử dụng cho việc xác thực. Giả sử chúng ta có một thông điệp với độ dài L được định nghĩa bởi các bit $\mathrm{m}_0, \dots, m_{L-1}$. Đặt $m_{\mathrm{L}} = 1$. Chú ý rằng m_{L} là padding, điều này rất quan trọng với an ninh của việc xác thực vì nó đảm bảo rằng m và m||0 là hai thẻ khác nhau. Để cung cấp chứng thực, hai thanh ghi 32 bits được sử dụng, một thanh ghi được gọi là "accumulator" và một thanh ghi là "shift register". Nội dung của thanh ghi "accumulator" tại thời điểm i được định nghĩa bởi $a_i^0, \dots, a_i^{31}$. Nội dung của thanh ghi "shift register" được định nghĩa bởi $r_i, \dots, r_{31}$.Thanh ghi "accumulator" được khởi tạo thông qua $r_i = y_i, 0 \le i \le 31$ và thanh ghi "shift register" được khởi tạo bởi $a_i^0 = y_{32+j}, 0 \le j \le 31$. Hai thanh ghi này sẽ lần được thay đổi bởi $r_{i+32} = y_{64+2i+1}$ và $a_{i+1}^j = a_i^j + m_i r_{i+j}$ với $0 \le j \le 31$ và

$0 \le i \le L$. Kết quả của "accumalator" $a_{L+1}^0, \dots, a_{L+1}^{31}$ là thẻ được sử dụng cho chứng thực. Ta viết $t_i = a_{L+1}^i, 0 \le i \le 31$. Hình dưới mô tả cơ chế xác thực của Grain-128a.

Để đảm bảo việc sử dụng các thẻ ngắn hơn, ta có thể sử dụng thẻ w bits thông qua $t_i^{(w)} = t_{32-w+i}, 0 \le i \le w-1$.



Hình 2-2: Cơ chế xác thực của Grain-128a

So sánh việc triển khai mà không có xác thực trong việc tạo ra 1 bit cho mỗi clock tính toán, Grain-128a thể hiện sự linh hoạt và an ninh nổi trội hơn so với Grain-128. Grain-128 yêu cầu 2133 cổng tương đương để thực hiện thiết kế cơ bản tạo ra 1 bit khóa dòng với mỗi clock. Trong khi đó Grain-128a có thể tạo ra 2 bits cho mỗi clock với số lượng cổng tương đương là 2243 (tăng 5%). Mặt khác Grain-128 được khởi tạo trong 256 clocks, nhưng Grain-128a (ở chế độ 2x) có thể tạo ra khóa dòng chỉ sau (256 + 64) / 2 = 160 clocks. Có thể nói Grain-128a đắt hơn Grain-128 một chút nhưng đem lại bảo mật tốt hơn và cung cấp khả năng xác thực.

Tấn công Grain-128a

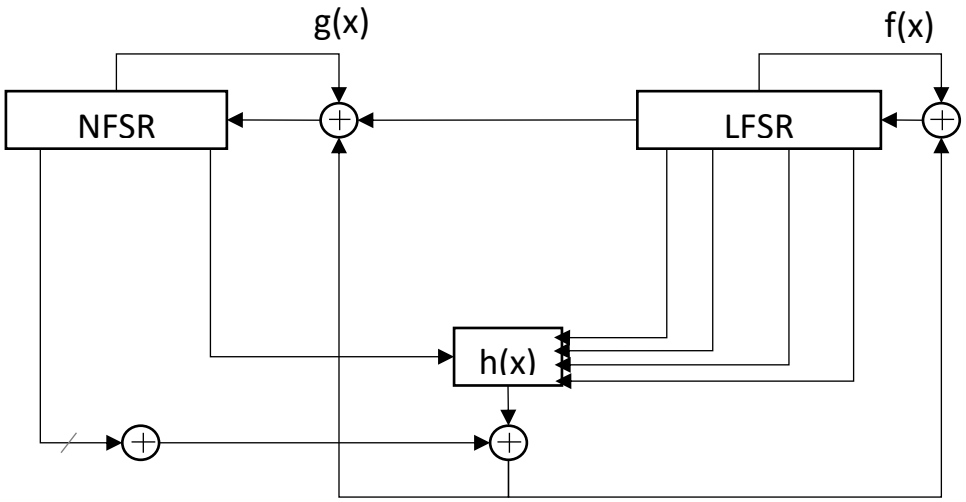
Trong Grain-128a, 64 bits đầu tiên không thể bị kẻ tấn công truy cập nếu bật chế độ xác thực. Banik, Maitra và Sarkar đề xuất một tấn công khác biệt lỗi [6] nhắm vào mục tiê là MAC thay vì khóa dòng thông thường. Cuộc tấn công này yêu cầu 2^{11} lỗi và 2^{12} thế hệ của MAC để truy cập vào khóa. Một cuộc tấn công thứ hai được đề xuất bởi Ding và Guan [4] đòi hỏi 2^{96} lựa chọn IV và $2^{103.613}$ bits khóa để phục hồi 128 bits khóa với độ phức tạp tính toán là $2^{96.322}$.

2.2.5. Tạo khóa

Trước khi tạo ra bất kỳ khóa dòng nào, hệ mã hóa cần khởi tạo khóa và giá trị IV. Khóa sẽ có k bits $k_i, 0 \le i \le \mathrm{k}-1$ và các bit của giá trị IV được xác định bởi $IV_i, 0 \le i \le \mathrm{l}-1$.

Để khởi tạo khóa, đầu tiên ta sử dụng NFSR với khóa $b_i = k_i, 0 \le i \le \mathrm{k}-1$, tiếp tục sử dụng 64 bits đầu tiên của LFSR với giá trị IV là $s_i = IV_i, 0 \le i \le \mathrm{l}-1$. Các bits còn ại của LFSR được xác định bởi $s_i = 1_i, \mathrm{l} \le i \le \mathrm{k}-1$. Tiếp theo, thuật

toán mã hóa được thực hiện 2k lần nhưng không sinh đầu ra trong bất kỳ lần chạy nào, thay vào đó hàm đầu ra sẽ đưa kết quả trở lại và XOR với đầu vào của cả LFSR và NFSR.



Hình 2-3: Quá trình tạo khóa của Grain

2.3. Nguyên lý thiết kế

2.3.1. Tiêu chuẩn thiết kế

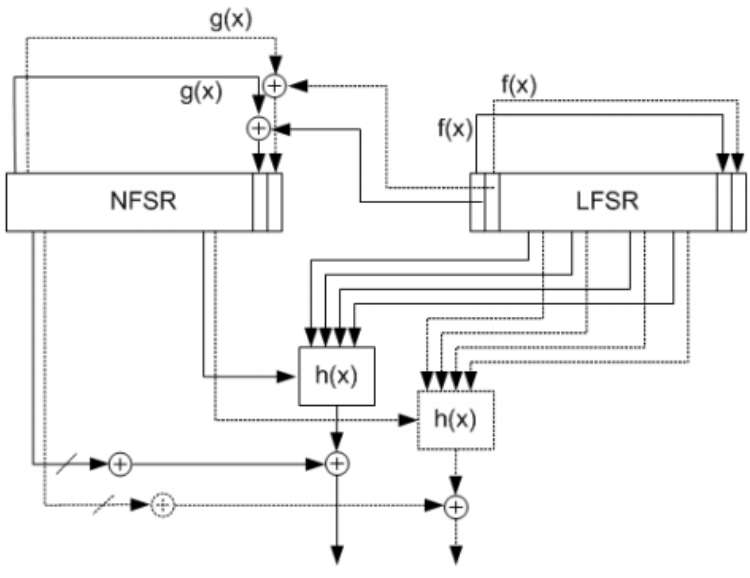
Thiết kế của giải thuật này được chọn là đơn giản nhất để có thể thực hiện trên phần cứng. Các yêu cầu bảo mật tương đương với mức độ tính toán là 2^{80}. Để đáp ứng yêu cầu này, giải thuật cần xây dựng hệ mật mã với bộ nhớ 160 bits. Việc thực hiện 160 bits trong bộ nhớ của phần cứng là một giới hạn dưới cho sự phức tạp tính toán. Để phát triển một thiết kế phần cứng nhỏ, chúng ta phải tập trung vào việc giảm thiểu các chức năng được sử dụng đối với bộ nhớ này. Các chức năng được sử dụng phải đủ nhỏ để có thể tiết kiệm được Gates nhưng vẫn phải đủ lớn để đảm bảo được độ an ninh cao. Ta biết rằng một LFSR với đa thức phản hồi nguyên thủy mức d có thể cho ra đầu ra với chu kỳ $2^d – 1$. Trong thuật toán này, LFSR có kích thước 80 bits và đa thức phản hồi là nguyên thủy nên nó đảm bảo xác suất trùng lặp của LFSR là $1 – 2^{-80}$ hay chu kỳ là $2^{80} – 1$. Vì thuật toán này sử dụng NFSR với đầu vào được kết hợp với đầu ra của LFSR nên chu kỳ trùng lặp còn phụ thuộc vào khóa IV được sử dụng. Việc kết hợp đầu vào của NFSR với đầu ra của LFSR đảm bảo sự cân bằng cho trạng thái của NFSR. Grain có chức năng lọc khá nhỏ chỉ với 5 biến số và phi tuyến 12. Tuy nhiên sự lo sợ về an ninh ở đây được bù đắp một phần bởi thực tế là một trong các đầu vào được lấy từ NFSR. Các bits đầu vào lấy từ NFSR sẽ phụ thuộc phi tuyến vào các bits trạng thái khác, kể cả từ NFSR lẫn từ LFSR. Chức năng lọc nhỏ cũng được bù đắp bằng cách them 7 bits tuyến tính từ NFSR tại các vị trí thích hợp để tạo thành đầu ra.

Trong giai đoạn khởi tạo, mục tiêu quan trọng là lấy được nội dung của thanh ghi thay đổi trước khi khóa chạy được tạo ra. Số lượng lần thực hiện khởi tạo chính là sự

cân bằng giữa an ninh và tốc độ. Nếu một mật mã được khởi tạo lại thường xuyên với một IV mới thì việc khởi tạo bị thắt cổ chai là khó tránh khỏi. Trước khi khởi tạo LFSR chứa IV và 16 bits khác. Đối với việc khởi tạo ra 2 IV khác nhau (khác nhau chỉ bằng 1 bit) xác suất để một bit đăng ký thay đổi là như nhau cho cả 2 khởi tạo là gần 0.5 sau 2k lần thực hiện khởi tạo. Cuối cùng không có điểm yếu nào được chèn vào bởi nhà thiết kế.

2.3.2. Tốc độ thực hiện

Cả hai thanh ghi được thay đổi đều đặn nên thuật toán này sẽ sinh ra 1 bit sau mỗi lần thực hiện. Ta có thể tăng tốc độ của thuật toán bằng cách đánh đổi chi phí phần cứng nhiều hơn nữa. Điều này có thể dễ dàng được thực hiện chỉ với việc thực hiện các chức năng phản hồi, f(x), g(x) và hàm ra nhiều lần hơn nữa. Để đơn giản hóa việc thực hiện, 15 bits cuối cùng của thanh ghi dịch chuyển, $s_i,\ \mathrm{l} + 1 \leq i \leq \mathrm{k} - 1$ và $b_i,\ \mathrm{l} + 1 \leq i \leq \mathrm{k} - 1$ không được sử dụng cho các chức năng phản hồi hay đầu vào cho hàm lọc. Điều này cho phép tăng tốc độ lên đến k/4 lần nếu phần cứng đáp ứng đủ nhu cầu. Một ví dụ về việc tăng tốc độ gấp đôi được thể hiện trong hình dưới đây. Các thanh ghi cần được cài đặt sao cho mỗi bit sẽ thay đổi t bước thay vì một như lúc trước để tăng tốc độ lên t lần. Bằng các tăng tốc độ k/4 lần thì sau mỗi clock, thuật toán sẽ tạo ra k/4 bits. Trong quá trình khởi tạo, mật mã được thực hiện 2k lần nên khả năng tăng tốc độ bị giới hạn bởi <= k/4 và phải bị chia hết bởi 2k. Như vậy số lượng lần lặp được sử dụng trong việc khởi tạo khi đó sẽ là 2k/t. Vì bộ lọc và các chức năng phản hồi nhỏ nên việc tăng tốc độ như vậy là khả thi, có thể thực hiện trong thực tế.



Hình 2-4: Thuật toán Grain với tốc độ tăng gấp đôi

2.3.3. Phức tạp phần cứng

Trong [10] và [11], các nhà nghiên cứu cũng đã đưa ra những chỉ dẫn thực tế về tính phức tạp và những tính năng quan trọng khác có thể thực hiện ở phần cứng của Grain dựa trên kiến trúc FPGA tiêu chuẩn với hai họ ALTERA MAX II và ALTERA Cyclone. Hai hệ thống này cho phép thực hiện mật mã với tốc độ cao hơn và nó cũng cho phép thực hiện mật mã Grain với tốc độ tăng 16 lần so với ban đầu, tức là với t = 16.

Số cổng cho mỗi chức năng phụ thuộc và độ phức tạp và tính năng thực hiện. Các kết quả đưa ra không phải là hằng số tự nhiên mà phụ thuộc vào việc thực hiện trên một chip thực tế. Với số cổng là 8 cho một flip flop, chúng ta có thể thấy sự khác nhau của việc triển khai Grain với mỗi chức năng và với mỗi phần cứng khác nhau qua 2 bảng mô tả dưới đây.

Bảng 2-1: Số cổng của Grain đối với các chức năng khác nhau

Chức năng	Số cổng
D flip flop	8
NAND2	1
NAND3	1.5
NAND4	2
NAND5	2.5
NAND6	3
XOR2	2.5
MUX3	5

Bảng 2-2: Số cổng và tốc độ của Grain với các giá trị t khá nhau

t	Số cổng	Tốc độ (Mb/s)		
		MAX 3000A	MAX II	Cyclone
1	1450	49	200	282
2	1637	98.4	422	576
4	2010	196	632	872
8	2756	-	1184	1736
16	4248	-	2128	3136

2.4. Một số cải tiến hệ mật Grain

Tăng tốc thông lượng nhờ công nghệ lượng tử

Việc sử dụng công nghệ automata di động lượng tử (QCA)[6] cho việc thiết kế các mạch logic đã cho thấy tăng tốc độ truyền dữ liệu lên đến 2 THz. Trong công nghệ

[6] Công nghệ di động lượng tử là một lĩnh vực mới của vật lý kỹ thuật, cho phép chuyển tiếp một số tính năng của cơ học lượng tử, cho phép giải quyết các vấn đề phức tạp một cách nhanh hơn nhiều lần phương pháp thông thường

QCA, các mạch được thiết kế để có một kích thước đặc biệt siêu nhỏ cũng như tiêu thụ điện năng cực thấp. Grain-128 là một trong những mật mã dòng tốt nhất trong danh sách cuối cùng của dự án eSTREAM. Trong nghiên cứu [2], Reza Sabbaghi-Nadooshan, Zahra Shahosseini và Davood Rezaeipour đã thiết kế và mô phỏng các khối chính của thuật toán này bao gồm cả cổng XOR, đăng ký thay đổi hồi quy tuyến tính (LFSR) và NLFSR (nonlinear-feedback feedback record) sử dụng công nghệ QCA. Thiết kế của các khối này sử dụng mô phỏng QCA Designer được đưa ra và các yếu tố chính như diện tích, độ phức tạp và độ trễ được ước tính. Hơn nữa, phần mềm ModelSim được sử dụng để mô phỏng mô hình HDLQ của Thuật toán mật mã dòng hạt Grad-128 QCA. Kết quả cho thấy các thông số chính của hạt Grain-128 được đề xuất như diện tích và thông lượng cũng được cải thiện.

2.5. Phân tích Grain

2.5.1. So sánh các phiên bản trong họ Grain

Trong phần này, luận văn đưa ra những so sánh các thông số thiết kế khác nhau cho các thành viên của họ Grain. Trong Bảng 2-3 chúng ta có thể thấy sự khác nhau về độ dài khóa, kích thước IV và padding sử dụng trong các thành viên của Grain.

Bảng 2-3: Độ dài khóa và IV của họ Grain

Hệ mật	Chiều dài khóa	Kích thước IV	Padding với IV
Grain V0	80	64	FFFF
Grain V1	80	64	FFFF
Grain-128	128	96	FFFFFFFF
Grain-128a	128	96	FFFFFFFE

Chỉ có phiên bản cuối cùng của họ Grain – Grain-128a có padding được hoàn thiện bởi việc ngoại trừ bit bên phải của LFSR, thay bởi giá trị 0 để tránh cuộc tấn công đồng bộ hóa được đề xuất bởi Kucuk. Trong tất cả các phiên bản khác padding được thực hiện với tất cả các bit.

Bảng 2-4 mô tả những cập nhật của các hệ mật trong họ Grain với hai thanh ghi LFSR và NFSR

Bảng 2-4: Hàm cập nhật của họ Grain

Hệ mật	Cập nhật của LFSR	Cập nhật của NFSR

Grain V0	$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i$	$b_{i+80} = s_i + b_{i+62} + b_{i+60} + b_{i+52} + b_{i+45} + b_{i+37} + b_{i+33} + b_{i+28} + b_{i+21} + b_{i+14} + b_{i+9} + b_i + b_{i+63}b_{i+60} + b_{i+37}b_{i+33} + b_{i+15}b_{i+9} + b_{i+60}b_{i+52}b_{i+45} + b_{i+33}b_{i+28}b_{i+21} + b_{i+63}b_{i+45}b_{i+28}b_{i+9} + b_{i+60}b_{i+52}b_{i+37}b_{i+33} + b_{i+63}b_{i+60}b_{i+21}b_{i+15} + b_{i+63}b_{i+60}b_{i+52}b_{i+45}b_{i+37} + b_{i+33}b_{i+28}b_{i+21}b_{i+15}b_{i+9} + b_{i+52}b_{i+45}b_{i+37}b_{i+33}b_{i+28}b_{i+21}$
Grain V1	$s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i$	$b_{i+80} = s_i + b_i + b_{i+9} + b_{i+14} + b_{i+21} + b_{i+28} + b_{i+33} + b_{i+37} + b_{i+45} + b_{i+52} + b_{i+60} + b_{i+62} + b_{i+9}b_{i+15} + b_{i+33}b_{i+37} + b_{i+60}b_{i+63} + b_{i+21}b_{i+28}b_{i+33} + b_{i+45}b_{i+52}b_{i+60} + b_{i+15}b_{i+21}b_{i+60}b_{i+63} + b_{i+33}b_{i+37}b_{i+52}b_{i+60} + b_{i+9}b_{i+28}b_{i+45}b_{i+63} + b_{i+9}b_{i+15}b_{i+21}b_{i+28}b_{i+33} + b_{i+37}b_{i+45}b_{i+52}b_{i+60}b_{i+63} + b_{i+21}b_{i+28}b_{i+33}b_{i+37}b_{i+45}b_{i+52}$
Grain-128	$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}$	$b_{i+128} = s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84}$
Grain-128a	$s_{i+128} = s_i + s_{i+7} + s_{i+38} + s_{i+70} + s_{i+81} + s_{i+96}$	$b_{i+128} = s_i + b_i + b_{i+26} + b_{i+56} + b_{i+91} + b_{i+96} + b_{i+3}b_{i+67} + b_{i+11}b_{i+13} + b_{i+17}b_{i+18} + b_{i+27}b_{i+59} + b_{i+40}b_{i+48} + b_{i+61}b_{i+65} + b_{i+68}b_{i+84} + b_{i+88}b_{i+92}b_{i+93}b_{i+95} + b_{i+22}b_{i+24}b_{i+25} + b_{i+70}b_{i+78}b_{i+82}$

Bảng 2-5 so sánh số lượng cổng khác nhau của các thành viên trong họ Grain và phản ánh sự phức tạp về phần cứng của thiết kế.

Bảng 2-5: Số cổng của họ Grain khi thực hiện với phần cứng

Hệ mật	Số cổng cho LFSR	Số cổng cho NFSR	Số cổng cho hàm đầu ra	Tổng số cổng
Grain V0	640	640	NA	1435
Grain V1	640	640	NA	1450
Grain-128	1024	1024	35.5	2133
Grain-128a không xác thực	1024	1024	35.5	2145.5
Grain-128a có xác thực	1024	1024	35.5	2769.5

Vì thiết kế của Grain V0 và Grain V1 tương tự nhau nên tổng số cổng đếm cũng tương tự nhau. Grain 128a không xác thực chỉ đòi hỏi nhiều hơn 12.5 cổng so với Grain-128. Điều đó có nghĩa là Grain-128a có thể sử dụng hiệu quả mà không xác thực với sự phức tạp phần cứng tương đương Grain-128 mà đem lại độ an toàn cao hơn. Grain-128a với tính xác thực đòi hỏi nhiều hơn khoảng 30% số cổng, nghĩa là nó không yêu cầu quá nhiều phần cứng bổ sung cho quá trình xác thực.

Trong Bảng 2-6 so sánh các thành viên của họ Grain trên cơ sở thời gian thiết lập khóa, thời gian thiết lập IV và tốc độ mã hóa. Các tốc độ mã hóa này được đo bằng các bộ xủ lý Pentium 4 2.80 Ghz cho hai loại dữ liệu, một cho các luồng dài và một cho các luồng dữ liệu ngắn dưới 40 bytes.

Bảng 2-6: Hiệu suất của họ Grain

Hệ mật	Thời gian thiết lập khóa	Thời gian thiết lập IV	Tốc độ mã hóa	
			Luồng dài	Luồng 40 bytes
Grain V0	29.27	73408.44	3729.79	5545.83
Grain V1	31.14	1498.23	57.31	102.95
Grain-128	38.89	1098.61	31.16	70.30

Có thể thấy Grain-128 có sự vượt trội về tốc độ tính toán, và hiệu quả phần cứng cao nhất trong họ Grain.

2.5.2. So sánh Grain với một số hệ mã hóa nhẹ khác

Khi thiết kế một hệ mật mã, người thiết kế cần phải tập trung vào một số điểm đặc biệt của giải thuật. Không thể thực hiện một thiết kế hoàn hảo, đáp ứng được tất cả các mong muốn của một hệ mật mã như đáp ứng được tất cả các độ dài bản rõ, tất cả các ứng dụng, tất cả các hạn chế về bộ nhớ… Grain được thiết kế cho một phần cứng rất nhỏ, sử dụng ít cổng nhất có thể trong khi vẫn đáp ứng được an ninh cao. Hệ mật này được sử dụng trong môi trường có cổng đếm, điện năng tiêu thụ, bộ nhớ cần thiết là rất nhỏ. Ngoài ra Grain vẫn có thể được sử dụng trong phần mềm nói chung nếu thực hiện tăng tốc độ cho nó. Nhưng việc so sánh hiệu suất của Grain trong các ứng dụng phần mềm là không có ý nghĩa. Luận văn chỉ so sánh hiệu suất của Grain với một số giải thuật khác trong việc ứng dụng vào phần cứng.

Thuật toán này cho phép thực hiện song song 16 mã hóa khác nhau, cho phép triển khai nhanh hơn, với chi phí sử dụng ít hơn nhưng đem lại hiệu quả cao hơn. Tính hiệu quả của phần cứng là tỷ lệ thông lượng với điện tích sử dụng trong thuật toán. Nhìn vào bảng thống kê dưới, ta có thể thấy thuật toán Grain có tính hiệu quả phần cứng cao hơn Trivium (77.28 > 38.48).

Hệ mật	Số bit khóa	Số bit khối	Chu kỳ xung nhịp trên một khối	Thông lượng ở 100 MHz (Kbps)	Xử lý logic (µm)	Diện tích (GEs)
Mã khối						
PRESENT	80	64	32	200	0,18	1.570
HIGHT	128	64	34	188	0,25	3.048
mCrypton	96	64	13	492	0,13	2.681
Mã dòng						
Trivium	80	1	1	100	0,13	2.599
Grain	80	1	1	100	0,13	1.294

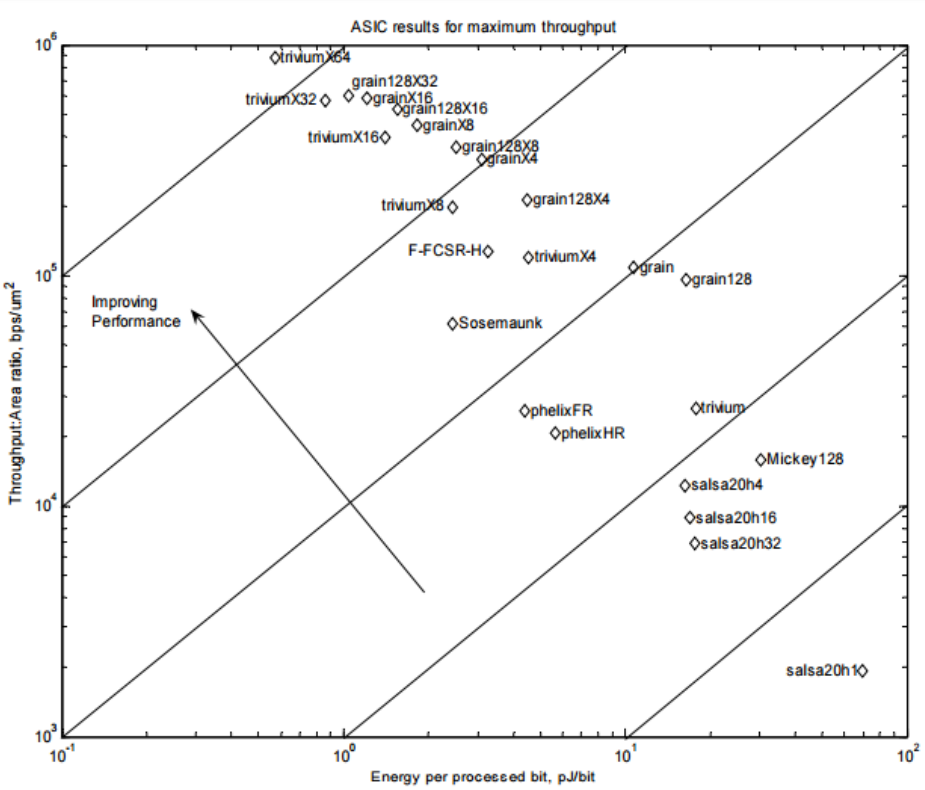
Với kích thước khóa là 80 bits và kích thước IV là 64 bits, các cuộc tấn công vào hệ mã này để tìm kiếm chìa khóa đầy đủ cần có yêu cầu phức tạp tính toán không thấp hơn 2^{80}.

Trong phiên bản gốc v0, tác giả khẳng định: “Grain cung cấp một bảo mật cao hơn so với một số thuật toán mã hóa cũng được biết đến khác, dự định sẽ được sử dụng trong các ứng dụng phần cứng. Ví dụ như trong mã hóa của E0 được sử dụng trong Bluetoot và A5/1 sử dụng trog GSM. So với E0 và A5/1, Grain cung cấp sự bảo mật cao hơn trong khi yêu cầu một phần cứng nhỏ hơn”. Ví dụ một cuộc tấn công chống lại E0 [24] đòi hỏi sự phức tạp tính toán của 2^{40} và 2^{35} khung hình độ dài 2745

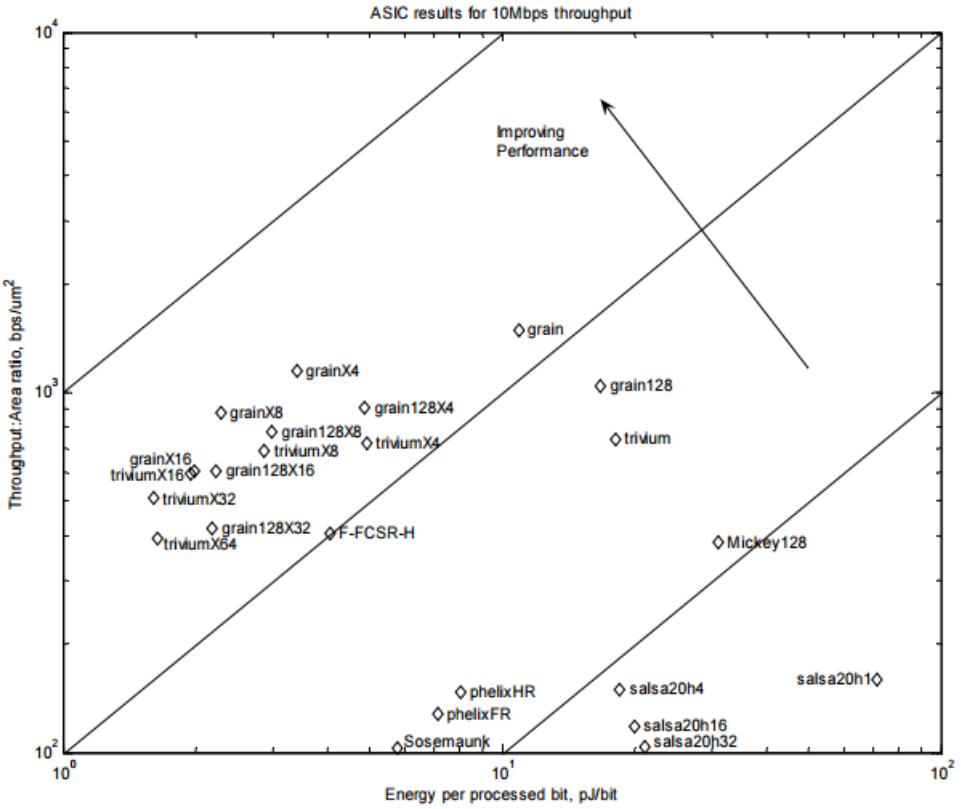
bits. Tuy nhiên để xác định 80 bits của khóa Grain, cần đến sự phức tạp tính toán 2^{43} và 2^{38} bits khóa dòng trong cuộc tấn công phục hồi khóa [1].

Trong phiên bản sửa đổi Grain v1, chức năng lọc nhỏ, chỉ cần 5 biến và 12 phi tuyến nhưng vẫn đạt được hiệu quả cao nhờ sự bù đắp bởi một trong các yếu tố đầu vào (7 bits tuyến tính) được lấy từ NFSR.

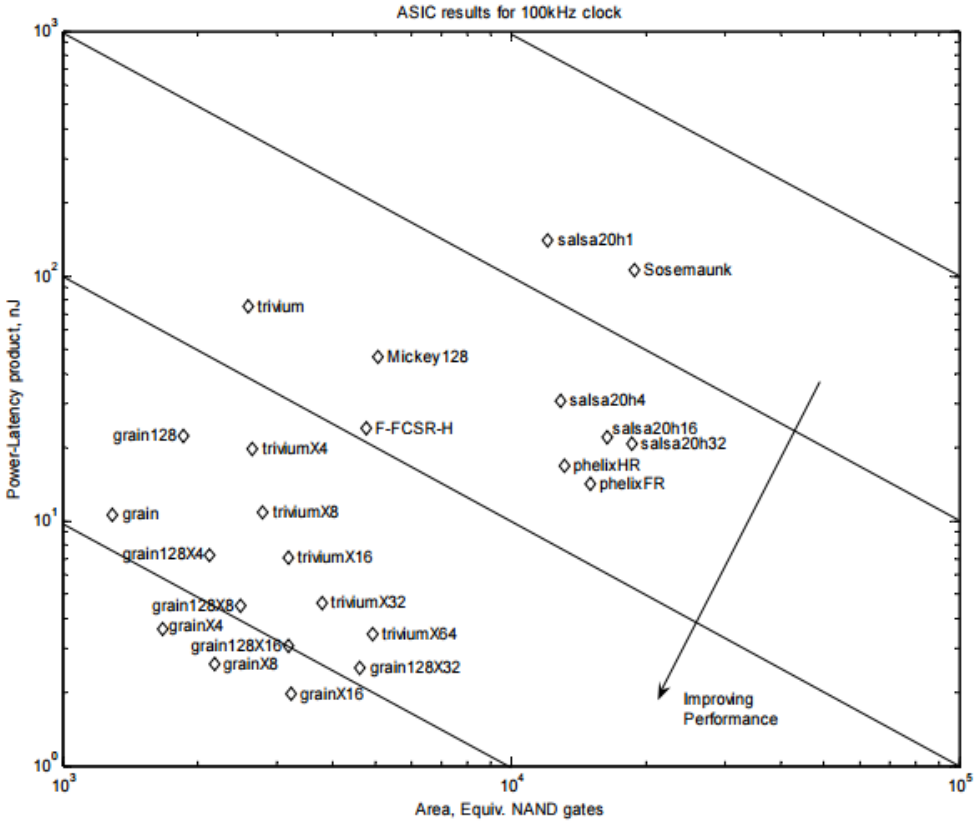
So với các thuật toán mã hóa dòng nhẹ khác, Grain cũng có ưu thế hơn trong thông lượng, hiệu suất sử dụng, phù hợp với các ứng dụng sử dụng WLAN, RFID/WSN. Thực nghiệm của Good, T., & Benaissa, M [17] đã chỉ rõ điều này.



Hình 2-5: Lưu lượng tối đa của các mật mã dòng nhẹ với thiết kế 0.13 µm Standard Cell CMOS



Hình 2-6: Hiệu suất của các giải thuật mật mã dòng nhẹ đối với mạng Wireless-LAN 10Mbps



Hình 2-7: Hiệu suất cho ứng dụng RFID / WSN cấp thấp đồng hồ 100kHz

Bảng 2-7: Khả năng ứng dụng của mật mã dòng nhẹ

Ứng dụng	Kích thước khóa	Ưu tiên sử dụng mật mã từ góc độ phần cứng	
		Ưu tiên nhất	Ưu tiên thứ hai
WLAN	80	Grain và Trivium	F-FCSR-H
RFID/WSN	80	Grain và Trivium	-
WLAN	128	Grain128	Mickey128
RFID/WSN	128	Grain128	-
WLAN	256	Sosemanuk và Phelix	Salsa20

Trong triển khai cơ bản, Grain có tốc độ 1 bit / 1 clock. Thông thường tốc độ của từ được định hướng mã hóa thường cao hơn tốc độ 1 word / 1 clock. Grain được tập trung phát triển để ứng dụng cho phần cứng nhỏ và điều này cũng được bù đắp bởi khả năng tăng tốc độ với chi phí phần cứng nhiều hơn. Điều này cho phép nhà cung cấp có thể lựa chọn tốc độ bảo mật với khả năng phần cứng có thể có để đem lại hiệu quả cao nhất trong thực tế triển khai.

2.5.3. Điểm yếu

Cũng như những hệ mã hóa khác, họ Grain cũng chứa đựng những lỗ hổng nguy hiểm. Luận văn đã tìm hiểu một số phương pháp tấn công vào Grain. Tuy nhiên khi áp dụng vào một hệ thống yêu cầu tính nhỏ gọn thì độ an toàn có thể cân nhắc ở một mức độ "đủ" nào đó.

Tấn công đại số

Các cuộc tấn công đại số lên mật mã dòng đã nhận được nhiều chú ý gần đây bởi chúng có thể mang lại hiệu quả cao nếu nhà thiết kế không cẩn thận. Bộ lọc của một máy khởi tạo chỉ sử dụng một LFSR và hàm Boolean phi tuyến $h(x)$ có thể rất dễ bị tấn công đại số. Tuy nhiên trong Grain, một NFSR được sử dụng để đưa tính phi tuyến vào hàm $h(x)$. Giải phương trình cho trạng thái 256 bits ban đầu khó có thể thực hiện được với cập nhật phi tuyến của NFSR. Độ đại số của bit đầu ra thể hiện ở các bit trạng thái đầu vào sẽ lớn, và thay đổi theo thời gian. Điều này giúp đánh bại bất kỳ cuộc tấn công đại số nào vào mật mã.

Phương pháp tấn công tính toán giá trị Key-IV yếu

Các phiên bản Grain sử dụng thanh ghi dịch hồi tuyến tính không chỉ để đảm bảo tính thống kê mà còn có những ràng buộc thấp hơn cho quá trình tạo keystream. Nhưng đây cũng chính là điểu yếu của thuật toán này - Key – IV. Thuật toán sử dụng kết hợp thanh ghi dịch hồi tuyến tính và thanh ghi phản hồi phi tuyến làm đầu vào cho quá trình mã hóa. Tuy nhiên, nếu tất cả các giá trị ban đầu của LFSR đều bằng 0 thì chỉ còn lại NFSR là khối duy nhất có tác dụng mã hóa. Điều này chính là một điểm

yếu, trình tự một keystream tạo ra bởi NFSR rất dễ bị tấn công qua các phương pháp thông dụng như xấp xỉ tuyến tính, chu kỳ ngắn. Chính vì thế nếu cặp khóa và IV được tạo ra từ LFSR với các giá trị khởi tạo đều bằng 0 thì cặp khóa – IV này là một cặp khóa – IV yếu. Trong thực tế, các nhà thiết kế cố gắng sử dụng k-l bit 1 làm đầu vào cho LFSR trước khi khởi tạo khóa. Tuy nhiên sau quá trình phân tích mã, ta nhận ra rằng, sau 2k lần chạy, trạng thái của LFSR có thể trở về 0.

Chúng ta lấy Grain v1 làm ví dụ để tìm ra điểm yếu của các thuật toán Grain này. Theo định nghĩa ta có LFSR được biểu diễn bằng $S_t = (s_i, s_{i+1}, \dots, s_{i+79})$ và NFSR được mô tả bằng $B_t = (b_i, b_{i+1}, \dots, b_{i+79})$. Gọi G và F là hàm chuyển đổi: $B_{t+1} = \mathrm{G}(B_t)$ và $S_{t+1} = \mathrm{F}(S_t)$. Khi đó, quá trình khởi tạo khóa có thể coi là quá trình chuyển đổi từ B_0 và S_0 thành B_{160} và S_{160}. Một cặp Key-IV yếu khi $S_{160} = (0, 0, \dots, 0)$. Dưới đây là thuật toán để tính toán ra được giá trị Key-IV yếu.

1. Khởi tạo $S_{160} = (0, 0, \dots, 0)$. Và chọn B_{160} ngẫu nhiên.
2. Lặp từ $t = 159$ đến $t = 0$
   a) Tính $z_s^t = \sum_{i \in A_1} b_{t+i} + h_1(s_{t+3}, s_{t+25}, s_{t+46}, s_{t+64}, b_{t+63})$
   b) Tính $s_t = z_s^t + s_{t+80} + s_{t+62} + s_{t+51} + s_{t+38} + s_{t+23} + s_{t+13})$
   c) Tính $b_t = z_s^t + b_{80+t} + s_t + P(B_t)$. Trong đó $P(B_t)$ là biểu thức của 79 biến $b_{i+1}, \dots, b_{i+79}$
3. Với $j = 64, \dots, 79$, nếu $S_i = 1$ thì dừng lại, nếu không quay lại bước 1

Thực hiện thuật toán này khoảng 2^{20} lần, chúng ta có thể tìm ra được 16 cặp Key-IV yếu. Tương tự với thuật toán Grain V0 và Grain-128 chúng ta có bảng sau:

Version	Grain v0	Grain v1
Key	0x6f22a2a70e1c363b62af	0xf57e358ecae6b3dc683d
IV	0x44b604a4d4479eb4	0x97652a7f1a112415
B_{160}	0xc2ced7db3189a9ad94b8	0xd99ea5abb8d0129212c7
S_{160}	0x00000000000000000000	0x00000000000000000000
Version	**Grain-128**	
Key	0xfd6af0ff0ad9bdad7037b91ef1b9cc13	
IV	0x014d3e274f8d3528ddad4310	
B_{160}	0xc1bc1c087a79b533f9018d230df2e744	
S_{160}	0x00000000000000000000000000000000	

Hình 2-8: Điểm yếu của giá trị IV trong Grain

Với phương pháp này Walsh tìm ra $2^{64}/2^{64}/2^{96}$ key – IV yếu trong tổng số $2^{144}/2^{144}/2^{224}$ key – IV và để tìm ra được các key – IV yếu cần $2^{12.6}/2^{44.2}/2^{86}$ bits khóa dòng và $2^{15.8}/2^{47.5}/2^{104.2}$ phép tính cho mỗi Grain v0, Grain v1 và Grain-128.

Phương pháp tấn công khôi phục Key-IV

Phương pháp tấn công khôi phục Key – IV cũng là một trong các phương pháp tấn công thường được áp dụng cho Grain. Với việc sử dụng các phương pháp Grobner, XL Zhuang-Zi để giải quyết bài toán NP-khó trong quá trình tìm Key-IV, người ta có thể phân tích đại số và tìm ra được một số bit của khóa khi đã biết trước một số bit. Dưới đây là phân tích đại số của các phiên bản Grain do Haina Zhang và Xiaoyun Wang đưa ra [9].

Version	Unknown bits	Degree of $g(x)$	Degree of $h(x)$	No. of bits guessed	No. of bits recovered	Time to find solution	Keystream bits used
Grain v1	160	6	3	83	77	0.204 sec	150
Grain-128	256	2	3	192	64	0.906 sec	100

Version	Unknown bits	Degree of $g(x)$	Degree of $h(x)$	No. of bits guessed	No. of bits recovered	Time to find solution	Keystream bits used
Grain v1	80	6	1	3	77	0.204 sec	150
Grain-128	128	2	0	64	64	0.906 sec	100

Với phương pháp này, hai nhà khoa học đã có thể khôi phục các khóa bí mật 150 bits trong khoảng 2 giây cho Grain v0, Grain v1 và tìm ra chìa khóa của Grain-128 với khoảng 100 bits sau $2^{93.8}$ phép tính.

Dynamic Cube Attacks

Mới đây cũng có một phương pháp tấn công mới do Itai Dinur and Adi Shamir đề xuất để phá vỡ cấu trúc của Grain-128: Dynamic Cube Attacks [7]. Khác với các phương pháp truyền thống, tìm ra chìa khóa bằng cách giải hệ phương trình tuyến tính với các bit quan trọng, phương pháp này tìm ra khóa bí mật bằng cách khai thác các kết quả thu được từ cube tester. Cuộc tấn công khối động này có thể tạo ra được các đại diện mật mã yếu, có thể tránh được các phương pháp chống lại tấn công trước đây. Cuộc tấn công đầu tiên của Itai Dinur and Adi Shamir chạy trong thực tế có thể khôi phục toàn bộ 128 bits của Grain khi số lượng vòng khởi tạo của Grain-128 giảm xuống 207. Cuộc tấn công thứ hai có thể phá vỡ Grain-128 sau 250 vòng thực hiện và nhanh hơn phương pháp tìm kiếm vét cạn khoảng 228 lần.

Một số phương pháp tấn công khác

Ngoài những phương pháp trên, việc tấn công vào hệ mật Grain còn là niềm đam mê của nhiều nhà nghiên cứu. Với phương pháp tấn công bằng đại số điển hình vào mật mã dòng, kẻ thám mã có thể dò ra được đầu ra của hàm NFSR và LFSR. Hay cuộc tấn công Time/Memory/Data Tradeoff có thể phá mã Grain với độ phức tạp tính toán là $O(2^{80})$...

Chương 3. MÃ HÓA GRAIN TRÊN THIẾT BỊ RASPBERRY

3.1. Mô tả bài toán

Nhà thông minh – Smart home là một ngôi nhà được trang bị hệ thống tự động điều khiển đèn chiếu sáng, nhiệt độ, ti vi, tủ lạnh, an ninh, rèm cửa, cửa ra vào … nhằm làm cho cuộc sống tiện nghi, an toàn hơn đồng thời góp phần sử dụng hợp lý các nguồn tài nguyên. Một trong những ứng dụng cơ bản của nhà thông minh là một hệ thống điều khiển nhiệt độ phù hợp với nhu cầu sử dụng, chẳng hạn như đưa cảnh báo khi nhiệt độ vượt ngưỡng nào đó và điều chỉnh điều hòa nhiệt độ phù hợp. Luận văn sử dụng thiết bị Raspberry để thu thập dữ liệu từ cảm biến SHT11 dùng để đo nhiệt độ, độ ẩm, trạng thái cửa ra vào của ngôi nhà, phòng làm việc; qua đó trả lại thông tin cho người dùng thông qua giao diện Web HTML5. Đồng thời cho phép người dùng gửi thông tin điều khiển các thiết bị như điều hòa, máy tạo độ ẩm, cửa ra vào về Raspberry để phù hợp với nhu cầu sử dụng. Trong giới hạn nghiên cứu, luận văn chỉ tập trung nghiên cứu về:

- Tạo một giao diện web hiển thị dữ liệu bằng ngôn ngữ HTML, Javascript, Jquery cơ bản.
- Thiết lập web server (NodeJS) trên Raspberry Pi.
- Sử dụng ngôn ngữ C để viết driver cho thiết bị, sau đó build dưới dạng node modules để có thể sử dụng được trong NodeJS.
- Thông tin điều khiển nhận được từ người dùng được mô phỏng giả lập thông qua hệ thống đèn LED kết nối đến Raspberry.
- Dữ liệu trao đổi được mã hóa bằng Grain, xác thực bằng Keccak.

Hoạt động của hệ thống:

- Thu thập dữ liệu 3 kênh: nhiệt độ, độ ẩm, trạng thái cửa ra vào.
- Hiển thị dữ liệu trên nền Web HTML5.
- Thời gian cập nhật dữ liệu là mỗi 3 giây.
- Cho phép người dùng gửi thông tin điều khiển về thiết bị Rasberry.
- Kết nối mạng wifi ở chế độ online và offline dùng mạng LAN nội bộ.

3.2. Giải quyết bài toán

Luận văn ứng dụng mã hóa đầu cuối Encrypt-and-MAC với hệ mật Grain-128 và hàm băm Keccak để giải quyết bài toán. Nội dung thực nghiệm đi sâu vào hai thành phần cốt lõi: mã hóa đầu cuối với hệ mật mã dòng trong mật mã nhẹ Grain nhằm đảm bảo độ tin cậy và mã xác thực thông báo với hàm băm Keccak nhằm đảm bảo tính xác thực cho thông điệp.

3.1.1. Mã hóa đầu cuối

Mã hóa đầu cuối (End-to-end encryption – E2EE) là phương pháp mã hóa cho phép chỉ người nhận mới biết được thông tin được gửi là gì, ngay cả các nhà cung cấp dịch vụ cũng không thể truy cập vào. Mã hóa đầu cuối thường được coi là an toàn hơn, bởi vì nó làm giảm số lượng của các bên (kể cả nhà cung cấp dịch vụ) có thể có thể can thiệp hoặc phá vỡ các mã hóa để lấy được thông tin. Hiện nay có rất nhiều nhà cung cấp dịch vụ tin nhắn đã áp dụng phương pháp mã hóa đầu cuối vào những ứng dụng di động của mình nhằm đảm bảo thông tin trao đổi của nhà sử dụng, đó là What'sApp, Facebook Messenger, Google Allo, Viber, ...

Hầu hết các giao thức mã hóa đầu cuối đều cần có thêm cơ chế xác thực để ngăn chặn các cuộc tấn công MITM (Man in The Middle), đồng thời đảm bảo cho dữ liệu không bị giả mạo bởi tin tặc, đảm bảo tính toàn vẹn của dữ liệu. Ví dụ, người ta có thể dựa vào các cơ quan chứng nhận hay một mạng lưới có thể tin tưởng được, hoặc tạo ra các bảng mã mật mã (dấu vân tay) dựa trên khóa công khai của người dùng hoặc các khoá bí mật dùng chung cho mục đích xác thực. Một phương pháp khác là sử dụng kết hợp mã xác thực thông báo cùng với mã hóa đầu cuối với mục đích đảm bảo an toàn cho dữ liệu và xác thực chủ thể của dữ liệu. Luận văn cũng nghiên cứu sự kết hợp này với thuật toán sử dụng trong mã hóa đầu cuối là mã dòng nhẹ Grain và kỹ thuật của mã xác thực thông báo là dùng HMAC với hàm băm nhẹ đang được đánh giá cao hiện nay Keccak.

3.1.2. Mã xác thực thông báo

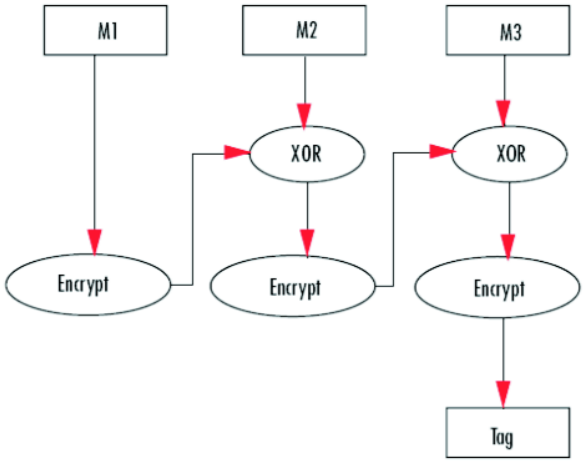
3.1.2.1. Khái niệm và mục đích sử dụng

Mã xác thực thông báo là một đoạn mã cho phép xác định nguồn gốc của dữ liệu, thuyết phục với người dùng là dữ liệu này chưa bị sửa đổi hoặc giả mạo. Đây là một cơ chế quan trọng để duy trì tính toàn vẹn và không thể từ chối dữ liệu.

MAC giúp các bên giao dịch có thể giao dịch với nhau đồng thời có thể phát hiện được thay đổi của thông báo trong quá trình vận chuyển. Chỉ cần các bên thỏa thuận khóa bí mật dùng chung. Đầu ra của MAC chính là thẻ MAC (MAC tag) được tính toán dựa trên thông báo đầu vào và khóa bí mật. Thông báo và thẻ MAC được gửi tới người nhận, người nhận tính giá trị MAC' từ thông báo nhận được và so sánh hai giá trị MAC và MAC' với nhau. Nếu hai giá trị thẻ MAC này giống nhau thì thông báo là chính xác, ngược lại thông báo đã bị thay đổi, cần phải có những thao tác cảnh báo, hay ngừng kết nối cho phù hợp.

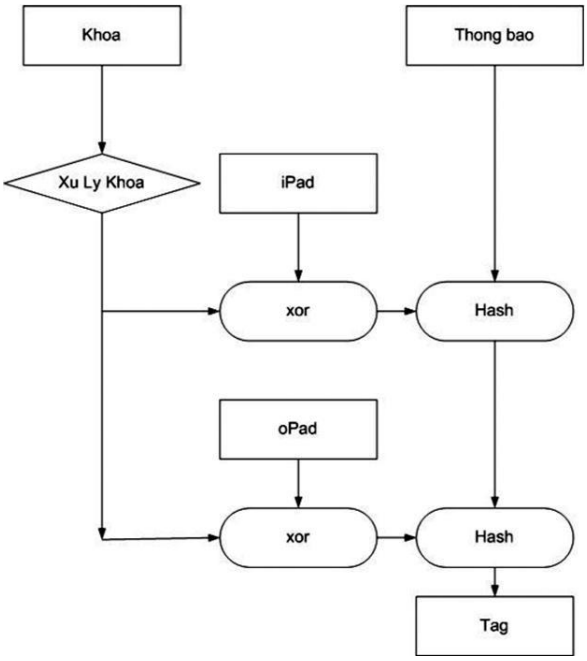
3.1.2.2. Phân loại

Viện tiêu chuẩn và công nghệ quốc gia của Mỹ (NIST) đã đề ra hai chuẩn về hàm MAC nhằm giúp các nhà phát triển thuận tiện khi ứng dụng MAC vào các sản phẩm khác nhau.



Hình 3-1: Sơ đồ CBC-MAC [32]

Mã xác thực thông báo mã hóa (Cipher Message Authentication Code - CMAC). CMAC dựa trên mã khối để thực hiện chức năng xác thực. CMAC rất phù hợp với các ứng dụng bộ nhớ hạn chế chỉ đủ để dùng cho mã hóa dữ liệu. Nguyên thủy của CMAC là CBC-MAC (Hình 3.1). Tuy nhiên thiết kế của CMAC khó có thể thực hiện song song hóa nên hiệu năng của CMAC chủ yếu phụ thuộc vào sự tối ưu hóa của thuật toán sử dụng.



Hình 3-2: Sơ đồ HMAC [32]

Mã xác thực thông báo sử dụng hàm một chiều (Key-Hash Messasge Authentication Code – HMAC). HMAC bản chất là một hàm một chiều kháng va chạm hash. HMAC ban đầu được đề nghị sử dụng với SHA-1. Nhưng sau những lỗ hổng bảo mật của SHA-1, HMAC đã có những chuyển biến với việc áp dụng các hàm hash khác, ví dụ như Keccak sẽ được trình bày trong phần tiếp theo.

3.1.2.3. Thứ tự thực hiện mã hóa và MACs

Mặc dù thứ tự của mã hóa và MAC không ảnh hưởng đến độ an toàn, nhưng cũng đem lại những sự khác biệt cơ bản về hiệu quả. Có nhiều cách để kết hợp mã hóa và MAC như: Mã hóa trước – MAC sau, MAC trước – Mã hóa sau hay MAC và Mã hóa đồng thời. Dù là thực hiện theo thứ tự nào thì quan trọng nhất vẫn là khóa mã và khóa MAC không có liên hệ với nhau. Để thực hiện điều này người thiết kế cần sử dụng hai hàm tạo khóa tách biệt dành cho khóa mã hóa và khóa MAC. Luận văn cũng nghiên cứu sử dụng MAC với hàm băm Keccak trong quá trình xác thực thông điệp.

3.1.3. Hàm băm Keccak

3.1.3.1. Hàm băm

Hàm băm là một hàm ánh xạ dữ liệu kích thước tùy ý thành dữ liệu có kích thước cố định. Các giá trị được trả về bởi một hàm băm được gọi là giá trị băm, mã băm, tiêu chí, hoặc đơn giản là băm. Ban đầu hàm băm ra đời nhằm phát hiện những sai sót phát sinh khi truyền/nhận dữ liệu do lỗi của thiết bị hay của đường truyền. Càng phát triển, hàm băm càng mang nhiều chức năng hơn nữa, nhất là đối với hàm băm mật mã (cryptography hash function). Những hàm băm này phát triển từ một hệ mật nào đó. Ngoài các chức năng của hàm băm thông thường, hàm băm mật mã còn thỏa mãn các yêu cầu sau:

- Là hàm một chiều: có thể tính toán giá trị băm h từ thông điệp M nhưng không thể tìm ra một thông điệp từ giá trị băm của nó, ngoài việc thử tất cả các thông điệp có thể.
- Khả năng kháng xung đột loại một: Khi biết trước thông báo M1, không thể tìm được thông báo M2 có cùng giá trị băm với M1.
- Khả năng kháng xung đột loại hai: Không thể tìm thấy hai thông điệp M1 và M2 khác nhau có cùng giá trị băm.

Nhờ những đặc tính vượt trội của mình hàm băm mật mã được ứng dụng khá rộng rãi, ví dụ như trong chữ ký số, mã xác thực thông báo, hỗ trợ kiểm tra mật khẩu, xác thực thông điệp trong các kênh truyền tin, các giao thức kết nối an toàn trên web, … Ngoài ra hàm băm còn có mặt trong các kiến trúc an ninh của hệ điều hành máy tính. Có thể nói, hàm băm có mặt ở bất kỳ nơi nào có nhu cầu bảo vệ thông tin dù là máy tính hay mạng giao tiếp [5].

3.1.3.2. Hàm băm nhẹ Keccak

Lịch sử

Năm 2007, NIST tổ chức cuộc thi chọn thuật toán cho chuẩn hàm băm của quốc tế, sau những tấn công thành công lên hàm băm SHA-1 và SHA-2. Đến năm 2012, nhóm các nhà mật mã người Bỉ đứng đầu là Daemen đã dành chiến thắng với hàm băm Keccak. Keccak được chọn làm thuật toán cho chuẩn hàm băm SHA-3.

Thuật toán

Chức năng cơ bản của Keccak là một hoán vị được chọn từ bộ bảy hoán vị $Keccak - f$, ký hiệu bởi $Keccak - f[b]$, trong đó $b \in \{25, 50, 100, 200, 400, 800, 1600\}$ là miền của phép hoán vị. Miền của phép hoán vị cũng là miền của trạng thái trong việc xây dựng sponge. Trạng thái được tổ chức thành một mảng 5x5 với chiều dài w bits, trong đó $w \in \{1, 2, 4, 8, 16, 32, 64\}, (b = 25w)$. Giả mã của $Keccak[r, c, d]$ được đưa ra trong thuật toán sau:

$Keccak[r,c,d](M)\{$

Initialization and padding:

$S[x,y] = 0, \qquad \forall (x,y) \text{ in } (0...4, 0...4)$

$P = M \| 0x01 \| byte(d) \| byte(r/8) \| 0x01 \| 0x00 \| ... \| 0x00$

Absorbing phase:

$\forall$ block P_i in P

$S[x,y] = S[x,y] \oplus P[x+5y], \qquad \forall (x,y) \text{ such that } (x+5y) < (r/w)$

$S = Keccak - f[r+c](S)$

Squeezing phase:

$Z = empty\ string$

while output is requested

$Z = Z \| S[x,y], \qquad \forall (x,y) \text{ such that } (x+5y) < (r/w)$

$S = Keccak - f[r+c](S)$

return Z

$\}$

Việc triển khai nhanh chóng và song song của Keccak được trình bày trong [19, 20]. Kích thước của module phụ thuộc vào sự lựa chọn độ rộng b của hoán vị Keccak-f[b]. Trong đề xuất của SHA-3, độ rộng này được chọn là 1600. Trong thực hiện song song hoàn toàn, điều này tương ứng với số cổng tối thiểu là 1600 flip-flops, 1600 inverters, 1600 AND và 4864 cổng XOR. Bảng 3-1 liệt kê số cổng tương ứng cho việc triển khai hoàn toàn song song của Keccakf [1600] và một vài ứng viên SHA-3 khác.

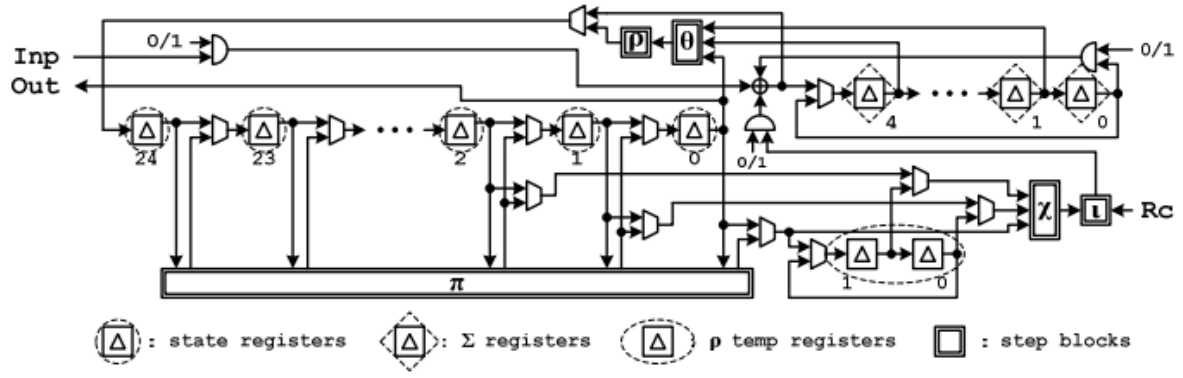
Bảng 3-1: So sánh Keccak với một vài ứng viên của SHA-3

Thuật toán	Vùng (KGE)
BLAKE-32	45.64
CubeHash16/32-h	58.87
Fugue-256	46.25
Grostl-256	58.40
Hamsi-256	58.66
JH-256	58.83
Keccak-256	**56.32**
Lufa-224/256	44.97
Shabal-256	54.19
SHAvite-3	57.39
Skein-256-256	58.61

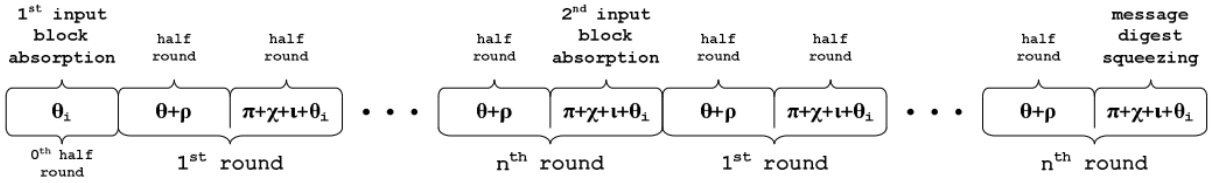
Như được thấy từ bảng, số cổng cho một thực hiện song song đầy đủ của Keccak vượt quá con số chấp nhận được cho một hàm băm nhẹ.

Kiến trúc Keccak tuần tự

Kiến trúc Keccak sử dụng lợi thế của quá trình xử lý hàng loạt. Dữ liệu được xử lý theo làn (1/25 của toàn bộ trang). Mỗi vòng lặp đăng ký số 24-0 để lưu trữ các trạng thái nội bộ. Bốn đăng ký tổng (bên phải với số đã đăng ký là 4-0) lưu trữ tổng của hàng. Các khối hoạt động để thực hiện từng bước của một vòng Keccak là θ, π, χ và ι module. Tất cả các module này (ngoại trừ π) hoạt động trên một làn đơn giản, giảm số cổng kết hợp một cách hiệu quả. Module π được thực hiện song song trên tất cả 25 làn.

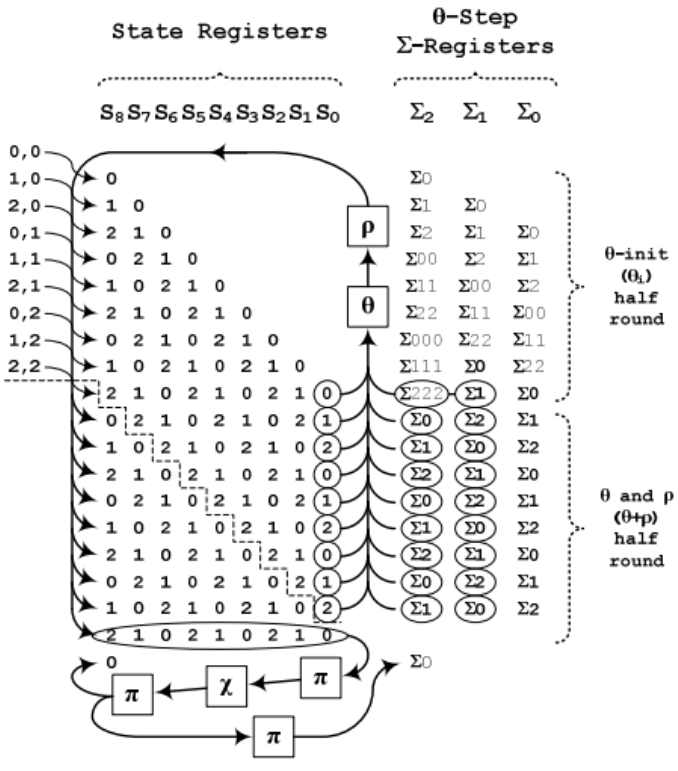


Hình 3-3: Kiến trúc Keccak nối tiếp



Hình 3-4: Các vòng xử lý dữ liệu Keccak tuần tự

Toàn bộ quá trình xử lý dữ liệu trong mỗi nửa vòng được giải thích bằng Hình 3-4 một phiên bản tinh chỉnh của Keccak, ở đây có 3 làn. Trong quá trình thực hiện, tác giả [11] áp dụng cùng thời gian với cấu trúc đường 5x5 thực tế.



Đánh giá Keccak [31]

Trong quá trình phân tích Keccak, các nhà phân tích đã cho rằng Keccak bao gồm nhiều cấu trúc nổi trội. Đầu tiên là cấu trúc Sponge. Khác với các hàm băm khác, Keccak không sử dụng hàm nén cho các khối đầu vào mà sử dụng một PRP (pseudorandom

permutation) không khóa – một hoán vị không thể đoán trước được. Ngoài ra, các nhà thiết kế còn biến Keccak thành mã xác thực thông báo khi thêm khóa bí mật cho đầu vào thông báo. Một cách phát triển khác khi đưa khóa bí mật vào vector khởi tạo và thực hiện chế độ gamma với độ dài tùy ý, đã biến cấu trúc Keccak trở thành mã dòng. Bởi vậy, Keccak có nhiều ưu điểm vượt trội so với các hàm băm khác:

- Keccak có số vòng lặp là 18 vòng và kích thước trạng thái thay đổi từ 25, 50, 100, 200, 400, 800 đến 1600.
- Keccak có khả năng thực hiện trên cả 2 nền tảng 32 bits và 64 bits.

Trong [11], tác giả đã thực hiện cả hai việc triển khai song song và triển khai hàng loạt (sử dụng kiến trúc được đề xuất) của Keccak cho chiều rộng là 8 bits (l = 3), 16 bits (l = 4) và 64 bits (l = 6) Đến Keccak-f [200], Keccak-f [400] và Keccak-f 1600, tương ứng trên một công nghệ CMOS kỹ thuật số 0.13µm tiêu chuẩn.

Các cổng tương ứng tính, giá trị thông lượng và điện năng tiêu hao được liệt kê trong. Ngoài ra, chúng tôi so sánh các ứng cử viên nhẹ của chúng tôi với MAME, một hàm băm được thiết kế đặc biệt cho các ứng dụng trọng lượng nhẹ và một thực hiện SHA-1.

	Hash output size	Data path size	Input data size	Cycles per block	T/put at 100 KHz (Kbps)	Area (KGE)	Efficiency (bps/GE)	Power cons. (µW/MHz)
Parallel *Keccak-f*[1600]	256	64	1088	24	4533	47.63	95.40	315.1
Parallel *Keccak-f*[400]	128	16	144	20	720	10.56	68.18	78.1
Parallel *Keccak-f*[200]	64	8	72	18	400	4.9	81.63	27.6
Serial *Keccak-f*[1600]	**256**	**64**	**1088**	**1200**	**90.66**	**20.79**	**4.36**	**44.9**
Serial *Keccak-f*[800] (estimate)	**128**	**32**	**544**	**1100**	**49.45**	**13.00**	**3.80**	**28.2**
Serial *Keccak-f*[400]	**128**	**16**	**144**	**1000**	**14.4**	**5.09**	**2.83**	**11.5**
Serial *Keccak-f*[200]	**64**	**8**	**72**	**900**	**8**	**2.52**	**3.17**	**5.6**

Hình 3-5: So sánh hiệu suất của Keccak triển khai song song và nối tiếp

	Hash output size	T/put at 100 KHz (Kbps)	Area (KGE)	Efficiency (bps/GE)
SHA-1 [5]	160	148.8	5.53	26.91
MAME [7]	256	146.7	8.1	18.10
Serialized *Keccak-f*[400]	**128**	**14.4**	**5.09**	**2.83**
Serialized *Keccak-f*[200]	**64**	**8**	**2.52**	**3.17**

Hình 3-6: So sánh hiệu suất giữa Keccak, MAME và SHA-1

3.1.4. Tạo và trao đổi khóa

Trong một hệ thống E2EE, các khóa mã hóa chỉ được biết đến với các bên giao tiếp. Để đạt được mục tiêu này, các hệ thống E2EE có thể mã hóa dữ liệu bằng cách sử dụng chuỗi ký hiệu được sắp xếp từ trước gọi là bí mật chia sẻ trước (Pre-shared secret - PGP) hoặc khóa dùng một lần xuất phát từ bí mật chia sẻ trước đó (DUKPT). Hoặc

một cách khác là thương lượng một khóa bí mật tại chỗ bằng cách sử dụng trao đổi khóa Diffie-Hellman (OTR – Off-The-Record Message).

Trao đổi khóa Diffie–Hellman (D-H) là một phương pháp trao đổi khóa mật mã an toàn trên một kênh công cộng và là một trong những giao thức khoá công khai đầu tiên được Ralph Merkle đề xuất và đặt tên theo Whitfield Diffie và Martin Hellman. Phương pháp trao đổi khóa này cho phép bên gửi và bên nhận cùng thiết lập một khóa bí mật trên một kênh không an toàn. Khóa này có thể được sử dụng cho một hệ mật mã khóa đối xứng tiếp theo.

Mô tả giao thức

- Thiết lập khóa
    - Alice và Bob sử dụng chung một module p và một phần tử sinh g của nhóm cyclic hữu hạn G (p và g là nguyên tố cùng nhau).
    - Alice chọn một số bí mật a và gửi $A = g^a \bmod p$ cho Bob.
    - Bob chọn một số bí mật b và gửi $B = g^b \bmod p$ cho Alice.
    - Alice tính $s = (B)^a \bmod p$.
    - Bob tính $s = (A)^b \bmod p$

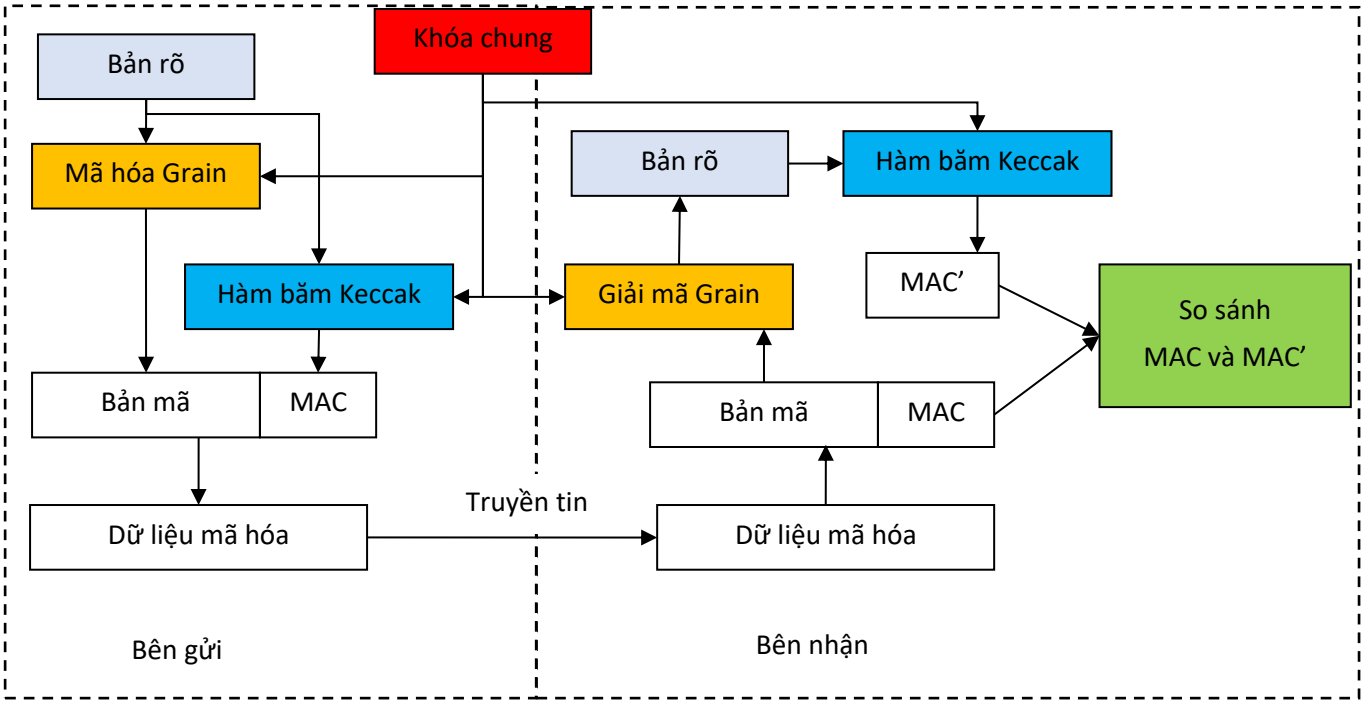
Cả Alice và Bob đều có cùng giá trị s bởi theo module p:

$A^b \bmod p = g^{ab} \bmod p = g^{ba} \bmod p = B^a \bmod p$.

Áp dụng cho thiết bị Raspberry, để hệ thống đạt được sự an toàn tối đa, cần sinh 2 khóa cho 2 quá trình chính mã hóa K_e và xác thực thông điệp K_m ngay từ khi client kết nối đến server. Các khóa này cần được giữa bí mật trong suốt quá trình truyền tin và chỉ được sử dụng trong một phiên kết nối của client và server. Sau khi mất kết nối giữa client và server, khóa chung này sẽ mất hiệu lực. Ứng dụng sử dụng Trao đổi khóa Diffie–Hellman để thực hiện tạo và trao đổi khóa chung trong quá trình mã hóa đầu cuối.

3.1.5. Mô hình mã hóa và xác thực

Ứng dụng sử dụng Grain-128 để thực hiện mã hóa tín hiệu/ thông điệp đồng thời sử dụng hàm băm Keccak để xác thực. Bên phía nhận, người nhận sẽ sử dụng khóa chung đã được thống nhất từ trước để giải mã đồng thời so sánh với kết quả trả về từ hàm băm Keccak cho bản rõ sau khi giải mã để kiểm tra tính toàn vẹn dữ liệu. Hình 3-7 mô tả mô hình mã hóa và xác thực được sử dụng trong luận văn.



Hình 3-7: Mô hình mã hóa và xác thực

Mã hóa xác thực

Đầu vào của quá trình xác thực mã hóa bao gồm 4 thành phần:

- Khóa bí mật K_e dùng trong mã hóa dữ liệu.
- Khóa bí mật K_m dùng trong xác thực thông điệp.
- Dữ liệu cần mã hóa P: Trong nội dung của báo cáo thì có thể hiểu đây là dữ liệu truyền từ thiết bị Raspberry đến người dùng.
- Đầu ra duy nhất: Bản mã C và được trao đổi thông qua đường tuyền tin.

Quá trình mã hõa xác thực diễn ra như sau:

1. Thông điệp sẽ được mã hóa trước tiên dựa trên mã dòng Grain với K_e tạo ra bản mã C.
2. Tính toán MAC của bản rõ bằng hàm băm Keccak (HMAC) với khóa K_m.
3. Gửi cả C và M trên kênh truyền tin

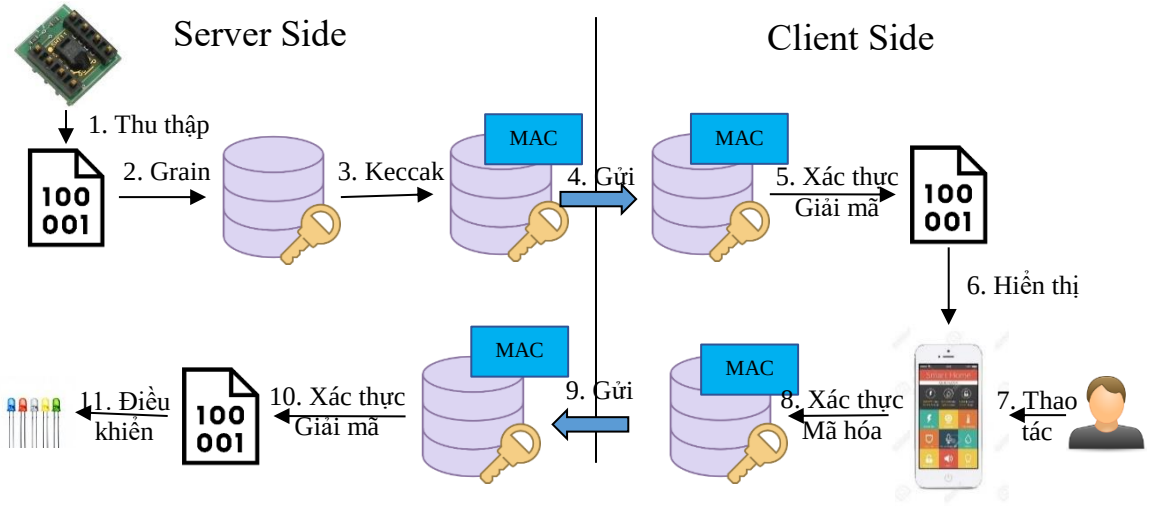
Giải mã xác thực

Đầu vào của mã hóa xác thực gồm 4 phần được định nghĩa như trên: K_e, K_m, M và C và đầu ra duy nhất hoặc là bản rõ (ở đây hiểu là thông tin thu thập được từ thiết bị Raspberry) hoặc là FAIL trong trường hợp một trong số các đầu vào không tin cậy.

Quá trình giải mã xác thực diễn ra như sau:

1. Tách C và M.
2. Giải mã bản rõ P từ bản mã M.
3. Tính toán lại giá trị M' là MAC của bản rõ nhận được.

4. So sánh giá trị 2 giá trị MAC để xem thông điệp có toàn vẹn không. Nếu toàn vẹn, người dùng có thể chấp nhận thông điệp và ngược lại, người dùng có thể bỏ qua thông điệp.



Hình 3-8: Quá trình thực hiện

3.3. Môi trường và dữ liệu thực nghiệm

3.3.1. Môi trường lập trình

- Môi trường lập trình
  - Chip: Intel Core i5 CPU 2.40 GHz
  - Ram: 8.00 GB
  - Hệ điều hành: Microsoft Windows 7 64 bits
- Công cụ lập trình: Visual Studio 2012
- Ngôn ngữ:
  - Server: nodejs, C
  - Client: HTML5, javascript, JQuery

3.3.2. Môi trường thực nghiệm

- Raspberry Model B
  - RAM: 512 MB
  - SoC: 700 MHz
  - Hệ điều hành: Raspbian cài trên thẻ MicroSD 1GB
- SHT11 (Xem phụ lục B) – Cảm biến nhiệt độ và độ ẩm.
- Công tắc từ giả lập cửa ra vào Door Sensor
- Hệ thống đèn LED

3.3.3. Thiết lập phần cứng

Cảm biến nhiệt độ và độ ẩm với SHT11

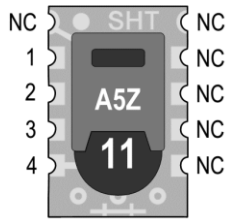
SHT11 là bộ cảm biến nhiệt độ và độ ẩm của hãng Sensirion với độ chính xác khá cao. Các cảm biến tích hợp các bộ phận cảm biến cộng với xử lý tín hiệu trên một máy in chân nhỏ và cung cấp đầu ra số đầy đủ. Một bộ cảm biến điện dung duy nhất được sử dụng để đo độ ẩm tương đối trong khi nhiệt độ được đo bằng một cảm biến khoảng cách băng. Công nghệ CMOSens áp dụng đảm bảo độ tin cậy cao và độ ổn định lâu dài. Cả hai bộ cảm biến được kết nối liền mạch với bộ chuyển đổi kỹ thuật số tương tự 14 bits và mạch nối tiếp. Điều này dẫn đến chất lượng tín hiệu vượt trội, thời gian đáp ứng nhanh và không nhạy cảm với nhiễu bên ngoài (EMC). Bảng dưới đây cho thấy các đặc điểm kỹ thuật của cảm biến SHT11.

Bảng 3-2: Đặc điểm kỹ thuật của SHT11

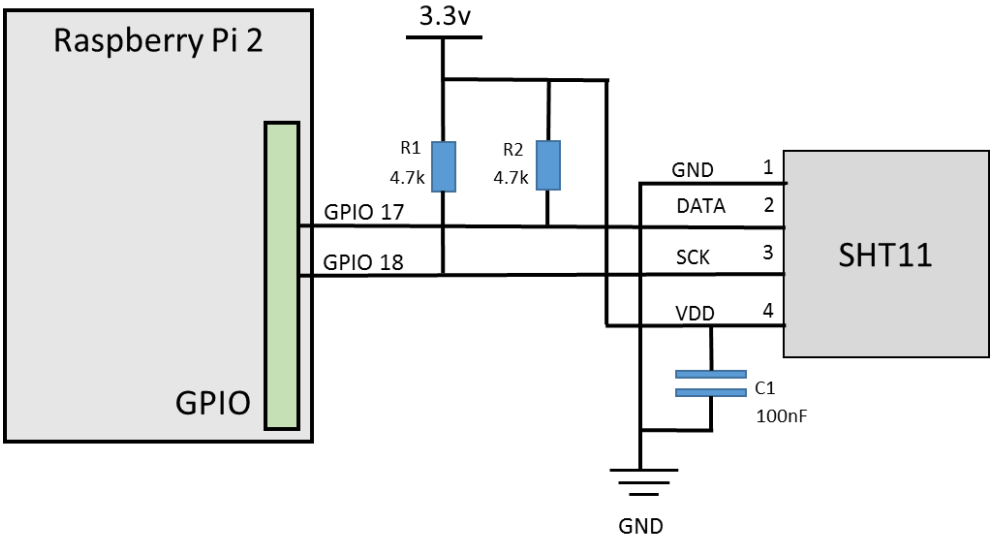
Đặc tính	Mô tả
Mẫu	SHT11
Nguồn cung cấp	2.4-5.5V DC
Tín hiệu đầu ra	Tín hiệu số với 2 dây giao điện
Phạm vi hoạt động	Độ ẩm 0-100%RH; Nhiệt độ -40~125 Celsius
Độ chính xác	Độ ẩm +-3%RH (Max +-5%RH); Nhiệt độ +-

	0.4Celsius (Max 2.5 Celsius)
Độ phân giải độ ẩm	0.4 - 0.05% của 8 – 12 bits
Độ phân giải nhiệt độ	0.04 – 0.01 Celsius của 12 – 14 bits
Kích thước	Mô hình nhỏ 14*18*5.5mm; Mô hình lớn 22*28*5mm

Pin	Name	Comment
1	GND	Ground
2	DATA	Serial Data, bidirectional
3	SCK	Serial Clock, input only
4	VDD	Source Voltage
NC	NC	Must be left unconnected



Hình 3-9: Thiết kế của SHT11



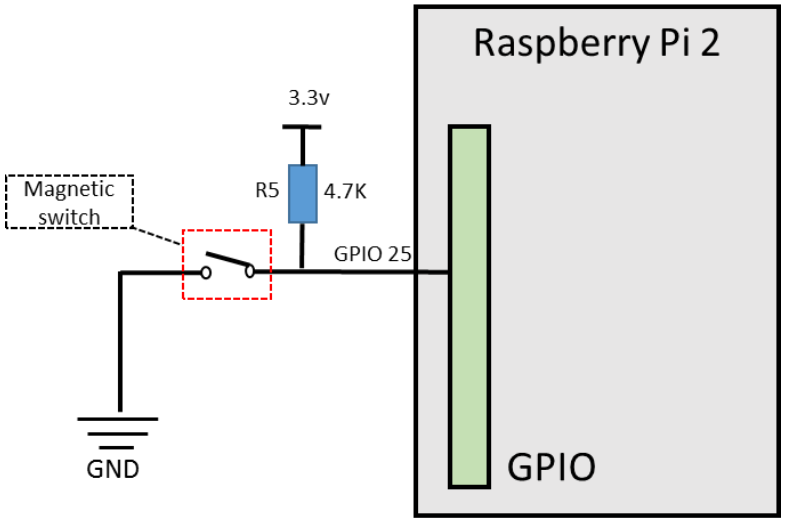
Hình 3-10: Kết nối của SHT11 và Raspberry

Như hình trên, chân dữ liệu của SHT11 được nối với chân GPIO 17 và chân SCK được nối với GPIO 18. Luận văn sử dụng hai điện trở: R1 và R2 cho điện trở kéo lên[7]. Sử dụng điện trở kéo để đảm bảo rằng tín hiệu (SCK và DATA) sẽ là mức logic hợp lý: mức logic cao hoặc thấp. Các điện trở khoảng 4700 Ω. Luận văn cũng sử dụng một tụ điện về 100nF cho bộ lọc tiếng ồn tại VDD pin.

[7] Điện trở kéo lên là một điện trở được dùng khi thiết kế các mạch điện tử lôgíc. Nó có một đầu được nối với nguồn điện áp dương (thường là Vcc hoặc Vdd) và đầu còn lại được nối với tín hiệu lối vào/ra của một mạch lôgíc chức năng.

Công tắc từ giả lập cửa ra vào

Để phát hiện mở / đóng cửa, luận văn sử dụng một chuyển đổi từ. Bộ chuyển đổi từ sẽ kết nối với Raspberry Pi 2 thông qua GPIO 25 như thể hiện trong hình bên dưới.

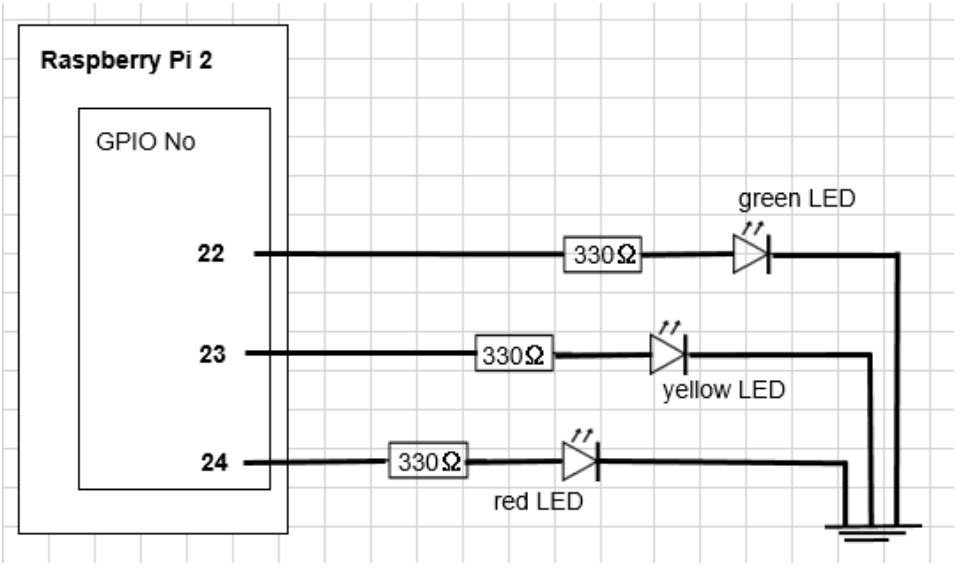


Hình 3-11: Kết nối giữa công tắc từ (magnetic switch) giả lập cửa ra vào và Raspberry

Luận văn sử dụng điện trở R_5 cho GPIO 25, nhưng điện trở này đã được cài sẵn trong Raspberry Pi 2. Vì vậy, luận văn chỉ cần cấu hình điện trở này bằng phần mềm. Khi bật OFF, GPIO 25 được nối với mặt đất. Vì vậy, điện thế ở mức thấp hay dữ liệu là 0. Khi bật ON, GPIO 25 được kết nối với V_CC. Vì vậy, điện thế ở mức cao hay dữ liệu đọc vào là 1.

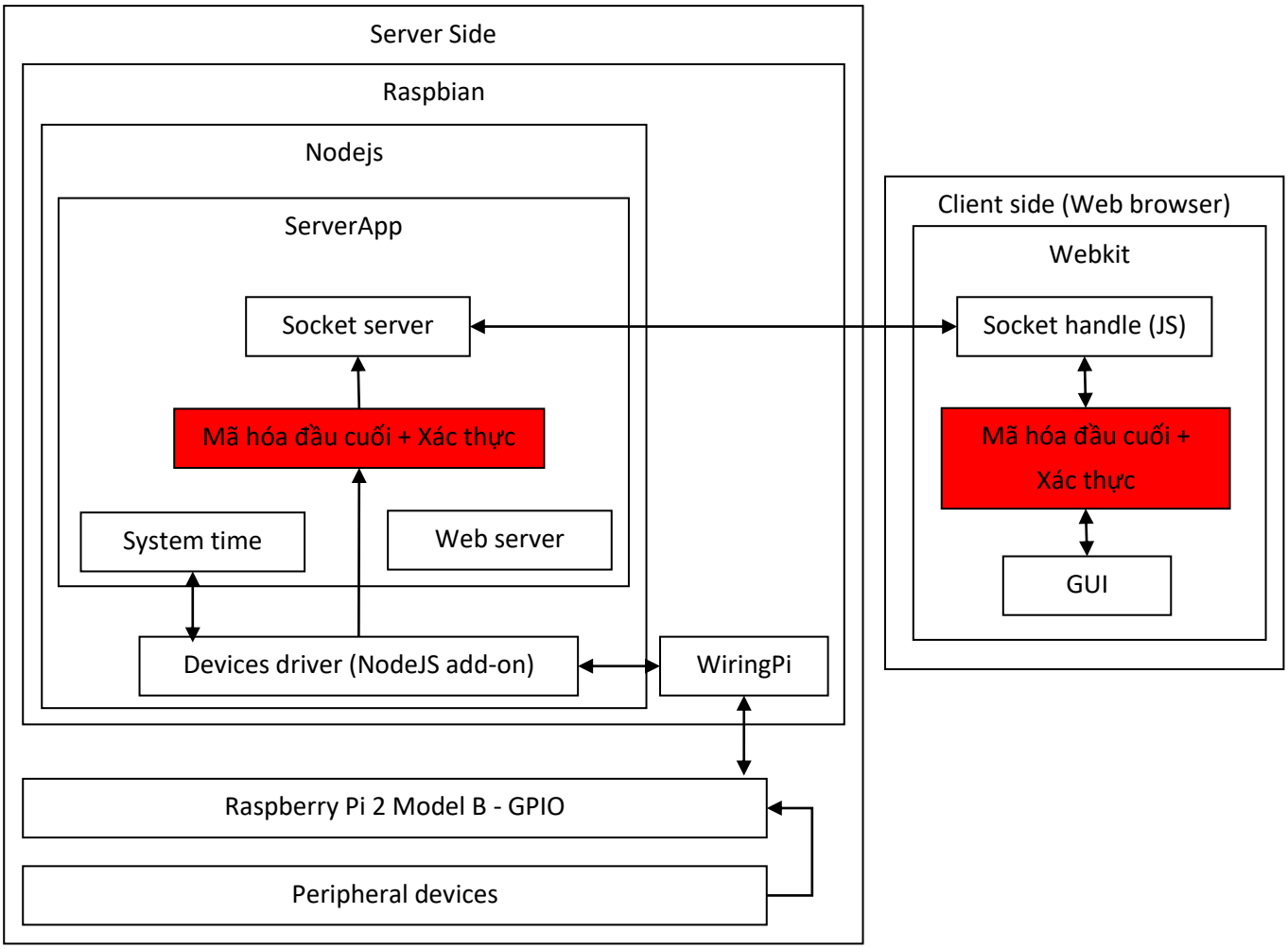
Thiết lập hệ thống đèn LED

Luận văn sử dụng đèn LED để mô phỏng các tín hiệu điều khiển từ phía Client. Luận văn sử dụng thư viện wiringPi để điều khiển các đèn LED. Hình dưới mô tả chân GPIO kết nối từ Raspberry đến các đèn LED.



Hình 3-12: Kết nối giữa Raspberry và hệ thống đèn LED

3.4. Ứng dụng mã hóa đầu cuối và mã xác thực trong giải quyết bài toán



Hình 3-13: Mô hình ứng dụng

Quá trình thực hiện bao gồm:

1. System timer định kỳ lấy dữ liệu từ cảm biến SHT11 thông qua driver.
2. Dữ liệu của cảm biến được mã hóa bằng phương pháp mã hóa dòng nhẹ Grain và thêm mã xác thực thông báo bằng Keccak.
3. Server sẽ truyền dữ liệu được mã hóa xuống cho client thông qua thư viện socket.io.
4. Client lấy thông điệp từ server thông qua thư viện socket.io.
5. Client xác thực và giải mã thông điệp trên.
6. Client hiển thị dữ liệu lên trên màn hình, nếu lỗi thông báo cho người dùng.
7. Người dùng thao tác điều khiển các thiết bị thông qua GUI.
8. Client mã hóa thông điệp bằng Grain-128 và thêm mã xác thực Keccak cho nó rồi gửi đến Server qua socket.io.
9. Server giải mã và xác thực thông điệp trên.
10. Server gửi tín hiệu điều khiển đến các devices kết nối đến nó. Trong luận văn này chỉ dừng lại ở việc mô phỏng việc gửi tín hiệu bằng các đèn LED.

- Tăng nhiệt độ điều hòa hiển thị đèn LED đỏ.
- Giảm nhiệt độ điều hòa hiển thị đèn LED xanh.
- Mở cửa hiển thị đèn LED vàng.
- Đóng cửa tắt đèn LED vàng.

3.5. Kịch bản thực nghiệm

3.5.1. Raspberry lấy dữ liệu từ các sensor và gửi đến client

Bước 1: Rasperry định kỳ lấy lần lượt các tín hiệu (nhiệt độ, độ ẩm, trạng thái cửa ra vào) từ Sensor qua các chân GPIO (3 giây / 1 lần). Các tín hiệu này được chuyển đến ServerApp qua WiringPi.

Bước 2: Server thêm các định danh, thiết lập định dạng cho các tín hiệu.

Định dạng dữ liệu: [Định danh][Chiều dài thông điệp][Thông tin]

- [Định danh]: Định danh của thông điệp (1 byte)
  - TEMP_MESS: thông tin về nhiệt độ
  - HUMI_MESS: thông tin về độ ẩm
  - DOOR_MESS: thông tin về trạng thái cửa
- [Chiều dài thông điệp]: Chiều dài của phần [Thông tin] (2 bytes)
- [Thông tin]: Nội dung thông tin về nhiệt độ, độ ẩm hay trạng thái cửa.

Sau đó áp dụng mã hóa Grain cho các tín hiệu nhận được lần lượt theo thứ tự: nhiệt độ, độ ẩm, trạng thái cửa ra vào. Kết quả là 3 chuỗi bản mã.

Bước 3: Server áp dụng hàm băm Keccak để thêm MAC cho 3 chuỗi bản mã.

Khóa sử dụng cho quá trình mã hóa và MAC đã được khởi tạo và thỏa thuận trước giữa Server và Client (Xem kịch bản trao đổi khóa mục 3.5.3).

Bước 4: Server sử dụng socket.io để gửi dữ liệu đến Client.

Bước 5: Client nhận thông tin, giải mã Grain được bản rõ.

Bước 6: Client thực hiện MAC với bản rõ => Đầu ra là bản MAC’. Client so sánh MAC và MAC’ để kiểm tra tính xác thực của thông tin nhận được. Nếu MAC = MAC’ thì hiển thị thông tin lên giao diện. Nếu ngược lại, MAC # MAC’ thì hiển thị thông báo lỗi cho người dùng và dừng kết nối.

3.5.2. Client gửi thông tin điều khiển đến Server

Bước 1: Người dùng nhấn các nút điều khiển trên giao diện:

- Tăng / giảm nhiệt độ
- Tăng / giảm độ ẩm
- Đóng / mở cửa ra vào

Bước 2: ClientApp thêm định danh và thiết lập định dạng cho thông tin điều khiển. Sau đó áp dụng mã hóa Grain cho thông điệp để được bản mã. Định dạng dữ liệu của Client cũng tương tự như bên Server, chỉ khác ở phần định danh cho dữ liệu, sử dụng 3 loại định danh sau:

- TEMP_MESS_CTRL: điều khiển nhiệt độ
- HUMI_MESS_CTRL: điều khiển về độ ẩm
- DOOR_MESS_CTRL: điều khiển về trạng thái cửa

Bước 3: Client thêm MAC cho bản mã.

Bước 4: Client gửi thông điệp đã mã hóa và thêm MAC đến Server.

Bước 5: Server thực hiện giải mã Grain thu được bản rõ. Server thực hiện MAC trên bản rõ đó thu được MAC'. Server so sánh MAC và MAC'. Nếu MAC = MAC' thì hiển thị thông tin đèn LED. Nếu ngược lại, MAC # MAC' thì hiển thị thông báo lỗi trên bảng điều khiển và dừng kết nối.

3.5.3. Tạo và trao đổi khóa giữa Server và Client

Bước 1: Khi Client kết nối đến Server, Server sử dụng hàm createDiffieHellman() và generateKeys() của thư viện crypto[8] để tạo khóa công khai.

Bước 2: Server lấy các thông tin về module p và g bằng cách sử dụng getPrime() và getGenerator().

Bước 3: Server gửi p, g và khóa công khai đến Client.

Bước 4: Client nhận được p, g, và khóa công khai của Server. Client tính toán khóa công khai và gửi đến cho Server. Đồng thời Client tính toán khóa bí mật chung của Client và Server và lưu lại session.

Bước 5: Server nhận được khóa công khai của Client. Server tính toán khóa bí mật dùng chung và lưu giữ khóa này cho 1 session Client liên kết với Server.

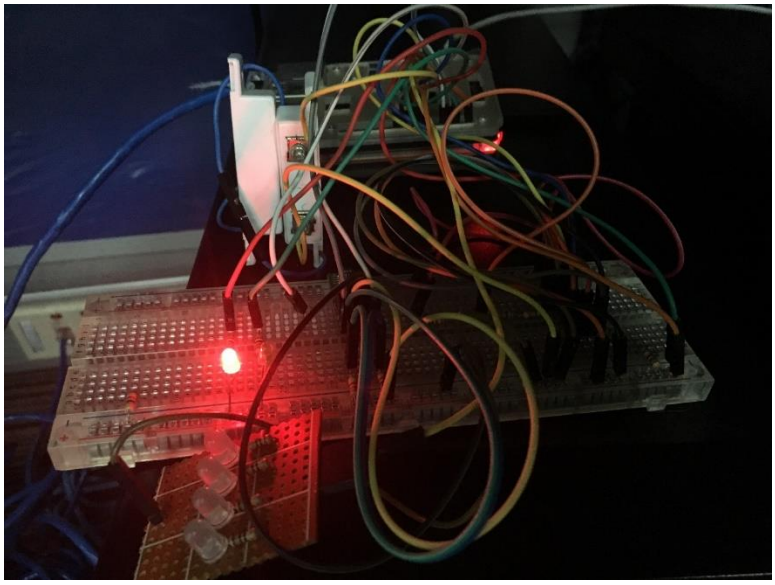
Khi Client ngắt kết nối với Server, Server tự động xóa toàn bộ những thông tin lưu trữ của Client và sẽ thiết lập khóa mới khi Client kết nối lại với Server.

Việc tạo và trao đổi khóa được thực hiện 2 lần cho khóa của hệ mật Grain và khóa của MAC với hàm băm Keccak. Cả Server và Client đều lưu trữ 2 khóa này cho các lần mã hóa và xác thực trong một phiên giao dịch.

[8] Thư viện crypto là một thư viện thông dụng của nodejs, hỗ trợ người lập trình các chức năng cơ bản khi làm về mật mã như: tạo và trao đổi khóa, hàm băm, HMAC với SHA256, tạo chữ ký,… Tài liệu về thư viện crypto có thể xem tại https://nodejs.org/api/crypto.html

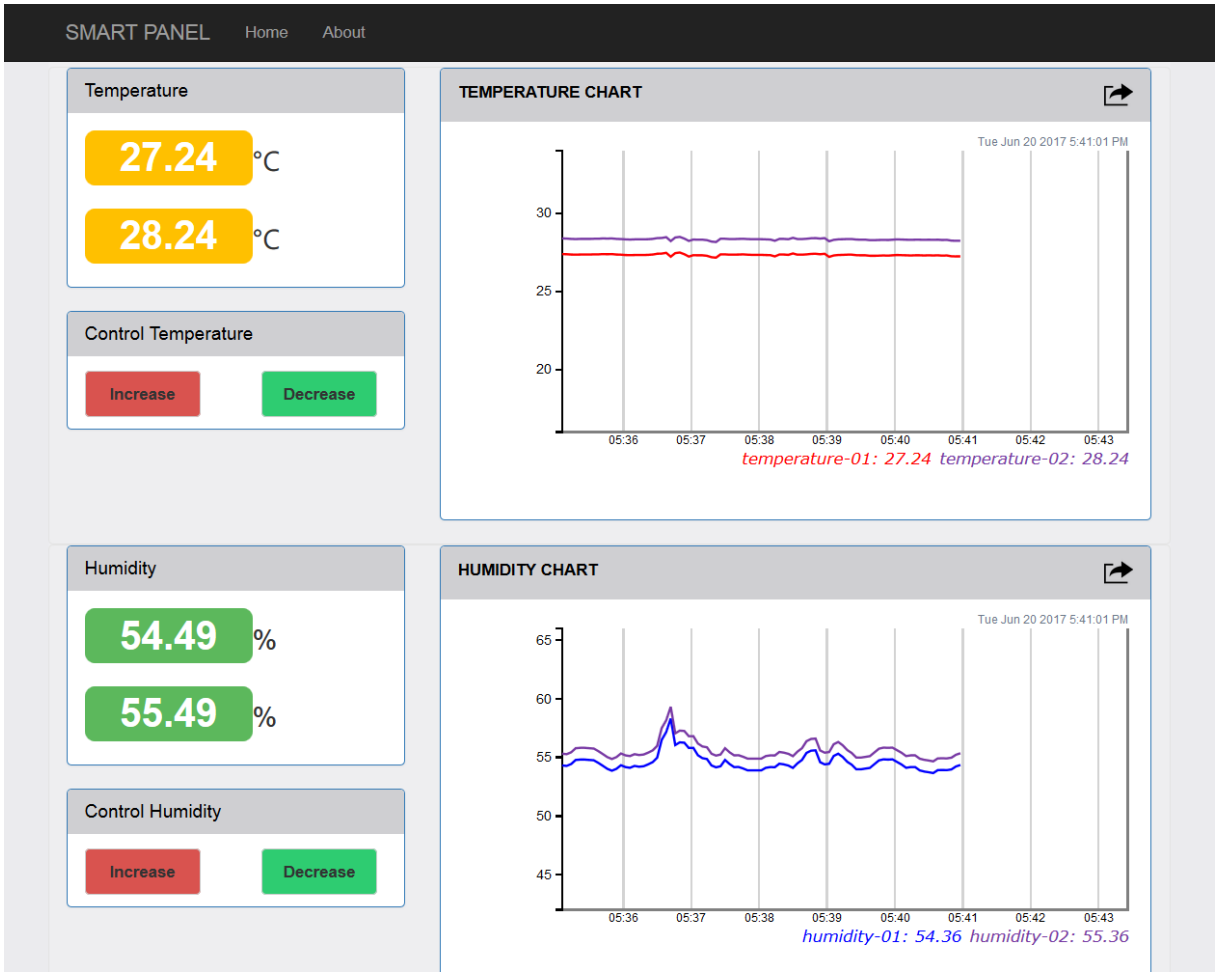
3.6. Kết quả thu được

Server Raspberry thu nhận thông tin từ các sensor (sensor đo độ ẩm, sensor đo nhiệt độ, cảm biến đóng mở cửa), truyền thông tin và hiển thị đồ thị trên client



Hình 3-14: Hình ảnh thực tế của Raspberry Pi cùng các cảm biến và đèn LED

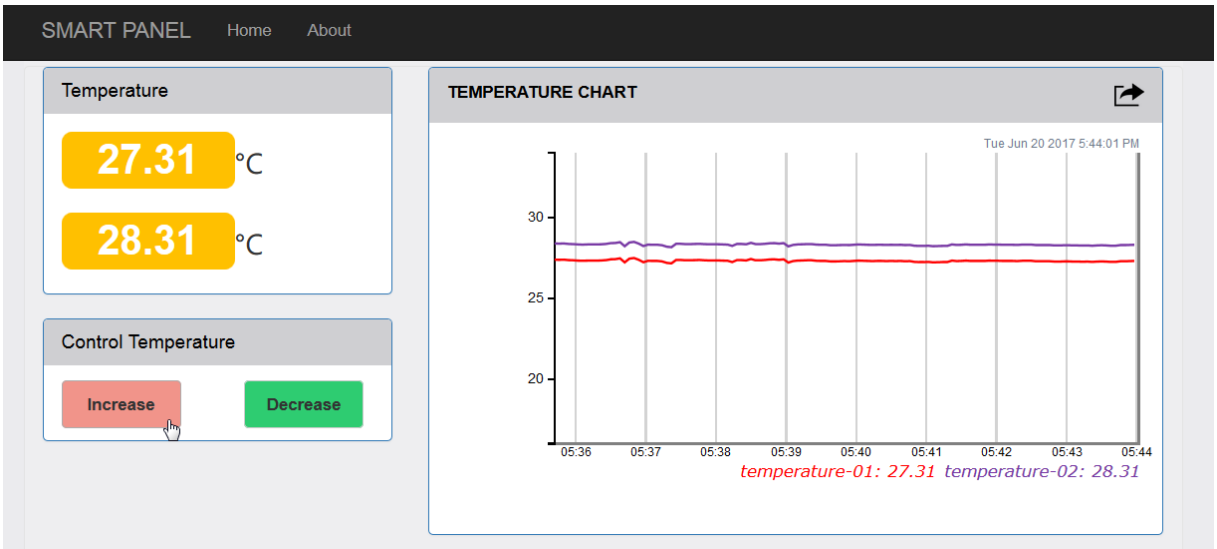
Giao diện chính của client



Hình 3-15: Giao diện chính của client

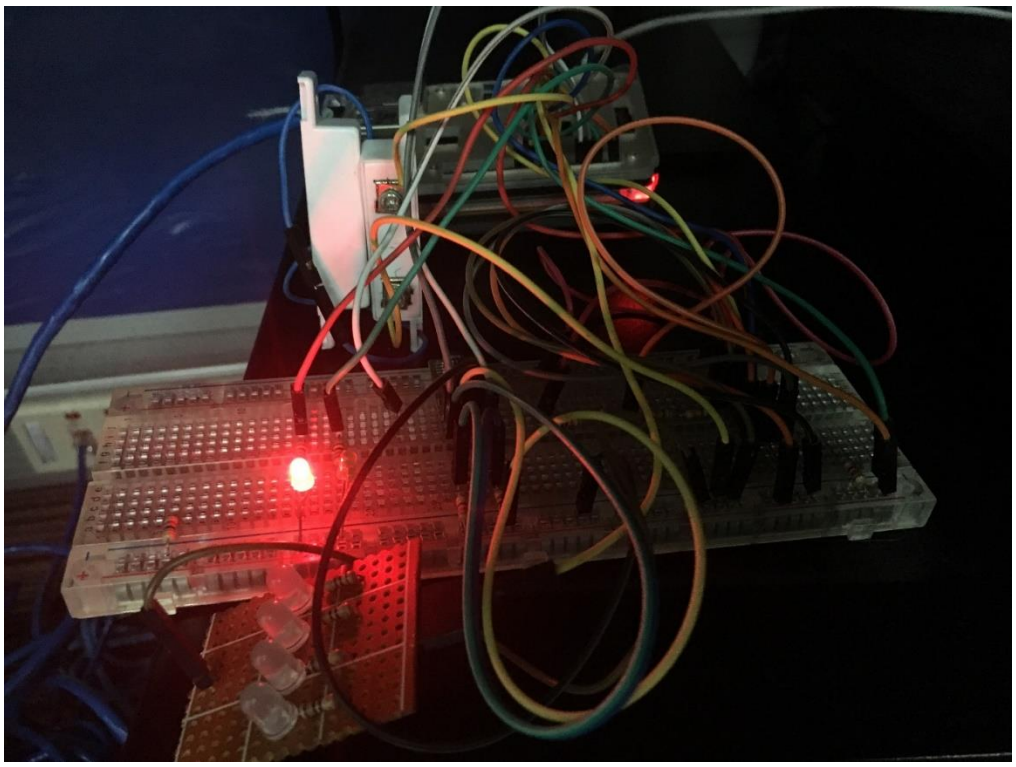
Khi nhận thấy nhiệt độ/độ ẩm thấp/cao người dùng có thể thực hiện lệnh điều khiển gửi đến các thiết bị để tăng nhiệt độ/độ ẩm. Hay khi muốn đóng/mở cửa người dùng cũng có thể gửi các lệnh điều khiển đến bộ phận điều khiển của cửa ra vào. Trong thiết bị giới hạn, luận văn chỉ mô phỏng các lệnh điều khiển này bằng cách hiển thị màu của đèn LED.

- Tăng nhiệt độ



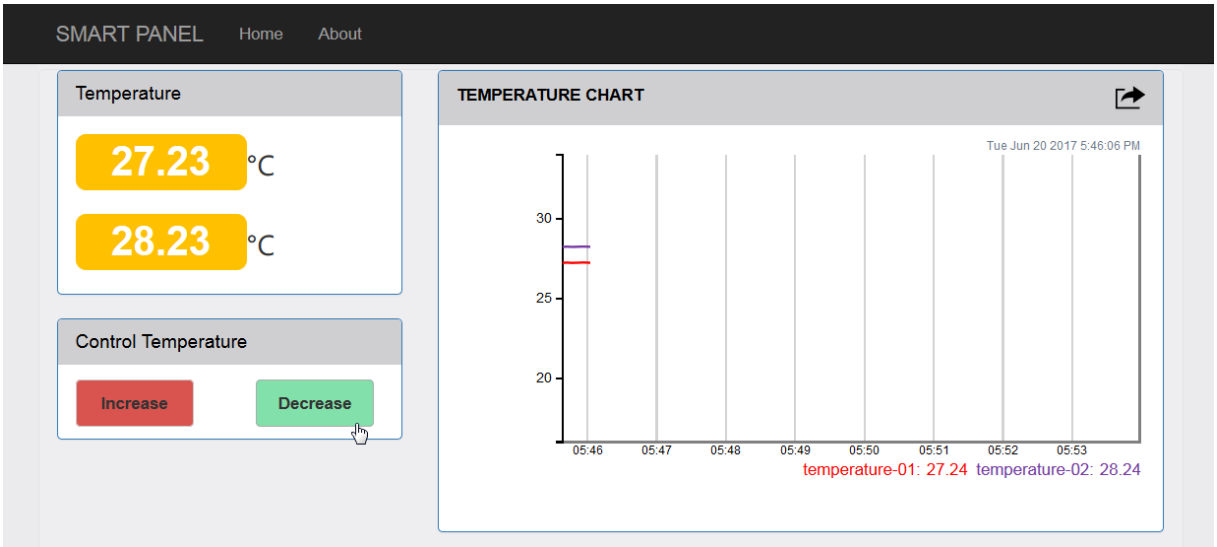
Hình 3-16: Màn hình tăng nhiệt độ

Thông tin điều khiển được hiển thị ở phía Raspberry thông qua đèn LED đỏ.



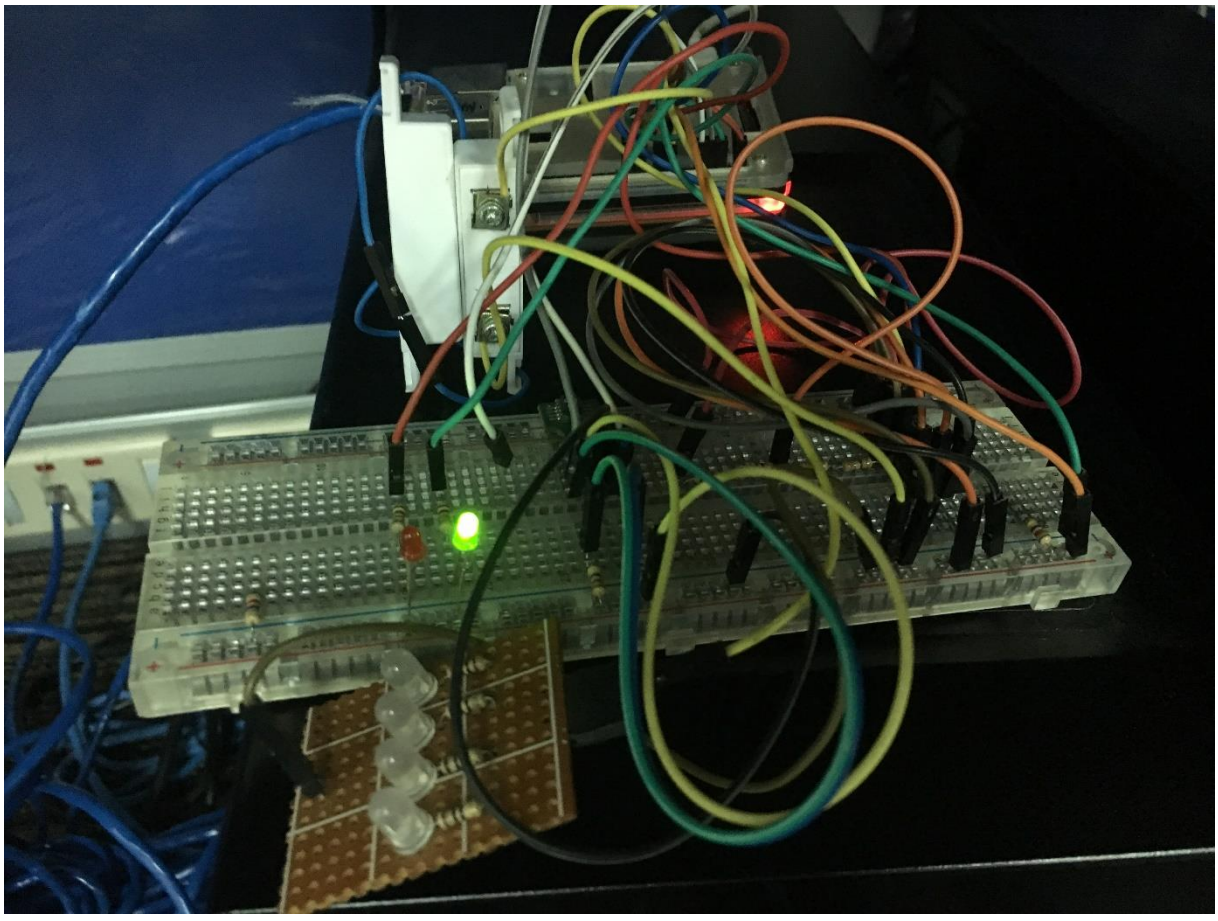
Hình 3-17: Giả lập Raspberry điều khiển tăng nhiệt độ qua đèn LED đỏ

- Giảm nhiệt độ



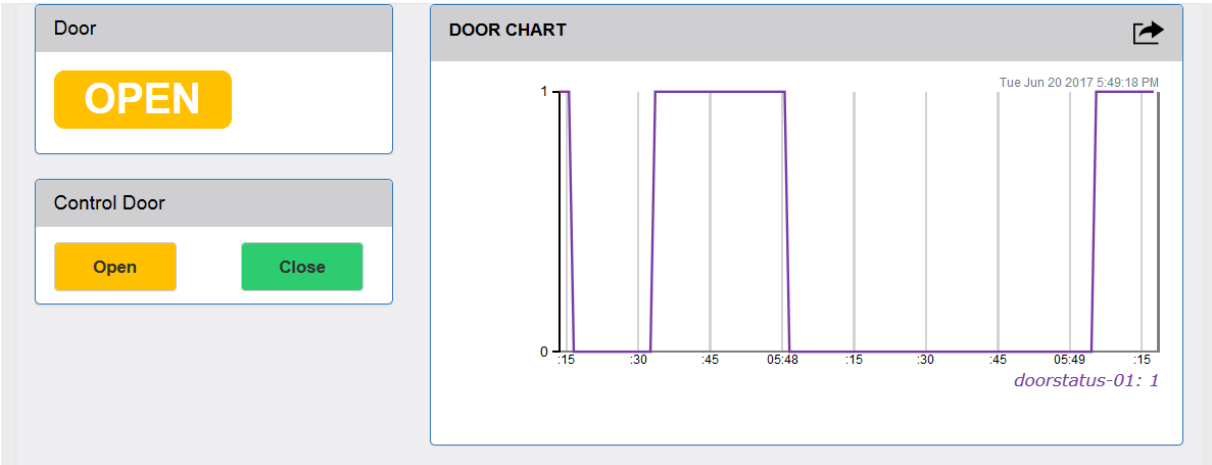
Hình 3-18: Màn hình giảm nhiệt độ

Thông tin điều khiển được hiển thị ở phía Raspberry thông qua đèn LED xanh.



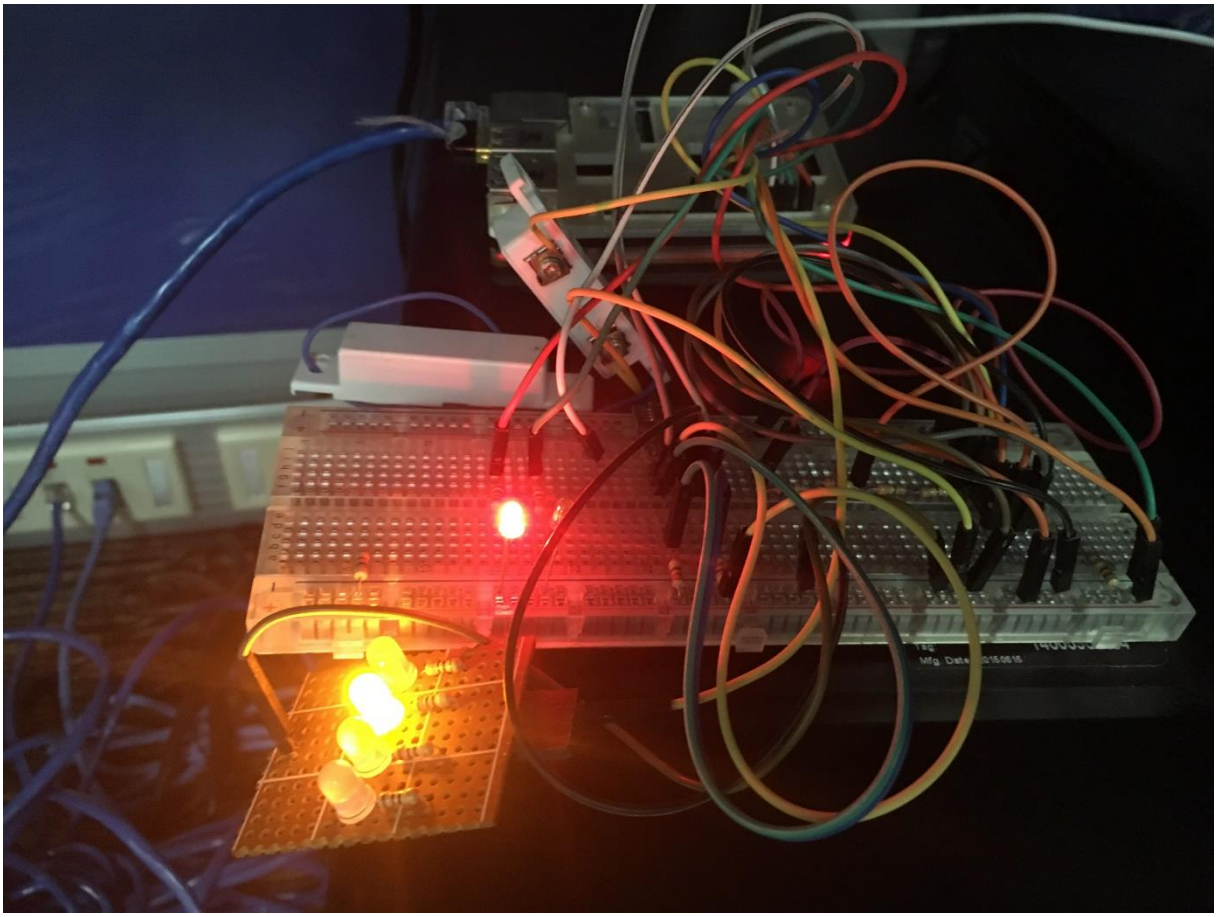
Hình 3-19: Giả lập Raspberry điều khiển giảm nhiệt độ qua đèn LED xanh

- Đóng cửa



Hình 3-20: Màn hình mở cửa

Thông tin điều khiển được hiển thị ở phía Raspberry thông qua đèn LED vàng.



Hình 3-21: Giả lập Raspberry điều khiển mở cửa qua đèn LED vàng

3.7. Đánh giá

3.7.1. Đánh giá an toàn

- An toàn: Việc thực hiện mã hóa Grain đem lại sự an toàn bảo mật cho dữ liệu trên kênh truyền. Đây cũng chính là mục đích chính của luận văn.

- Xác thực: Việc ứng dụng kỹ thuật HMAC tại các điểm cuối khiến cho tin nhắn không bị giả mạo, thay đổi trên đường truyền. Điều này cũng được coi có thể hạn chế kiểu tấn công Man-in-the-Middle attacks.
- Backdoors: Việc mã hóa sử dụng mật mã đối xứng hạn chế được việc nhà cung cấp dịch vụ có thể mở cửa hậu nhằm thu thập thông tin người dùng.

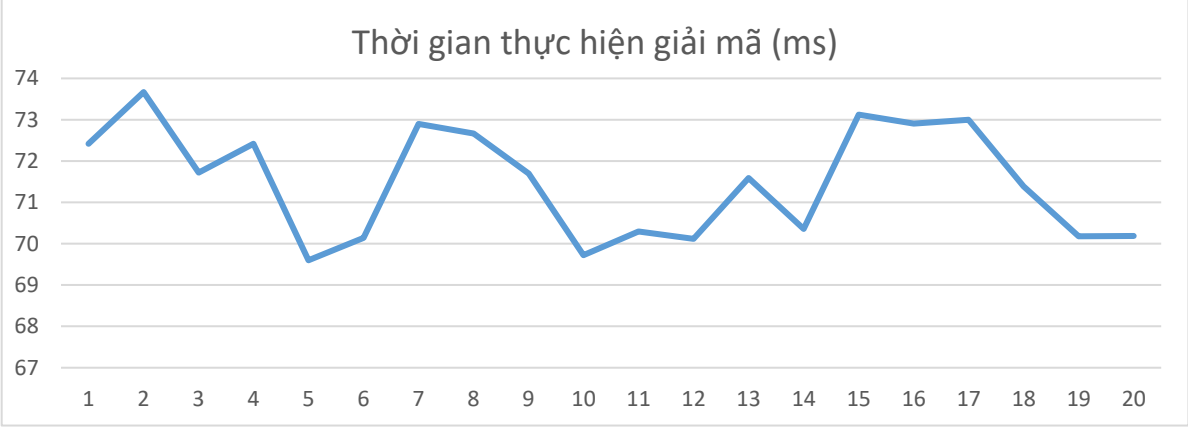
Tuy việc xây dựng kênh truyền tin an toàn trên cũng tiềm ẩn một số rủi ro:

- Do sử dụng hệ mã hóa khóa đối xứng nên phải giữ khóa bí mật chung trong suốt quá trình truyền dữ liệu.
- Mô hình chưa đủ sức để có thể ngăn chặn hoàn toàn kiểu tấn công Manin-the-Middle mà chỉ là hạn chế khả năng thực hiện nó.

3.7.2. Đánh giá hiệu năng

Tốc độ thực hiện mã hóa đầu cuối và xác thực trên thiết bị Raspberry tương đối nhanh, đảm bảo tính thời gian thực của quá trình truyền nhận dữ liệu giữa thiết bị Raspberry với các clients. Hiệu năng của luận văn được đo đạc và lấy giá trị trung bình sau 20 lần thực nghiệm.

Hình 3-22: Hiệu năng thực hiện mã hóa



Hình 3-23: Hiệu năng thực hiện giải mã

3.7.3. So sánh với các giải thuật khác ứng dụng trên Raspberry

Luận văn cũng đã áp dụng một số giải thuật mã hóa khác trên thiết bị Raspberry để thực nghiệm hiệu quả cài đặt cũng như đánh giá hiệu năng của hệ mật Grain so với một số hệ mật mã dòng và mã khối khác. Bảng dưới đây mô tả hiệu quả cài đặt, hiệu năng của các thuật toán mà luận văn đã áp dụng (với cùng một mô hình mã hóa đầu cuối và mã xác thực thông báo với hàm băm Keccak)

Bảng 3-3: So sánh Grain và một số hệ mã hóa nhẹ khác trên Raspberry

Thuật toán	**Kích thước khóa**	**Kích thước IV**	**Thông lượng 100KHz (Kb/s) ***	**Thời gian thực hiện mã hóa / giải mã (ms) ***	**Thời gian thực hiện chu trình mã hóa đầu cuối và xác thực thông báo (ms) ***
Grain v1	80	80	101.2	105.3	225.5
Grain-128	128	96	109.4	71.5	169.7
Trivium	80	80	102.2	86.9	183.5
AES	256		55.6	75.1	171.2
KATAN64	64		25.2	96.7	197.4

** Giá trị trung bình sau 20 lần thực hiện mỗi thuật toán với dữ liệu ngắn*

Có thể thấy, Grain là một hệ mật mã nhẹ có ưu điểm vượt trội về việc cài đặt cũng như sử dụng trong các thiết bị yêu cầu năng lượng nhỏ, chi phí thấp.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

1. Kết quả đạt được

1.1. Lý thuyết

Mật mã nhẹ đem lại độ an toàn phù hợp với một giải pháp cài đặt gọn nhẹ cho các thiết bị chuyên dụng, là sự cân bằng giữa độ an toàn, tính hiệu quả và giá thành. Mật mã dòng trong mật mã nhẹ được đánh giá cao về tính nhỏ gọn, dễ dàng trong cài đặt, tốc độ nhanh, độ an toàn cao hơn các giải thuật mã hóa khác. Với việc nghiên cứu tổng quan về mật mã nhẹ cùng các thuật toán đặc trưng của mật mã dòng trong mật mã nhẹ như ChaCha, E0, FCSR..., luận văn đã đưa ra những đánh giá về ưu điểm vượt trội của mật mã dòng trong mật mã nhẹ so với các giải thuật mật mã nhẹ khác. Đó chính là tiền đề cho khả năng ứng dụng của mật mã dòng trong mật mã nhẹ cho các hệ thống vạn vật kết nối IoT hiện nay.

Đi sâu nghiên cứu họ mật mã dòng Grain trong mật mã nhẹ - một trong những họ mật mã đầu tiên của mật mã dòng nhẹ, 3 phiên bản Grain V0, Grain V1 và Grain 128 cùng một phiên bản nâng cao Grain-128a cho phép triển khai nhanh hơn, với chi phí thực hiện ít hơn nhưng đem lại hiểu quả cao hơn về thông lượng, điện tích sử dụng so với một số hệ mật mã dòng và khối nhẹ khác. Đặc biệt là Grain-128a, không những đem lại hiệu quả về bảo mật mà còn hỗ trợ tính năng xác thực. Grain phù hợp với các ứng dụng phần cứng, có khả năng cung cấp bảo mật cao hơn trong khi yêu cầu phần cứng nhỏ hơn, có ưu thế hơn trong thông lượng, hiệu suất sử dụng, phù hợp với các ứng dụng sử dụng WLAN, RFID/WSN.

Ngoài mật mã dòng trong mật mã nhẹ, luận văn còn nghiên cứu về một nhánh khác trong mật mã nhẹ là mã xác thực thông báo với hàm băm Keccak – đây cũng là hàm băm đã được sử dụng trong phần thực nghiệm. Keccak phù hợp với cả triển khai nối tiếp và triển khai song song, có ưu điểm vượt trội cả về tốc độ lẫn độ an toàn so với các hàm băm trước đây được sử dụng như SHA-1, MAME, ...

1.2. Thực nghiệm

Dựa trên những nghiên cứu về lý thuyết, luận văn ứng dụng mã hóa đầu cuối với mật mã dòng Grain trong mật mã nhẹ cùng mã xác thực thông báo HMAC – Keccak cho thiết bị Raspberry trong điều khiển một vài thông số của smart home. Luận văn sử dụng thiết bị Raspberry Pi để thu thập dữ liệu từ các cảm biến SHT11 để đo nhiệt độ, độ ẩm, trạng thái cửa ra vào của ngôi nhà, phòng làm việc; qua đó trả lại thông tin cho người dùng thông qua giao diện Web HTML 5. Đồng thời cho phép người dùng gửi thông tin điều khiển các thiết bị như điều hòa, máy tạo độ ẩm, cửa ra vào về Raspberry để phù hợp với nhu cầu sử dụng dưới sự mô phỏng qua hệ thống đèn LED kết nối đến Raspberry. Dữ liệu được mã hóa bằng mật mã Grain và gắn chuỗi MAC bên trong Raspberry trước khi được gửi đi. Chỉ người dùng cuối thực sự mới có thể giải mã và xác thực được dữ liệu nhận được này. Các lệnh điều khiển từ phía người dùng cũng được thực hiện quy trình mã hóa và xác thực tương tự để đảm bảo độ an toàn của thông tin.

Kết quả thực nghiệm đã chứng minh tính đúng đắn, khả năng ứng dụng, ưu điểm vượt trội của mật mã dòng trong các hệ thống trên môi trường vạn vật kết nối. Đây cũng là tiền đề cho những nghiên cứu, ứng dụng về bảo mật trong mô hình smart home nói riêng và mô hình IoT nói chung.

2. Hướng phát triển

Trong tương lai, luận văn sẽ tiếp tục nghiên cứu ứng dụng những hệ mật mã nhẹ khác cho các thiết bị chuyên dụng của IoT để có thể đưa ra những đánh giá chính xác nhất về khả năng sử dụng cũng như ứng dụng của mật mã nhẹ trong IoT. Đồng thời nghiên cứu và phát triển hệ mã hóa đầu cuối với Grain và Keccak này vào ứng dụng smart home một cách hoàn thiện nhất để có thể đưa vào thực tế đời sống.

TÀI LIỆU THAM KHẢO

[1] Alexander Maximov, Côme Berbain, Henri Gilbert – "Cryptanalysis of Grain" (PDF). eSTREAM, 2016/01/02.

[2] Davood Rezaeipour, Reza Sabbaghi-Nadooshan, Zahra Shahosseini – "Design of New QCA LFSR and NLFSR for Grain-128 Stream Cipher" - J CIRCUIT SYST COMP 25, 1650005, 2016.

[3] Elie Bursztein – Google security blog: "Speeding up and strengthening https connections for chrome on android" – 2014/08/24. https://security.googleblog.com/2014/04/ speeding-up-and-strengthening-https.html.

[4] Ding, Jie Guan and Lin – "Related Key Chosen IV Attack on Grain-128a Stream Cipher." – Information Forensics and Security, IEEE Transactions on 8.5 page 803-809 – 2013.

[5] Tạp chí An toàn thông tin – "Chọn thuật toán trong chuẩn hàm băm năm 2012" – http://antoanthongtin.vn/Detail.aspx?CatID=b30679c6-ff8f-416f-a7b1-90921f26aea3&NewsID=0d85ae52-70e4-4939-aaad-7394a87669f4

[6] Banik, Santanu Sarkar, Subhadeep and Subhamoy Maitra – "A differential fault attack on grain-128a using MACs." – Security, Privacy, and Applied Cryptography Engineering. Springer Berlin Heidelberg, page 111-125 – 2012.

[7] Adi Shamir and Itai Dinur – Computer Science Department the Weizmann Institute Rehovot 76100, Israel – "Breaking Grain-128 with Dynamic Cube Attacks", International Association for Cryptologic Research – 2011.

[8] Arnault, F., Berger, T., Lauradoux, C., Minier, M., & Pousse, B. – "A new approach for FCSRs. In Selected Areas in Cryptography" – Springer Berlin Heidelberg, page 433-448 – 2009/01.

[9] Haina Zhang, Xiaoyun Wang – "Cryptanalysic of Stream Cipher Graim Family" – https://eprint.iacr.org – 2009.

[10] Tillich, S. – "High-Speed Hardware Implementations of BLAKE, BMW, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein" – Cryptography ePrint – 2009/11.

[11] A.H., Hasan, M.A., Namin – "Hardware Implementation of the Compression Function for Selected SHA-3 Candidates" – CACR 2009, page 28 – 2009/06.

[12] A. Maximov, M. Hell, T. Johansson and W. Meier – "The Grain Family of Stream Ciphers" – In M. Robshaw and O. Billet Editors, New Stream Cipher Designs, LNCS 4986, page 179-190 – 2008.

[13] M., Hell, & Johansson, T. – "Breaking the F-FCSR-H stream cipher in real time". In Advances in Cryptology-ASIACRYPT, Springer Berlin Heidelberg, page 557-569 – 2008.

[14] Bart Preneel, Christophe, De Cannière, Özgül Küçük – "Analysis of Grain's initialization algorithm." – Progress in Cryptology–AFRICACRYPT. Springer Berlin Heidelberg, page 276-289 – 2008.

[15] Lee, Yuseop – "Related-key chosen IV attacks on Grain-v1 and Grain-128." – Information Security and Privacy. Springer Berlin Heidelberg – 2008.

[16] A. Mollin, Richard– "An Introduction to Cryptography – 2nd ed", Taylor &Francis Group, LLC – 2007

[17] Good, T., & Benaissa, M. – "Hardware results for selected stream cipher candidates" – State of the Art of Stream Ciphers, page 191-204 – 2007.

[18] PGS.TS. Trịnh Nhật Tiến, GV. Lý Hùng Sơn – "Giáo trình an toàn dữ liệu và mã hóa", Trường Đại học Công nghệ - Đại học Quốc Gia Hà Nội – 5/2006.

[19] Joseph Lano – "CRYPTANALYSIS AND DESIGN OF SYNCHRONOUS STREAM CIPHERS", Katholieke Universiteit Leuven – FaculteitIngenieurswetenschappen Arenbergkasteel, B-3001 Heverlee (Belgium) – 2006

[20] Alexander Maximov, Berbain, Côme, Henri Gilbert– "Cryptanalysis of grain." Fast Software Encryption. Springer Berlin Heidelberg – 2006

[21] M. Hell, T. Jonasson, and W. Meier. Grain – "A Stream Cipher for Constrained Enviroments", ECRYPT Stream Cipher Project Report 2005/001 – 2005 http://www.ecrypt.eu.org/stream.

[22] Khazaei, Mehdi Hassanzadeh, Mohammad Kiaei, Shahram – "Distinguishing attack on grain." http://www. ecrypt. eu. org/stream/papersdir/071. Pdf – 2005/12/01

[23] Adi Shamir – "Stream Ciphers: Dead or Alive?", ASIACRYPT, page 22-41 – 2004.

[24] Yi Lu, http://lasecwww.epfl.ch/~vaudenay/ - "Cryptanalysis of Bluetooth Keystream Generator Two-Level E0" (PDF). Advances in Cryptology - Asiacrypt 2004, LNCS vol. 3329, page 483-499, Springer – 2004.

[25] Ari Renvall, Cunsheng Ding, Thomas W. Cusick – "Stream Ciphers and Number Theory", North-Holland Mathematical Library – 2003.

[26] A. Biryukov, Shamir, Wagner, D. – "Real time cryptanalysis of A5/1 on a PC", In Fast Software Encryption (page 1-18). Springer Berlin Heidelberg – 2001/01.

[27] Masanobu Katagi and Shiho Moriai – "Lightweight Cryptography for the Internet of Things"

[28] International standard ISO/IEC 29192 – "Information Technology - Security Techniques - Lightweight cryptography".

[29] Martin Agren, Martin Hell, Thomas Johansson, and Willi Meier – "A New Version of Grain-128 with Authentication" (http://citeseerx.ist.psu.edu)

[30] Alex Biryukow and Léo Perrin – "State of the Art in Lightweight Symmetric Cryptography" – page 8.

[31] Elif Bilge Kavun and Tolga Yalcin – "A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications" – page 260-268.

[32] KS. Vũ Văn Xứng – "Một số chú ý khi triển khai mã xác thực thông báo" – Tạp chí An toàn thông tin.

PHỤ LỤC

A. RASPBERRY Pi

A.1. Raspberry là gì?

Raspberry Pi bản chất là một vi máy tính kích cỡ như iPhone và chạy HĐH Linux. Được phát triển bởi Raspberry Pi Foundation – là tổ chức phi lợi nhuận với tiêu chí xây dựng hệ thống có thể tùy biến tùy nhu cầu người sử dụng.

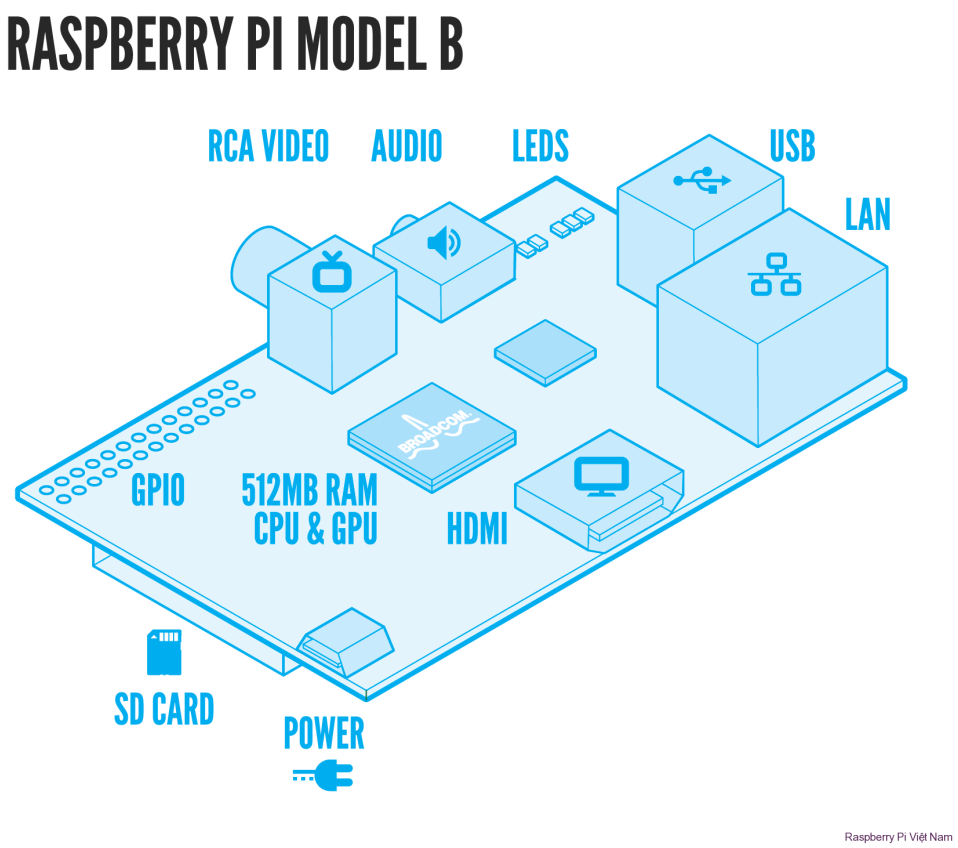
Bo mạch Raspberry bên cạnh Iphone 4

Raspberry Pi sản xuất bởi 3 OEM: Sony, Qsida, Egoman. Và được phân phối chính bởi Element14, RS Components và Egoman.

Mặc dù đối tượng hướng tới ban đầu của Raspberry Pi là những sinh viên, nhưng Pi đã được sự quan tâm từ nhiều đối tượng khác nhau. Đặc tính của Raspberry Pi xây dựng xoay quanh bộ xử lí SoC Broadcom BCM2835 (là chip xử lí mobile mạnh mẽ có kích thước nhỏ hay được dùng trong điện thoại di động) bao gồm CPU, GPU, bộ xử lí âm thanh /video, và nhiều tính năng khác.

Raspberry Pi không thể thay thế hoàn toàn hệ thống để bàn hoặc máy xách tay. Mặc dù không thể chạy Windows trên đó nhưng Raspberry vẫn có thể chạy bằng Linux với các tiện ích như lướt web và một vài nhiệm vụ khác. Raspberry Pi là một thiết bị đa năng đáng ngạc nhiên với nhiều phần cứng có giá thành rẻ nhưng rất hoàn hảo cho những hệ thống điện tử, tiêu biểu như dự án DIY thiết lập hệ thống tính toán cho những bài học trải nghiệm lập trình …

A.2. Phần cứng



Sơ đồ cấu tạo Raspberry Pi

Raspberry Pi có hai phiên bản, **Model A** và **Model B**. Model B như hình trên thông dụng hơn cả. Model B bao gồm những phần cứng và những cổng giao diện

- SoC 700MHz với 512MB RAM.
- 1 cổng HDMI cho đầu ra âm thanh / video số.
- 1 cổng video RCA cho đầu ra video Analog.
- Jack Headphone Stereo 3.5mm cho đầu ra âm thanh Analog.
- 02 cổng USB.
- 01 đầu đọc thẻ nhớ SD để tải hệ điều hành.
- 01 cổng Ethernet LAN.
- 01 giao diện GPIO (General Purpose Input/Output).

Model A cũng gần tương tự như Model B nhưng có sự khác biệt như sau

- 1 cổng USB
- Không có cổng Ethernet vì thế người dùng phải thêm Adapter USB Wi-Fi hoặc Ethernet nếu cần kết nối mạng.
- 256MB RAM.

A.3. Hệ điều hành và phần mềm

Về cơ bản Raspberry Pi có khá nhiều OS linux chạy được ngoại trừ Ubuntu (do CPU ARMv6). Một số Distributions Linux (nhúng) chạy trên Raspberry Pi như Raspbian, Pidora, openSUSE, OpenWRT, OpenELEC….

A.4. Ưu nhược điểm của Raspbery

Một số ưu nhược điểm của Raspberry Pi.

Ưu điểm:

- Giá rẻ.
- Nhỏ gọn.
- Siêu tiết kiệm điện.
- GPU mạnh.
- Phục vụ cho nhiều mục đích.
- Khả năng hoạt động liên tục 24/7.

Nhược điểm:

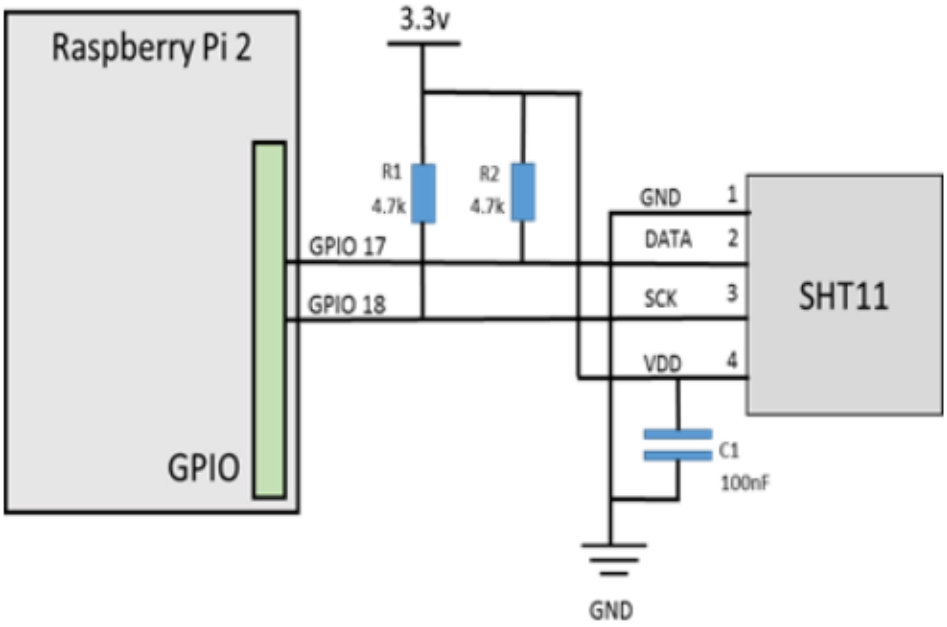
- CPU cấu hình thấp.
- Lan 100.
- Không có tích hợp WiFi (có thể mua USB WiFi về gắn vô).
- Yêu cầu phải có kiến thức cơ bản về Linux, điện tử.

A5. Thông số kỹ thuật của thiết bị Raspberry được thực nghiệm trong luận văn

Thành phần	Đặc tả
System-on-a-Chip	Broadcom BCM2836 (CPU + GPU. SDRAM is a separate chip stacked on top)
CPU	900MHz quad-core ARM Cortex-A7
GPU	Broadcom Video Core IV, OpenGL ES 2.0, OpenVG 1080p30 H.264 high-profile encode/decode)
Memory (SDRAM)	1024 MB
Power ratings	650 mA, (3.0 W)
Power source	5 V (DC) via Micro USB type B or GPIO header

B. SHT11

SHT11 là cảm biến nhiệt độ và độ ẩm. Nó ra đời sau và được sử dụng thay thế cho dòng SHT1x ở những nơi không cần độ chính xác cao về nhiệt độ và độ ẩm.



Kết nối Raspberry Pi 2 với SHT11

C. TIÊU CHUẨN CỦA MẬT MÃ NHẸ

Mỗi thuật toán lại có một ưu / nhược điểm riêng, phù hợp với từng trường hợp, yêu cầu cụ thể của từng bài toán. Bảng dưới đây mô tả một số tiêu chuẩn và thư viện của từng hệ mật mã trong từng trường hợp cụ thể, tránh sự chồng chéo trong sử dụng.

Tiêu chuẩn và thư viện liên quan đến mật mã nhẹ[9]

Loại	Tên	Hệ mật / Thư viện
ISO / IEC	29167	AES-128, PRESENT-80, Grain-128A
	29192-2	PRESENT, CLEFIA
	29192-3	Enocoro, Trivium
	29192-5	PHOTON, Lesamnta-LW, Spongent
	18033-3	AES, MISTY1, HIGHT
	18033-4	SNOW 2.0
Regoinal	FIPS 185 (USA)	Skipjack
	FIPS 197 (USA)	AES
	NESSIE (EU)	AES, MISTY1
	eSTREAM portfolio (EU)	Grain, Trivium, Salsa20, MICKEY

[9] Nguồn: Alex Biryukov and Léo Perrin, “State of the Art in Lightweight Symmetric Cryptography”

	GOST R 34.12-2015 (Russia)	Magma
Protocols	GSM	A5/1, A5/2, A5/3 (KASUMI)
	3G	SNOW 3G, ZUC, AES, KASUMI
	Bluetooth smart	E0, AES
	WEP	RC4
	WPA	RC4
	WPA2	AES
	Lora Alliance	AES
	IEEE 802.15.4 (Zigbee)	AES
Embedded Lib.	Tinysec	Skipjack (CBC), (RC5)
	Minisec	Skipjack (OCB)
	mbedTLS (ciphers)	AES, RC4, XTEA, Blowfish, 3-DES, Camellia
	mbedTLS (hash function)	MD5, SHA-1. SHA-256, SHA-512

C.1. Tiêu chuẩn mã hóa ISO/IEC

Tổ chức tiêu chuẩn quốc tế (The International Organization for Standards – ISO) và Ủy ban kỹ thuật điện quốc tế (The International Electrotechnical Commission – IEC) có nhiệm vụ ban hành và duy trì các tiêu chuẩn về thông tin và công nghệ truyền thông. Tính đến thời điểm hiện tại, có ba tiêu chuẩn của họ đặc biệt phù hợp với mật mã nhẹ. Đầu tiên là tiêu chuẩn ISO/IEC 29167: Công nghệ thông tin – Kỹ thuật nhận diện và thu thập dữ liệu tự động, trong phần 10, 11 và 13. Cụ thể là mật mã đối xứng nên được sử dụng trong "air interface communications" (giao tiếp không gian), trong các thẻ RFID. Các phần này được mô tả cụ thể trong AES-128, PRESENT-80 và Grain-128A.

Một tiêu chuẩn khác có liên quan mật thiết đến mật mã nhẹ là tiêu chuẩn ISO/IEC 29192 với một loạt các tiêu chuẩn về mã khối như PRESENT, CLEFIA, mã dòng như Trivium, Enocoro hay hàm băm PHOTON, Spongent và Lesamnta-LW. Dưới đây là một vài tiêu chuẩn của một hệ mật mã nhẹ được đề cập trong ISO/IEC 29192

- Sự an toàn của cơ chế mã hóa. Bảo mật 80 bits được xem là sức mạnh an ninh tối thiểu cho một hệ mật mã nhẹ. Tuy nhiên, tiêu chuẩn cũng đề nghị rằng ít nhất phải áp dụng bảo mật 112 bits cho các hệ thống yêu cầu bảo mật trong thời gian dài.
- Yêu cầu khi triển khai phần cứng. Ví dụ như vùng chip được sử dụng cho cơ chế mã hóa, sự tiêu thụ năng lượng…

- Yêu cầu khi triển khai phần mềm. Đặc biệt về code size, kích thước RAM
- Sự trưởng thành của cơ chế mã hóa
- Tổng quát các thuộc tính nhẹ được yêu cầu cho một hệ mật. Ví dụ như trọng lượng nhẹ trong một khoảng cho phép.

C.2. Tiêu chuẩn mã hóa khu vực

Tại USA, các tiêu chuẩn mã hoá được xử lý bởi Viện Tiêu chuẩn và Công nghệ Quốc gia (NIST). Tổ chức này hiện đang làm việc hướng tới một tiêu chuẩn về mật mã học nhẹ. Ý định của họ là thỏa thuận một số cấu hình tương ứng với các thuật toán khác nhau, các trường hợp sử dụng và khó khăn. Sau đó, có thể các thuật toán khác nhau sẽ được chuẩn hóa để sử dụng trong mỗi cấu hình này.

Ở châu Âu, dự án Nessie đã chọn một số mật mã khối bao gồm AES và MISTY1. Sự thất bại của nó để tìm mật mã dòng tốt dẫn đến cuộc cạnh tranh eSTREAM. Cuối cùng, một danh mục các mật mã dòng được xuất bản. Nó được chia thành hai cấu hình, một phần mềm định hướng và một phần cứng theo định hướng. Một vài hệ mật mã dòng có thể được coi là nhẹ: Trivium, Grain, Mickey và Salsa20.

Cuối cùng, tiêu chuẩn mới nhất của Nga về mật mã khối chứa mật mã khối 64 bits Magma.

C.3. Giao thức truyền thông

Một số giao thức truyền thông chỉ định một vài hệ mật mã có yêu cầu nhẹ. Ví dụ như điện thoại di động không cần hệ mật mạnh mẽ như máy tính. Mạng GSM và 3G xử lý truyền thông trên điện thoại di động yêu cầu cần có mã hóa A5/1, A5/2 hay A5/3…

Các kết nối wifi được bảo mật bằng cách sử dụng WPA hoặc WPA2. Một số giao thức gần đây được đề xuất để kết nối các thiết bị IoT không dây với các thiết bị khác. Một trong những đề xuất này là AES. Điều này cũng đúng với IEEE 802.15.4, được sử dụng trong Zigbee.

C.4. Thư viện định hướng IoT

Chúng ta hãy xem xét hai thư viện dành cho các thiết bị nhúng. Đầu tiên là Tinysec được sử dụng trong ngăn xếp liên quan đến an ninh của hệt điều hành TinyOS. Nó sử dụng Skipjack ở chế độ CBC.

Thứ hai là thư viện minisec cũng dùng cho thiết bị nhúng nhưng không phục vụ TinyOS. Thư viện này bao gồm một số thuật toán mã hóa AES, RC4, XTEA, Blowfish… cũng như một số hàm băm MD5, SHA-1, SHA-256 và SHA-512.

D. MÃ NGUỒN

Mục này chỉ miêu tả mã nguồn của thuật toán cốt lõi được sử dụng trong chương 3 – Grain-128.

D.1. Grain128.c

```
#include "ecrypt-sync.h"

void ECRYPT_init(void){}
/*
 * Function: grain_keystream
 *
 * Description:
 *  Generates a new bit and updates the internal state of the cipher.
 */

u8 grain_keystream( ECRYPT_ctx* ctx )
{
	u8 i, NBit, LBit, outbit;

	/* Calculate feedback and output bits */
	// Output of Grain-128
	outbit = ctx->NFSR[2]^ctx->NFSR[15]^ctx->NFSR[36]^ctx->NFSR[45]^
			ctx->NFSR[64]^ctx->NFSR[73]^ctx->NFSR[89]^ctx->LFSR[93]^
			(ctx->NFSR[12]&ctx->LFSR[8])^(ctx->LFSR[13]&ctx->LFSR[20])^
			(ctx->NFSR[95]&ctx->LFSR[42])^(ctx->LFSR[60]&ctx->LFSR[79])^
			(ctx->NFSR[12]&ctx->NFSR[95]&ctx->LFSR[95]);
	/* Output of NFSR function
	 b_(i+128)= s_i+b_i+b_(i+26)+b_(i+56)+b_(i+91)
				+b_(i+96)+b_(i+3) b_(i+67)+b_(i+11) b_(i+13)
				+b_(i+17) b_(i+18)+b_(i+27) b_(i+59)
				+b_(i+40) b_(i+48)+b_(i+61) b_(i+65)
			+b_(i+68) b_(i+84)
	*/
	NBit=ctx->LFSR[0]^ctx->NFSR[0]^ctx->NFSR[26]^ctx->NFSR[56]^ctx->NFSR[91]^ctx->NFSR[96]^
		(ctx->NFSR[3]&ctx->NFSR[67])^(ctx->NFSR[11]&ctx->NFSR[13])^
		(ctx->NFSR[17]&ctx->NFSR[18])^(ctx->NFSR[27]&ctx->NFSR[59])^
		(ctx->NFSR[40]&ctx->NFSR[48])^(ctx->NFSR[61]&ctx->NFSR[65])^
		(ctx->NFSR[68]&ctx->NFSR[84]);

	// Output of LFSR function
	// s_(i+128)=s_i+s_(i+7)+s_(i+38)+s_(i+70)+s_(i+81)+s_(i+96)
	LBit=ctx->LFSR[0]^ctx->LFSR[7]^ctx->LFSR[38]^ctx->LFSR[70]^ctx->LFSR[81]^ctx->LFSR[96];

	/* Update registers */
	for (i = 1; i < (ctx->keysize); ++i)
	{
		ctx->NFSR[i-1] = ctx->NFSR[i];
		ctx->LFSR[i-1] = ctx->LFSR[i];
	}

	ctx->NFSR[(ctx->keysize)-1] = NBit;
	ctx->LFSR[(ctx->keysize)-1] = LBit;

	return outbit;
}


/* Functions for the ECRYPT API */
/*
 * @param: ctx
```

```
 *         keysize   Key size in bits
 *         ivsize    IV size in bits
*/

void ECRYPT_keysetup( ECRYPT_ctx* ctx, const u8* key, u32 keysize, u32 ivsize )
{
       ctx->p_key = key;
       ctx->keysize = keysize;
       ctx->ivsize = ivsize;
}

/*
 * Function: ECRYPT_ivsetup
 *
 * Description
 *  Load the key and perform initial clockings.
 *
 * Assumptions
 *  The key is 16 bytes and the IV is 12 bytes. The
 *  registers are loaded in the following way:
 *
 *  NFSR[0] = lsb of key[0]
 *  ...
 *  NFSR[7] = msb of key[0]
 *  ...
 *  ...
 *  NFSR[120] = lsb of key[16]
 *  ...
 *  NFSR[127] = msb of key[16]
 *  LFSR[0] = lsb of IV[0]
 *  ...
 *  LFSR[7] = msb of IV[0]
 *  ...
 *  ...
 *  LFSR[88] = lsb of IV[12]
 *  ...
 *  LFSR[95] = msb of IV[12]
 */
void ECRYPT_ivsetup ( ECRYPT_ctx* ctx, const u8* iv )
{
       u32 i, j;
       u8 outbit;
       /* Load registers */
       for (i = 0; i <(ctx->ivsize)/8; ++i)
       {
              for (j = 0; j < 8; ++j)
              {
                     ctx->NFSR[i*8 + j] = ((ctx->p_key[i] >> j) & 1);
                     ctx->LFSR[i*8 + j] = ((iv[i] >> j) & 1);
              }
       }
       for (i = (ctx->ivsize)/8; i < (ctx->keysize)/8; ++i)
       {
              for (j = 0; j < 8; ++j)
              {
                     ctx->NFSR[i*8 + j] = ((ctx->p_key[i] >> j) & 1);
                     ctx->LFSR[i*8 + j] = 1;
              }
       }

       /* Do initial clockings */
       for (i = 0; i < 256; ++i)
       {
              outbit = grain_keystream(ctx);
              ctx->LFSR[127] ^= outbit;
```

```
                ctx->NFSR[127] ^= outbit;
        }
}

/*
 * Function: ECRYPT_keystream_bytes
 *
 * Synopsis
 *  Generate keystream in bytes.
 *
 * Assumptions
 *  Bits are generated in order z0,z1,z2,...
 *  The bits are stored in a byte in order:
 *
 *  lsb of keystream[0] = z0
 *  ...
 *  msb of keystream[0] = z7
 *  ...
 *  lsb of keystream[1] = z8
 *  ...
 *  msb of keystream[1] = z15
 *  ...
 *  ...
 *  ...
 */
void ECRYPT_keystream_bytes( ECRYPT_ctx* ctx, u8* keystream, u32 msglen )
{
        u32 i, j;
        for (i = 0; i < msglen; ++i)
        {
                keystream[i] = 0;
                for (j = 0; j < 8; ++j)
                {
                        keystream[i] |= (grain_keystream(ctx) << j);
                }
        }
}

/*
* Function: ECRYPT_encrypt_bytes
*
* Synopsis
*  Generate ciphertext bytes from plaintext bytes.
*/
void ECRYPT_encrypt_bytes( ECRYPT_ctx* ctx, const u8* plaintext, u8* ciphertext, u32
msglen )
{
        u32 i, j;
        u8 k = 0;
        for (i = 0; i < msglen; ++i)
        {
                k = 0;
                for (j = 0; j < 8; ++j)
                {
                        k |= (grain_keystream(ctx) << j);
                }
                ciphertext[i] = plaintext[i]^k;
        }
}

/*
* Function: ECRYPT_decrypt_bytes
*
* Synopsis
*  Generate plaintext bytes from ciphertext bytes.
```

```
*/
void ECRYPT_decrypt_bytes( ECRYPT_ctx* ctx, const u8* ciphertext, u8* plaintext, u32
msglen )
{
	u32 i, j;
	u8 k = 0;
	for (i = 0; i <msglen; ++i)
	{
		k = 0;
		for (j = 0; j < 8; ++j)
		{
			k |= (grain_keystream(ctx)<<j);
		}
		plaintext[i] = ciphertext[i]^k;
	}
}
```

D.2. ecrypt-sync.c

```
#include "ecrypt-sync.h"

#ifdef ECRYPT_USES_DEFAULT_ALL_IN_ONE

/*
 * Default implementation of all-in-one encryption/decryption of
 * (short) packets.
 */

void ECRYPT_encrypt_packet(
	ECRYPT_ctx* ctx,
	const u8* iv,
	const u8* plaintext,
	u8* ciphertext,
	u32 msglen)
{
	// Setup IV
	ECRYPT_ivsetup(ctx, iv);

	// Encrypt for list bytes
	ECRYPT_encrypt_bytes(ctx, plaintext, ciphertext, msglen);
}

void ECRYPT_decrypt_packet(
	ECRYPT_ctx* ctx,
	const u8* iv,
	const u8* ciphertext,
	u8* plaintext,
	u32 msglen)
{
	// Setup IV
	ECRYPT_ivsetup(ctx, iv);

	// Decrypt for list bytes
	ECRYPT_decrypt_bytes(ctx, ciphertext, plaintext, msglen);
}

#endif
```

D.3. Ví dụ kết quả mã hóa và giải mã với Grain-128

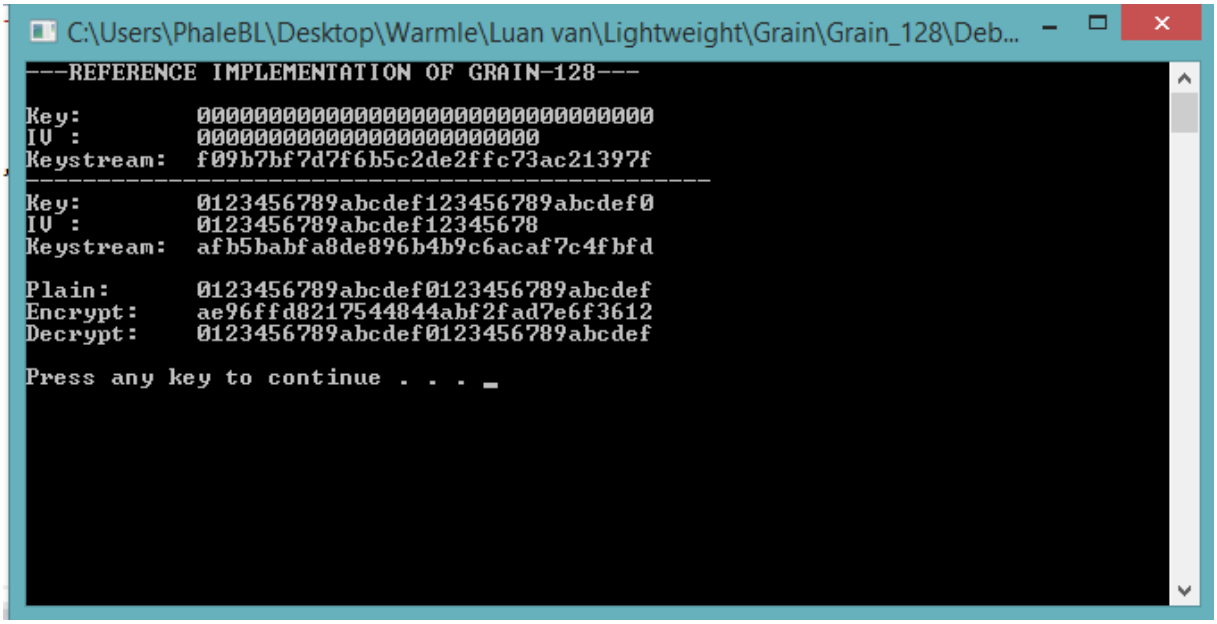
Ví dụ quá trình tạo khóa dòng

```
Key: 00000000000000000000000000
IV : 000000000000
```

```
Keystream: f09b7bf7d7f6b5c2de2ffc73ac21397f
```

Ví dụ quá trình mã hóa và giải mã

```
Key: 0123456789abcdef123456789abcdef0
IV : 0123456789abcdef12345678
Keystream : afb5babfa8de896b4b9c6acaf7c4fbfd
Plaintext : 0123456789abcdef0123456789abcdef
Ciphertext: ae96ffd8217544844abf2fad7e6f3612
```



Ví dụ về Grain

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ
------🕮------

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc
****

Hà Nội, ngày 02 tháng 12 năm 2017

QUYẾT NGHỊ
CỦA HỘI ĐỒNG CHẤM LUẬN VĂN THẠC SĨ

Căn cứ Quyết định số **1140/QĐ-ĐT, ngày 23 tháng 11 năm 2017** của Hiệu trưởng trường Đại học Công nghệ về việc thành lập Hội đồng chấm luận văn thạc sĩ của học viên **Lê Thị Len,** Hội đồng chấm luận văn Thạc sĩ đã họp vào **15h, thứ 7, ngày 02 tháng 12 năm 2017, Phòng 102, Nhà E3,** Trường Đại học Công nghệ - ĐHQGHN.

Tên đề tài luận văn: **Mật mã dòng trong mật mã nhẹ và triển vọng trong IoT**

Ngành: **Công nghệ Thông tin**

Chuyên ngành: **Hệ thống thông tin** Mã số:

Sau khi nghe học viên trình bày tóm tắt luận văn Thạc sĩ, các phản biện đọc nhận xét, học viên trả lời các câu hỏi, Hội đồng đã họp, trao đổi ý kiến và thống nhất kết luận:

1. Về tính cấp thiết, tính thời sự, ý nghĩa lý luận và thực tiễn của đề tài luận văn:

- Luận văn có tính cấp thiết và thời sự.
- Luận văn có ý nghĩa lý luận và thực tiễn cao

2. Về bố cục, phương pháp nghiên cứu, tài liệu tham khảo, ... của luận văn:

- Luận văn có bố cục phù hợp với một luận văn khoa học.
- Phương pháp nghiên cứu và thử nghiệm được thực hiện tốt.

3. Về kết quả nghiên cứu:

Xây dựng hệ thống Smart Home sử dụng mật mã nhẹ cho quá trình truyền thông tin.

- Có các thử nghiệm và đánh giá đầy đủ.

4. Hạn chế của luận văn *(nếu có)*:

- Văn phong của luận văn đôi chỗ còn chưa phù hợp.
- Kịch bản thử nghiệm cần chặt chẽ hơn.

5. Đánh giá chung và kết luận:

- Luận văn đáp ứng các yêu cầu của một luận văn khoa học.
- Luận văn cần chỉnh sửa theo các góp ý của đề tài.

Luận văn đạt 8,7 / 10 điểm. Quyết nghị này được 05 / 05 thành viên của Hội đồng nhất trí thông qua.

THƯ KÝ HỘI ĐỒNG

Trần Trọng Hiếu

CHỦ TỊCH HỘI ĐỒNG

Nguyễn Ngọc Hoá

XÁC NHẬN CỦA CƠ SỞ ĐÀO TẠO

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

Hà Nội ngày 30/11/2017

BẢN NHẬN XÉT PHẢN BIỆN LUẬN VĂN THẠC SĨ

Họ và tên cán bộ phản biện: Nguyễn Ngọc Cương

Học hàm, học vị: Tiến sĩ

Cơ quan công tác: Phó Cục trưởng Cục Công nghệ thông tin – Bộ Công an

Họ và tên học viên cao học: Lê Thị Len

Tên đề tài luận văn: Mật mã dòng trong mật mã nhẹ và triển vọng trong IoT

Chuyên ngành: Hệ thống thông tin Mã số: 60.48.01.04

Ý KIẾN NHẬN XÉT

1. Tính cấp thiết, tính thời sự, ý nghĩa khoa học và thực tiễn của đề tài luận văn

Trong xu thế phát triển mạnh mẽ của cách mạng công nghiệp lần thứ 4, các thiết bị kết nối internet, sẽ gia tăng với tốc độ vũ bão (*đến năm 2020 sẽ có hơn 50 tỷ thiết bị kết nối*) đã đặt bài toán đảm bảo các kết nối được an ninh và an toàn của các thiết bị đó dưới một góc độ mới, đòi hỏi cần phải có các hệ mật vừa có độ mật cần thiết nhưng vừa tiêu tốn ít năng lượng, bộ nhớ và các cổng logic. Đây là điều kiện để các hệ mật mã nhẹ, gồm mã khối hạng nhẹ, mã dòng hạng nhẹ và các mã xác thực hạng nhẹ được quan tâm nghiên cứu và phát triển mạnh mẽ trong thời gian gần đây.

Luận văn của học viên Lê Thị Len có mục tiêu và nội dung: tìm hiểu một số hệ mật trong mật mã nhẹ, mật mã dòng và khả năng ứng dụng sau đó đề xuất xây dựng một kênh truyền tin an toàn bằng phương pháp mã hóa đầu cuối sử dụng mã dòng Grain và thử nghiệm, đánh giá- theo người nhận xét là có ý nghĩa thời sự, có ý nghĩa thực tiễn.

2. Sự không trùng lặp của đề tài nghiên cứu so với các công trình khoa học, luận văn đã công bố ở trong và ngoài nước; tính trung thực, rõ ràng và đầy đủ trong trích dẫn tài liệu tham khảo

Luận văn của Học viên Lê Thị Len cùng hướng với nhiều nghiên cứu đã có nhưng học viên có các kết quả nghiên cứu riêng nên không trùng lặp hoàn toàn với các công trình đã công bố trước.

3. Sự phù hợp giữa tên đề tài với nội dung nghiên cứu cũng như với chuyên ngành và mã số đào tạo

Tên đề tài phù hợp với nội dung nghiên cứu và phù hợp với chuyên ngành và mã số đào tạo. Việc nghiên cứu mật mã dòng trong mật mã hạng nhẹ để ứng dụng xây dựng hệ thống truyền tin an toàn là nội dung của chuyên ngành hệ thống thông tin.

4. Độ tin cậy và tính hiện đại của phương pháp nghiên cứu đã sử dụng để hoàn thành uận văn

Học viên sử dụng phương pháp tổng hợp tài liệu, nghiên cứu thuật toán, đề xuất mô hình và lập trình thử nghiệp, đánh giá so sánh với các kết quả đã có – là phương pháp nghiên cứu phù hợp, hiện đại được các nhà khoa học sử dụng.

5. Kết quả nghiên cứu mới của tác giả, đóng góp mới cho sự phát triển chuyên ngành, đóng góp mới phục vụ sản xuất, kinh tế, xã hội, an ninh, quốc phòng và đời sống. Giá trị và độ tin cậy của những kết quả nghiên cứu

- Tác giả đã tìm hiểu tổng quan về mật mã hạng nhẹ trong đó đi sâu vào hệ mật mã dòng tiêu biểu là Grain và khả năng ứng dụng Grain trong các thiết bị IoT.

- Sử dụng mật mã Grain và mã xác thực thông báo với hàm băm nhẹ Keccak trong sử dụng thiết bị Rasberry để thu thập dữ liệu từ cảm biến SHT11 và so sánh với các thuật toán mã hóa khác; ứng dụng vào kênh truyền tin an toàn.

Các kết quả trong luận văn là xác thực, có độ tin cậy.

6. Nhận xét về nội dung, bố cục và hình thức của luận văn

Luận văn được kết cấu hợp lý, bố cục logic.

Tuy nhiên trong phần thực nghiệm nên mô tả rõ hơn kịch bản thử nghiệm.

7. Các ý kiến nhận xét khác (về khả năng viết báo, phát triển sản phẩm, hoặc định hướng nghiên cứu tiếp theo,...)

Luận văn đã được đăng báo được đăng nhưng mức độ đóng góp của tác giả trong bài báo đó chưa được làm rõ.

8. Kết luận chung

Luận văn về cơ bản đáp ứng yêu cầu của một luận văn thạc sĩ. Bản tóm tắt luận văn phản ánh trung thực nội dung cơ bản của luận văn. Tôi đề nghị cho học viên bảo vệ luận văn trước Hội đồng chấm luận văn để nhận học vị Thạc sĩ.

Người nhận xét

TS. Nguyễn Ngọc Cương

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập - Tự do - Hạnh phúc

==========

BẢN NHẬN XÉT PHẢN BIỆN LUẬN VĂN THẠC SĨ

Họ và tên cán bộ phản biện: Nguyễn Trường Thắng

Học hàm, học vị: Tiến sỹ

Chuyên ngành: Công nghệ phần mềm

Cơ quan công tác: Viện Công nghệ Thông tin – Viện Hàn lâm Khoa học và Công nghệ Việt Nam

Họ và tên học viên cao học: Lê Thị Len

Tên đề tài luận văn: Mật mã dòng trong mật mã nhẹ và triển vọng trong IoT

Chuyên ngành: Hệ thống thông tin Mã số: 60 48 01 04

Ý KIẾN NHẬN XÉT

1. Tính cấp thiết, tính thời sự, ý nghĩa khoa học và thực tiễn của đề tài luận văn

Với sự phát triển của CNTT gần đây trong CMCN 4.0 với nền tảng kết nối là Internet of Things (IoT), vấn đề đảm bảo an toàn dữ liệu/an ninh mạng đối với các hệ thống này rất quan trọng. Trong khi những hệ thống PC, server sử dụng chip đa dụng của Intel, AMD có công năng tính toán cao thì các thiết bị cảm biến IoT bị hạn chế về năng lực tính toán, bộ nhớ cùng với thời hạn pin của thiết bị. Kéo theo đó là những giải pháp mật mã AES, DES… không phù hợp. Vấn đề này trở nên cấp thiết gần đây và giải pháp là những hệ mật mã nhẹ (light-weight).

Luận án này đi sâu vào hệ mật mã hạng nhẹ (streaming) Grain và cài đặt thử nghiệm trên Single-Board-Computer (SCB) với chip Raspberry Pi. Như đã nói ở trên, SCB không có công năng tính toán mạnh nhưng giá rẻ, phù hợp yêu cầu tính toán cho các hệ cảm biến, nhúng… nên hoàn toàn là lựa chọn phù hợp.

Luận án có ý nghĩa thực tiễn, có tính thời sự nhưng tính khoa học chưa rõ ràng. Bản thân nền tảng khoa học của hệ mật mã nhẹ nói chung, Grain cũng chưa đi sâu phân tích mà chỉ ở mức giới thiệu. Phần cài đặt thử nghiệm không cho thấy rõ việc hệ mật mã nhẹ này có được tích hợp vào SCB chứa bộ cảm biến cũng như ở hệ thống quản lý trung tâm hay không?

2. Sự không trùng lặp của đề tài nghiên cứu

IoT và những giải pháp bảo đảm an toàn thông tin trên đó là lĩnh vực mới ở Việt Nam khoảng 2-3 năm gần đây, sự tích hợp của nhiều công nghệ. Luận văn dừng ở mức mô hình tổng quan nên không có sự trùng lặp về mặt khoa học và thử nghiệm.

3. Sự phù hợp giữa tên đề tài với nội dung nghiên cứu

Tên đề tài và nội dung tương đối phù hợp – tập trung vào điểm chính sau:

- Mật mã nhẹ và loại mật mã dòng (Chương 1) khoảng 20 trang;
- Họ mật mã dòng Grain (Chương 2) khoảng 20 trang;
- Cài đặt thử nghiệm một SCB dựa trên chip Raspberry Pi trong Chương 3 với khoảng 25 trang;

4. Kết quả nghiên cứu mới của tác giả

- Không có kết quả mới về mặt lý thuyết và khoa học. Ở đây chỉ là giới thiệu tổng qua lý thuyết hệ mật mã nhẹ và sử dụng họ Grain (thiếu so sánh, cơ sở lựa chọn giữa các loại hệ nhẹ);

5. Nhận xét về nội dung, bố cục và hình thức của luận văn

- Hình thức luận văn tương đối rõ ràng, phân bổ hợp lý giữa các chương. Tuy nhiên cần đi sâu hơn vào chi tiết kỹ thuật;
- Danh sách tài liệu tham khảo không đúng chuẩn. Ví dụ: không theo thứ tự ABC (phần tiếng Anh cần theo thứ tự của họ). Việc trích dẫn các tài liệu cũng rất sơ sài so với danh sách có trong tài liệu tham khảo, cụ thể nhiều đoạn giới thiệu trong Chương 1-2 không có trích dẫn;

6. Các ý kiến nhận xét khác

- Cần làm rõ phần cài đặt sử dụng công cụ có sẵn hay tự phát triển, ưu nhược điểm của nó. Người nhận xét không rõ phần Grain được tích hợp trên SCB như thế nào? Tương tự như vậy tại hệ quản lý trung tâm và cơ chế truyền dẫn từ thiết bị tới trung tâm thông qua kênh truyền với giao thức nào, có E2EE không;

7. Kết luận chung

Chất lượng khoa học của luận văn có tính hệ thống. Luận văn có nhiều ý nghĩa bước đầu về mảng nóng bỏng, liên ngành hiện nay là IoT và hệ mật mã nhẹ cùng thực nghiệm với giải pháp SCB/Raspberry.

Luận văn có thể đưa ra bảo vệ trước hội đồng: Đồng ý.

Hà Nội, ngày 28 tháng 11 năm 2017

XÁC NHẬN CỦA CƠ QUAN **CÁN BỘ PHẢN BIỆN**

Nguyễn Trường Thắng

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Hà Nội, ngày 13 tháng 12 năm 2017

BẢN XÁC NHẬN ĐÃ SỬA CHỮA CÁC THIẾU SÓT CỦA LUẬN VĂN

Trường Đại học Công nghệ (ĐHCN) đã có Quyết định số 1140/QĐ-ĐT ngày 23 tháng 11 năm 2017 về việc thành lập Hội đồng chấm luận văn Thạc sĩ cho học viên Lê Thị Len, sinh ngày 21/08/1993, tại Bắc Ninh, chuyên ngành Hệ thống thông tin, ngành Hệ thống thông tin.

Ngày 02 tháng 12 năm 2017, Trường ĐHCN đã tổ chức cho học viên bảo vệ luận văn Thạc sĩ trước Hội đồng chấm (có biên bản kèm theo). Theo Quyết nghị của Hội đồng chấm luận văn Thạc sĩ, học viên phải bổ sung và sửa chữa các điểm sau đây trước khi nộp quyển luận văn cuối cùng cho Nhà trường để hoàn thiện hồ sơ sau bảo vệ:

1. Sử dụng văn phong phù hợp với luận văn nghiên cứu.
2. Kịch bản thử nghiệm cần được nêu rõ.
3. Mô tả chi tiết thiết kế phần cứng của thực nghiệm.
4. Chỉnh sửa bố cục chương 2.

Ngày 13 tháng 12 năm 2017, học viên đã nộp bản luận văn có chỉnh sửa. Chúng tôi nhận thấy rằng nội dung, hình thức của luận văn và tóm tắt luận văn đã được sửa chữa, bổ sung theo các điểm trên của Quyết nghị.

Đề nghị Trường Đại học Công nghệ, Đại học Quốc gia Hà Nội cho phép học viên được làm các thủ tục khác để được công nhận và cấp bằng Thạc sĩ.

Xin trân trọng cảm ơn!

XÁC NHẬN CỦA THÀNH VIÊN HỘI ĐỒNG/HỘI ĐỒNG
ĐỀ NGHỊ HỌC VIÊN SỬA CHỮA LUẬN VĂN

Nguyễn Ngọc Cường

Nguyễn Thanh Thủy

HỌC VIÊN	**CÁN BỘ HƯỚNG DẪN**	**XÁC NHẬN CỦA TRƯỜNG ĐẠI HỌC CÔNG NGHỆ**
Lê Thị Len	Phùng Văn Ổn	

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Số: 1140 /QĐ-ĐT

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

Hà Nội, ngày 23 tháng 11 năm 2017

QUYẾT ĐỊNH
Về việc thành lập Hội đồng chấm luận văn thạc sĩ

HIỆU TRƯỞNG
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Căn cứ Quy định về Tổ chức và hoạt động của các đơn vị thành viên và đơn vị trực thuộc Đại học Quốc gia Hà Nội ban hành theo quyết định số 3568/QĐ-ĐHQGHN ngày 08/10/2014 của Giám đốc Đại học Quốc gia Hà Nội;

Căn cứ Quy định về Tổ chức và hoạt động của Trường ĐH Công nghệ ban hành kèm theo Quyết định số 520/QĐ-ĐHCN ngày 19/7/2016 của Hiệu trưởng Trường ĐH Công nghệ;

Căn cứ Quy chế Đào tạo thạc sĩ tại Đại học Quốc gia Hà Nội, ban hành theo Quyết định số 4668/QĐ-ĐHQGHN ngày 10/12/2014 của Giám đốc Đại học Quốc gia Hà Nội ;

Căn cứ Quyết định công nhận học viên cao học số 1005/QĐ-CTSV ngày 27/12/2015 của Hiệu trưởng Trường Đại học Công nghệ;

Căn cứ Công văn số 153/CNTT-ĐT ngày 31/10/2017 và Công văn số 169/CNTT-ĐT ngày 21/11/2017 của Chủ nhiệm Khoa Công nghệ thông tin về việc đề xuất hội đồng chấm luận văn;

Xét đề nghị của Trưởng phòng Đào tạo,

QUYẾT ĐỊNH:

Điều 1. Thành lập Hội đồng chấm luận văn thạc sĩ của học viên Lê Thị Len, sinh ngày 21/08/1993 tại Bắc Niinh là học viên cao học khóa 22,

Ngành: Hệ thống thông tin

Chuyên ngành: Hệ thống thông tin Mã số: 60480104

Tên đề tài luận văn: Mật mã dòng trong mật mã nhẹ và triển vọng trong IoT.

Cán bộ hướng dẫn: TS. Lê Phê Đô (CBHD chính)

TS. Phùng Văn Ổn (CBHD phụ)

Danh sách các thành viên Hội đồng kèm theo quyết định này.

Điều 2. Chủ nhiệm Khoa Công nghệ thông tin có nhiệm vụ tổ chức để học viên bảo vệ luận văn thạc sĩ trước Hội đồng theo đúng Quy chế Đào tạo thạc sĩ ở Đại học Quốc gia Hà Nội và các quy định hiện hành khác. Hội đồng tự giải thể sau khi hoàn thành nhiệm vụ.

Điều 3. Trưởng phòng Hành chính – Quản trị, Trưởng phòng Đào tạo, Chủ nhiệm Khoa Công nghệ thông tin, các Thủ trưởng đơn vị có liên quan, các thành viên Hội đồng và học viên Lê Thị Len chịu trách nhiệm thi hành quyết định này./.

KT. HIỆU TRƯỞNG
PHÓ HIỆU TRƯỞNG

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

Chử Đức Trình

Nơi nhận:
- Như Điều 3;
- Lưu: VT, ĐT, CH11.

ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

DANH SÁCH HỘI ĐỒNG CHẤM LUẬN VĂN THẠC SĨ

(theo Quyết định số: 1440/QĐ-ĐT ngày 23 tháng 11 năm 2017
của Hiệu trưởng Trường Đại học Công nghệ)

STT	Họ và tên	Cơ quan công tác	Trách nhiệm trong Hội đồng
1	PGS. TS. Nguyễn Ngọc Hóa	Trường ĐH Công nghệ, ĐHQGHN	Chủ tịch
2	TS. Trần Trọng Hiếu	Trường ĐH Công nghệ, ĐHQGHN	Thư ký
3	TS. Nguyễn Ngọc Cương	Bộ Công an	Phản biện 1
4	TS. Nguyễn Trường Thắng	Viện Công nghệ thông tin, Viện Hàn lâm KH&CN Việt Nam	Phản biện 2
5	TS. Nguyễn Tuệ	Trường ĐH Công nghệ, ĐHQGHN	Ủy viên

Hội đồng gồm có 05 thành viên./.