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

TÓM TẮT LUẬN VĂN THẠC SĨ

Ngành: Hệ thống thông tin

HÀ NỘI - 2017

MỞ ĐẦU

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

Do sự phát triển của Tính toán khắp nơi (ubiquitous computing) người ta cần những thuật toán nhẹ để có thể cài đặt trong các thiết bị Thâm nhập khắp nơi (pervasive devices) với kích thước nhỏ, bộ vi điều khiển hoặc vi xử lý có khả năng tính toán hạn chế, phục vụ cho những bài toán chuyên dụng. Vì thế mà mật mã nhẹ (lightweight cryptograhy) với các thuật toán có khả năng tính toán nhanh, an toàn và chi phí thực hiện thấp ra đời và ngày càng phát triển.

Tùy từng yêu cầu bảo mật của thiết bị, cũng như khả năng đáp ứng của chúng 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).

Với các thiết bị IoT này, có một vài mối nguy hiểm và cảnh báo mà chúng ta cần nhận thức rõ, như là vấn đề bảo mật. 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 các 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 4 chương:

Chương 1: Giới thiệu tổng quan về mật mã nhẹ, mật mã dòng trong mật mã nhẹ, một số khái niệm quan trọng 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 là trong IoT.

Chương 2: Tìm hiểu một số hệ mật mã dòng nhẹ phổ biến hiện nay.

Chương 3: Nghiên cứu và đánh giá về một hệ mật mã dòng tiêu biểu trong mật mã nhẹ – Grain và khả năng ứng dụng mật mã dòng nhẹ Grain trong IoT.

Chương 4: 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 nhẹ trên Raspberry so với các 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 các hệ mật mã nhẹ khác.

Chương 1. MẬT MÃ DÒNG TRONG MẬT MÃ NHẸ

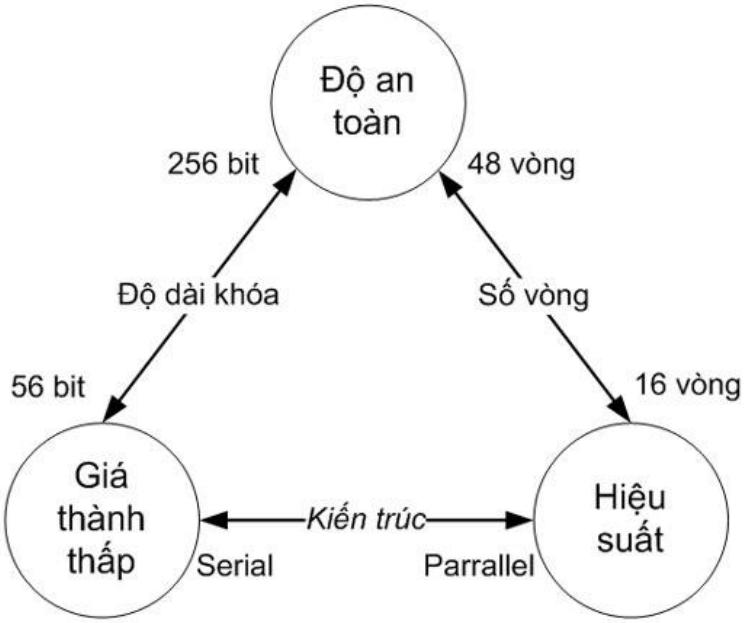
1.1. Mật mã nhẹ

Trong phần tổng quan chung của tiêu chuẩn ISO/IEC 29192-1 [1] đã đưa ra khái niệm cơ bản về mật mã nhẹ. Mật mã nhẹ là một loại mật mã dùng cho mục đích bảo mật, xác thực, nhận dạng và trao đổi khóa; phù hợp cài đặt cho những môi trường hạn chế. Những hạn chế đó dựa trên các đánh giá về diện tích chip (chiparea), năng lượng tiêu thụ (energy consumption), kích cỡ mã nguồn chương trình (program code size) kích cỡ RAM, băng thông (communication bandwidth) và thời gian thực thi (execution time). Trong những trường hợp này, sử dụng các thuật toán mã khối nhẹ là phù hợp và cần được quan tâm nghiên cứu.

1.1.1. 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 40 năm mới có sự ra đời và áp dụng chính thức của những giải thuật mật mã nhẹ đầu tiên: DES (1977), AES (1998), 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.

1.1.2. Nguyên lý thiết kế thuật toán mật mã nhẹ

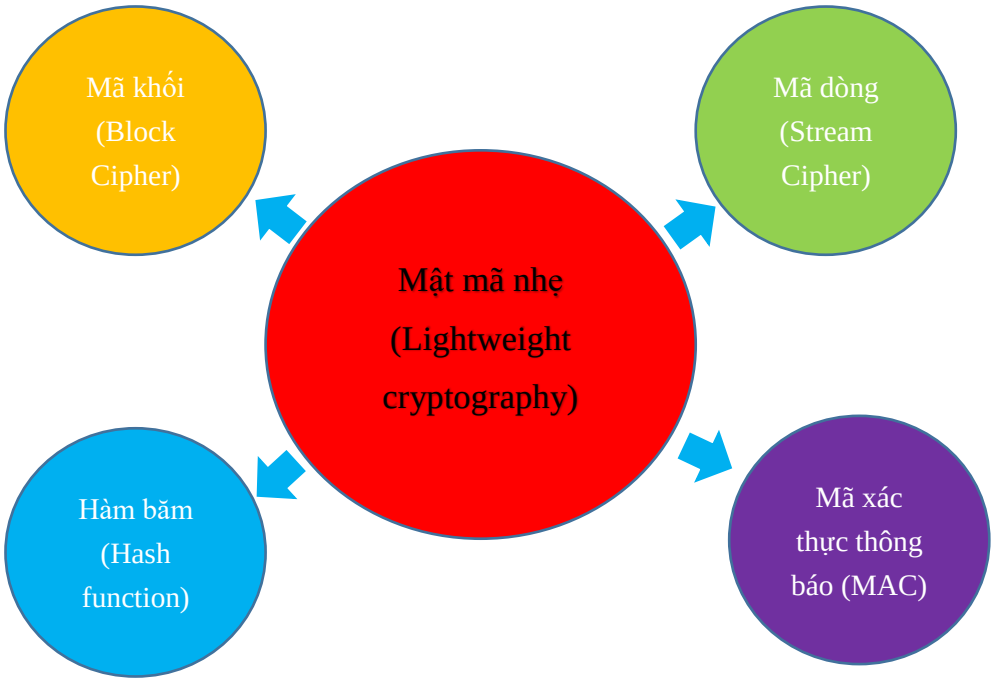


Hình 1-1: Ba nguyên lý thiết kế thuật toán mật mã nhẹ

Một hệ mật tốt nhất cần phải thỏa hiệp giữa giá thành, hiệu suất và độ an toàn. Với các mã khối, độ dài khóa là sự thỏa hiệp giữa độ an toàn và giá thành, trong đó, số vòng là sự cân bằng giữa hiệu suất và độ an toàn, như biểu diễn trên hình 1.3. Tuy nhiên, rất khó để có thể tối ưu hóa cả 3 khía cạnh trên.

1.1.3. Các nguyên thủy mật mã nhẹ

ECRYPT (European Network of Excellence for Cryptology) là một sáng kiến nghiên cứu về mật mã ở châu Âu, được bắt đầu vào năm 2004. ECRYPT đã giới thiệu 4 loại nguyên thủy mật mã nhẹ tương tự với mật mã truyền thống là mã khối, mã dòng, hàm băm và mã xác thực thông báo.



Hình 1-2: Các nguyên thủy mật mã nhẹ

1.1.4. Ứng dụng mật mã nhẹ trong IoT

Hầu hết các thuật toán mật mã nhẹ đều ra đời và phát triển cho từng yêu cầu cụ thể. Chúng phù hợp với những ứng dụng, thiết bị có cấu hình nhỏ gọn, tốc độ xử lý nhanh và nhiều trong một khoảng thời gian cố định, yêu cầu bảo mật không quá cao.

Với những thiết kế riêng của mình, mật mã nhẹ có những lợi ích đặc trưng:

- 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 rất gọn nhẹ trên những thiết bị có năng lực tính toán thấp.
- Giá thành rẻ. Mật mã nhẹ thường được ứng dụng trong những thiết bị có tính thâm nhập khắp nơi, dẫn đến việc triển khai hàng loạt, làm giảm giá thành của công nghệ được sử dụng.
- Hoạt động rất nhanh, thực hiện đầy đủ và hiệu quả công việc mà nó cần hoàn thành.

1.2. Mật mã dòng trong mật mã nhẹ

1.2.1. Khái niệm

Mật mã dòng là một kỹ thuật mã hóa thuộc loại mã đối xứng, trong đó dữ liệu đầu vào được mã hóa từng bit một. Có hai loại mật mã khóa đối xứng là mã dòng (stream cipher) và mã khối (block cipher). Trong đó như ta đã biết, mã khối sẽ làm việc bằng cách chia khối dữ liệu cần mã hóa ban đầu thành những khối dữ liệu nhất định, nghĩa là phải biết trước kích thước cũng như bản thân khối dữ liệu đó. 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, thậm chí biến thiên theo thời gian (time-varying). Mã dòng hoạt động với biến đổi của nó biến thiên theo thời gian trên những khối bản rõ (plaintext) riêng biệt.

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

Các kỹ thuật mã hóa dòng nhẹ nên được sử dụng trong IoT vì 2 lý do sau đây [14]:

- 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 thu 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.

Chương 2. MỘT SỐ HỆ MẬT MÃ DÒNG TRONG MẬT MÃ NHẸ PHỔ BIẾN HIỆN NAY

2.1. A5/1

A5/1 đã được sử dụng trong GSM suốt hơn 2 thập kỷ. Sự sáng tạo ra A5/1 đến nay vẫn còn được giữ bí mật. Tuy nhiên A5/1 đã được thiết kế lại bởi Briceno, Golberg, và Wagner vào năm 1999.

2.2. ChaCha

ChaCha là một biến thể của hệ mật mã dòng Salsa20[1] – 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.

2.3. 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.

2.4. 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 [15]. Tuy nhiên sau sự phá mã của Hell và Johansson [16], 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.

2.5. 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.

2.6. 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] 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

2.7. 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

2.8. 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.

2.9. 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.

2.10. 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.

Chương 3. HỌ GRAIN

3.1. Lịch sử

Grain là hệ mật mã dòng được đăng trên eSTREAM bởi Martin Hell, Thomas Johansson và Willi Meier năm 2004 với phiên bản đầu tiên Grain v0 [8]. Sau đó hệ mật này tiếp tục được phát triển thành Grain v1 [7] – là một trong bảy dự án được eSTREAM đưa vào các danh mục đầu tư từ 09/09/2008.

3.2. Grain V0

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). 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, f(x) được định nghĩa là: $f_0(x) = 1 + x^{18} + x^{29} + x^{42} + x^{57} + x^{67} + x^{80}$

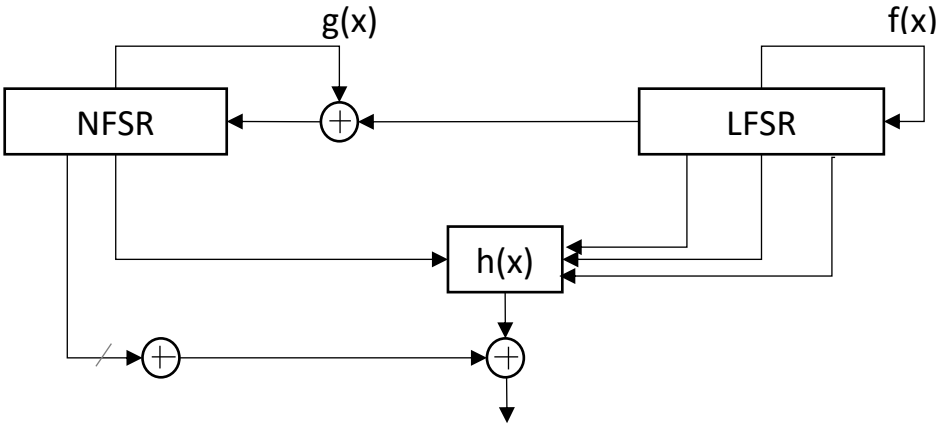
Phiên bản cập nhật của LFSR: $s_{i+80} = s_{i+62} + s_{i+51} + s_{i+38} + s_{i+23} + s_{i+13} + s_i$

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 3-1: 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

$$z^0{}_i = \sum_{k \in A} b_{i+k} + h_0(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63})$$

Trong đó $A = \{1, 2, 3, 10, 31, 43, 56\}$.

3.3. Grain V1

Tương tự như Grain v0, Grain v1 cũng sử dụng $k = 80$ và số bits của đầu ra là $l = 64$. Hàm LFSR $f_1(x)$ cũng tương tự như $f_0(x)$. Hàm NFSR $g_1(x)$ cũng tương tự như $g_0(x)$. Và bộ lọc $h_1(x)$ cũng tương tự như $h_0(x)$. Tuy nhiên các bit đầu ra cua Grain v1 được định nghĩa khác với Grain v0:

$$z^1{}_i = \sum_{i \in A_1} b_{k+i} + h_1(s_{i+3}, s_{i+25}, s_{i+46}, s_{i+64}, b_{i+63})$$

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

3.4. Grain 128

Thuật toán Grain-128 có đầu vào $k = 128$ và đầu ra $l = 96$. Hàm của LFSR được định nghĩa như sau: $f_{128}(x) = 1 + x^{32} + x^{47} + x^{58} + x^{90} + x^{121} + x^{128}$

Hàm của NFSR được định nghĩa như sau:

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

Bộ lọc: $h_{128}(x) = x_0x_1 + x_2x_3 + x_4x_5 + x_6x_7 + x_0x_4x_8$

Đầu ra:

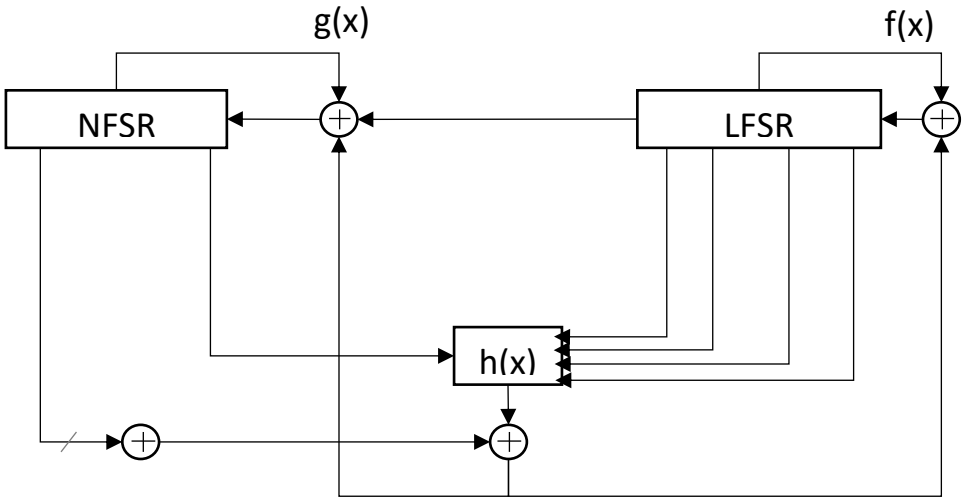
$$z^{128}{}_i = \sum_{i \in A_{128}} b_{k+i} + s_{93+i} + h_{128}(b_{i+12}, s_{i+8}, s_{i+13}, s_{i+20}, b_{i+95}, s_{i+42}, s_{i+60}, s_{i+79}, s_{i+95})$$

Trong đó $A_{128} = \{2, 15, 36, 45, 64, 72, 89\}$.

Tạo khóa

Để khởi tạo khóa, đầu tiên ta sử dụng NFSR với khóa $b_i = k_i, 0 \le i \le 79$, tiếp tục sử dụng 64 bits đầu tiên của LFSR với giá trị IV là $s_i = IV_i, 0 \le i \le 63$. Các bits còn ại của LFSR được xác định bởi $s_i = 1_i, 64 \le i \le 79$. Tiếp theo, thuật toán mã hóa được

thực hiện 160 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 3-2: Quá trình tạo khóa của Grain

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

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).

3.6. Điểm yếu

Cũng như những hệ mã hóa khác, họ Grain cũng chứa đựng những lỗ hổng nguy hiểm. Dưới đây là 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.

- Phương pháp tấn công tính toán giá trị Key-IV yếu
- Phương pháp tấn công khôi phục Key-IV
- Dynamic Cube Attacks

Chương 4. MÃ HÓA GRAIN TRÊN THIẾT BỊ RASPBERRY

4.1. Mô tả bài toán

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.

4.2. Giải quyết bài toán

Luận văn ứng dụng mã hóa đầu cuối với Encrypt-then-MAC 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 Kaccak nhằm đảm bảo tính xác thực cho thông điệp.

4.2.1. Mã hóa đầu cuối

Mã hóa đầu cuối (End-to-end encryption – E2EE) là phương pháp mã hóa cho phép người nhận biết được thông tin được gửi là gì, ngay cả các nhà cung cấp dịch vụ cũng không thể truy cập vào. Về nguyên tắc nó ngăn chặn những kẻ trộm tiềm năng – bao gồm cả nhà cung cấp dịch vụ viễn thông, các nhà cung cấp dịch vu Internet và thậm chí cả nhà cung cấp dịch vụ truyền thông – từ việc truy cập các khóa mật mã cần thiết đến giải mã cuộc hội thoại.

4.2.2. Mã xác thực thông báo

4.2.2.1. Khái niệm và mục đích sử dụng

Tính toàn vẹn dữ liệu

Tính toàn vẹn dữ liệu là dữ liệu hay thông tin không bị thay đổi, mất mát trong quá trình lưu trữ hay truyền tải dữ liệu. Tính toàn vẹn dữ liệu cho phép các bên liên quan xác minh rằng tin nhắn đã được xác thực

Xác thực thông điệp

Mã xác thực thông điệp 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

Mục đích sử dụng

Với các giao thức trực tuyến (online), mã xác thực thông báo mật mã (cryptographic Message Authentication Code – MAC) là rất quan trọng và có tính chất như bắt buộc để đảm bảo tính xác thực giữa các bên tham gia giao dịch.

4.2.3. Hàm băm Keccak

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:

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

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

Absorbing phase:

$\forall$ block P_i in P

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

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

Squeezing phase:

$Z = empty\ string$

while output is requested

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

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

return Z

$\}$

Đánh giá Keccak

Keccak có nhiều ưu điểm vượt trội so với các hàm băm khác:

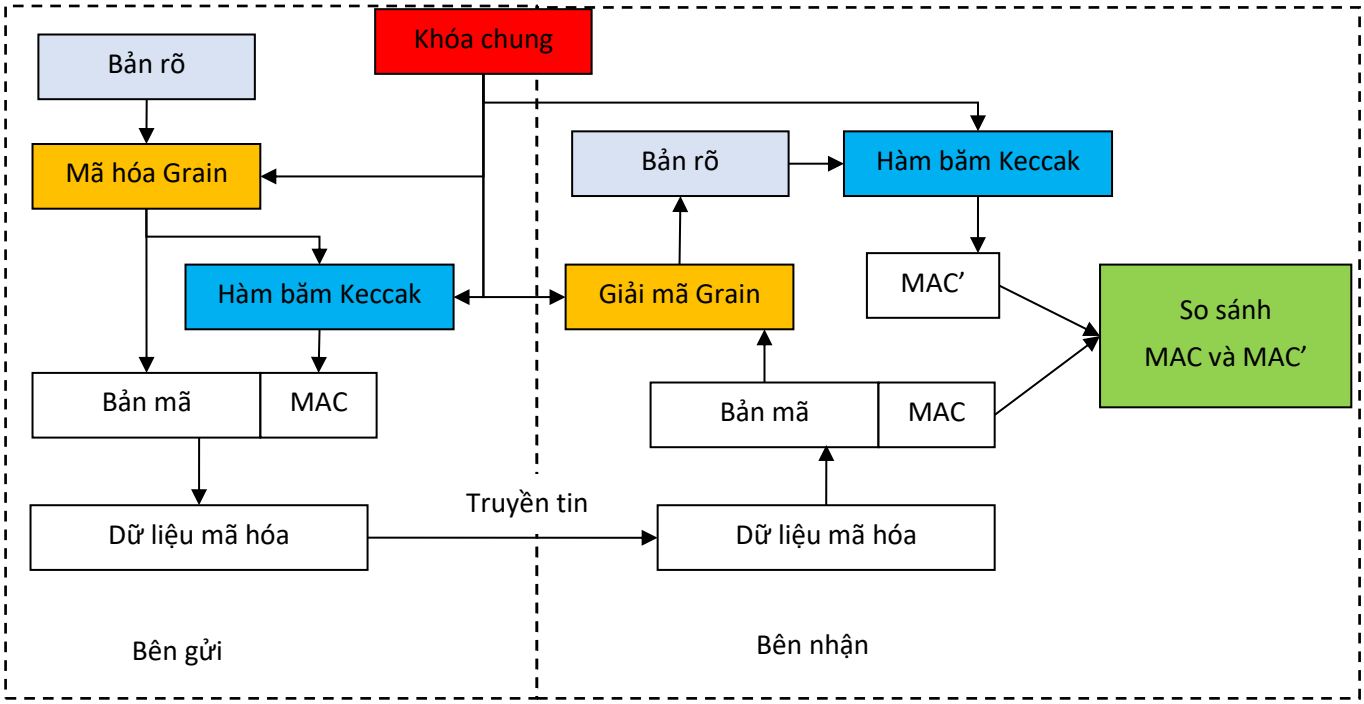
- Keccak có số vòng lặp là 18 vòng và kích thước trạng thái thay đổi từ 25, 50, 100, 200, 400, 800 đến 1600.
- Keccak có khả năng thực hiện trên cả 2 nền tảng 32 bits và 64 bits.

4.2.4. Tạo và trao đổi khóa

Trao đổi khóa Diffie–Hellman (D-H)

Giao thức này được công bố đầu tiên bởi Whitfield Diffie và Martin Hellman vào năm 1976. Áp dụng cho thiết bị Raspberry, để hệ thống đạt được sự an toán 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 được sinh ra một cách ngẫu nhiên. 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.

4.2.5. Mô hình mã hóa và xác thực



Hình 4-1: Mô hình mã hóa và xác thực

Mã hóa xác thực

Quá trình mã hõa xác thực diễn ra như sau:

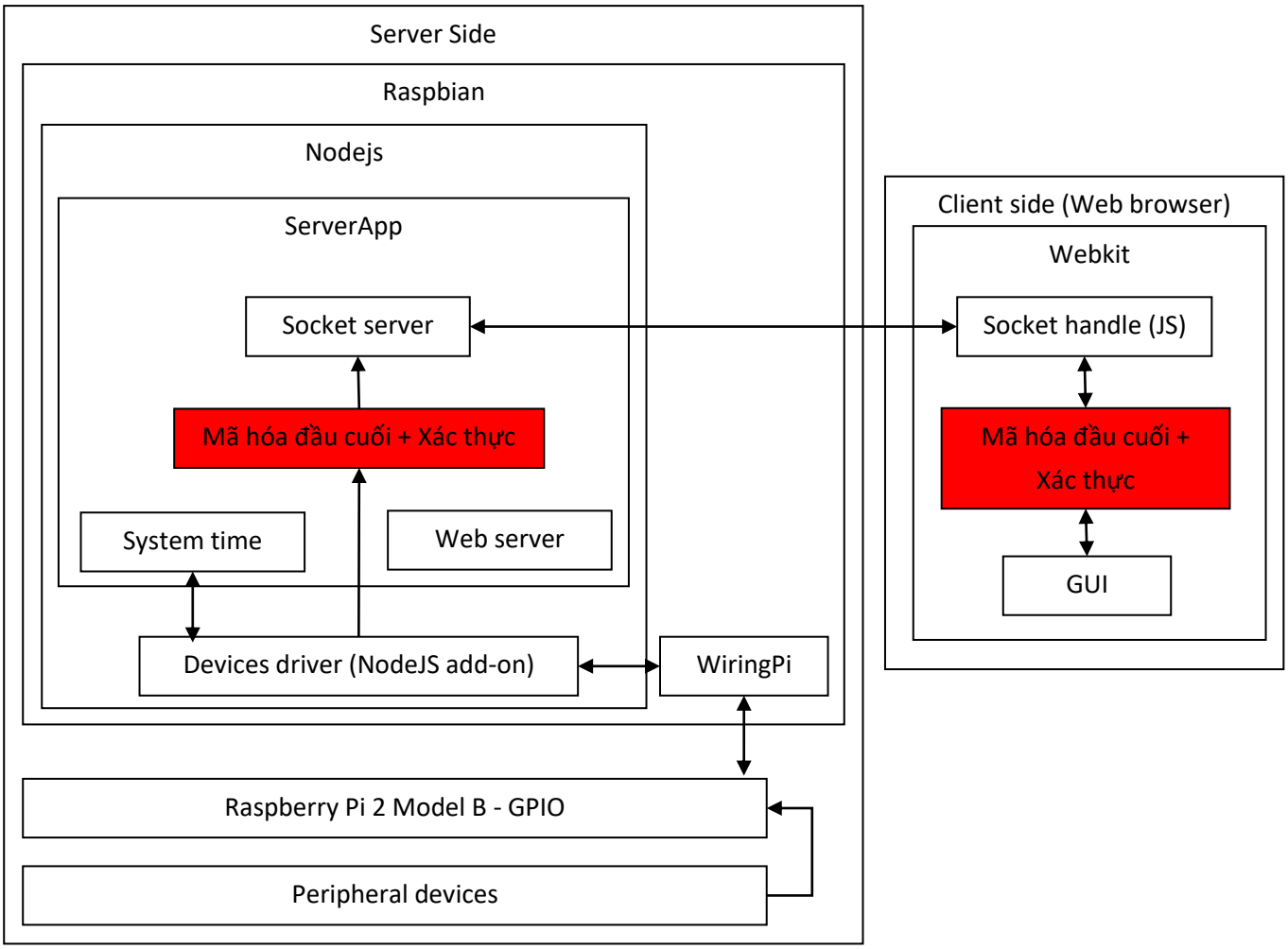
1. Thông điệp sẽ được mã hóa trước tiên dựa trên mã dòng Grain với K_e tạo ra bản mã C.
2. Tính toán giá trị M là MAC của bản mã bằng hàm băm Keccak (HMAC) với khóa K_m.
3. Gửi cả C và M trên kênh truyền tin

Giải mã xác thực

Quá trình giải mã xác thực diễn ra như sau:

1. Tách C và M. Vì sau thuật toán băm giá trị băm là cố định nên ta có thể tách riêng 2 phần trên.
2. Giải mã bản rõ P từ bản mã M.
3. Tính toán lại giá trị M' là MAC của bản mã 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.

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



Hình 4-2: 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 và thêm mã xác thực cho nó rồi gửi đến Server qua socket.io.
9. Server giải mã và xác thực thông điệp trên.

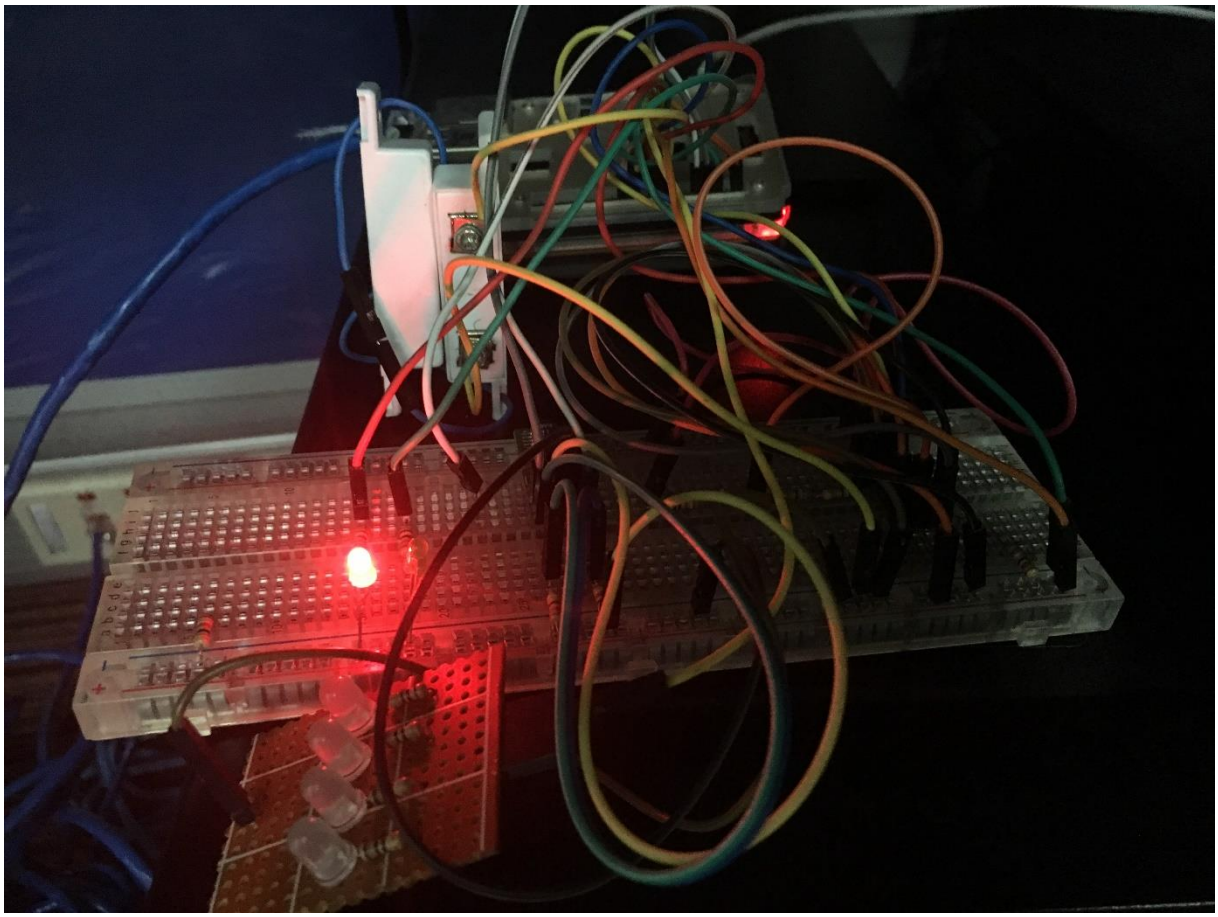
10. Server gửi tín hiệu điều khiển đến các devices kết nối đến nó. Trong luận văn này chỉ dừng lại ở việc mô phỏng việc gửi tín hiệu bằng các đèn LED.
    - Tăng nhiệt độ điều hòa hiển thị đèn LED đỏ.
    - Giảm nhiệt độ điều hòa hiển thị đèn LED xanh.
    - Mở cửa hiển thị đèn LED vàng.
    - Đóng cửa tắt đèn LED vàng.

4.3. Môi trường và dữ liệu thực nghiệm

- 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

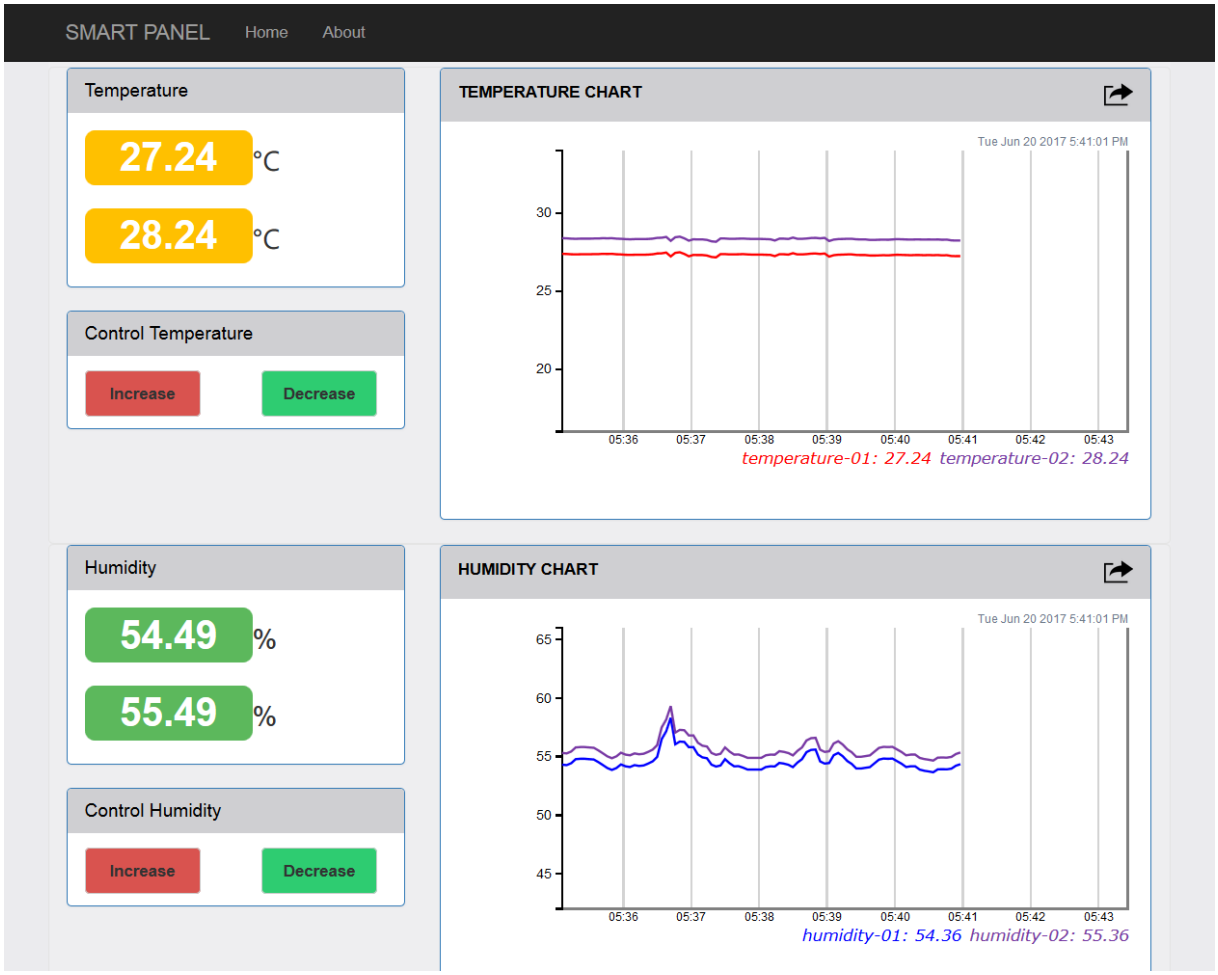
4.4. Kết quả thu được

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 4-3: Hình ảnh thực tế của Raspberry Pi cùng các cảm biến và đèn LED

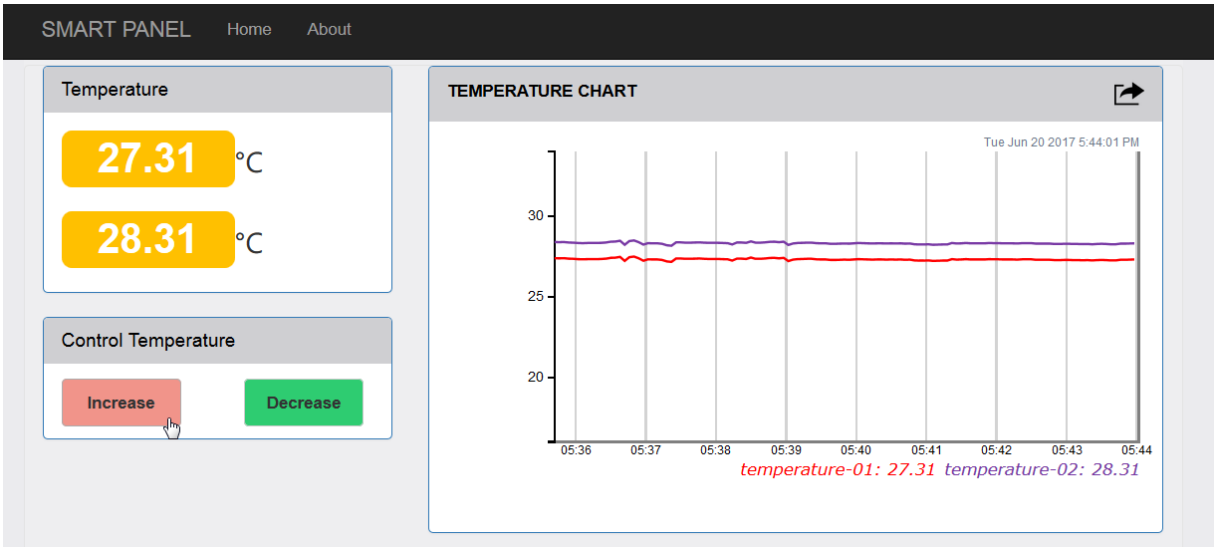
Giao diện chính của client



Hình 4-4: Giao diện chính của client

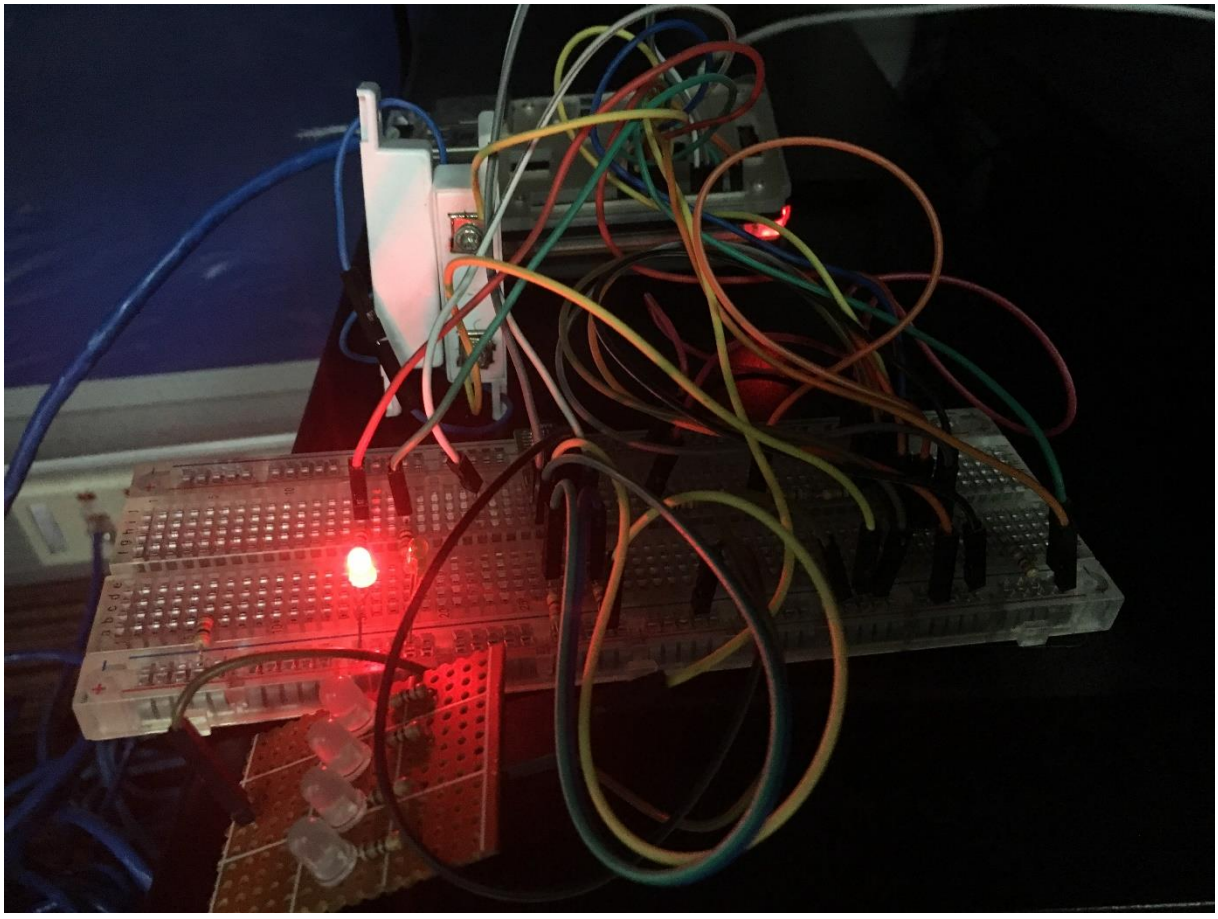
Khi nhận thấy nhiệt độ/độ ẩm thấp/cao người dùng có thể thực hiện lệnh điều khiển gửi đến các thiết bị để tăng nhiệt độ/độ ẩm. Hay khi muốn đóng/mở cửa người dùng cũng có thể gửi các lệnh điều khiển đến bộ phận điều khiển của cửa ra vào. Trong thiết bị giới hạn, luận văn chỉ mô phỏng các lệnh điều khiển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

- Tăng nhiệt độ



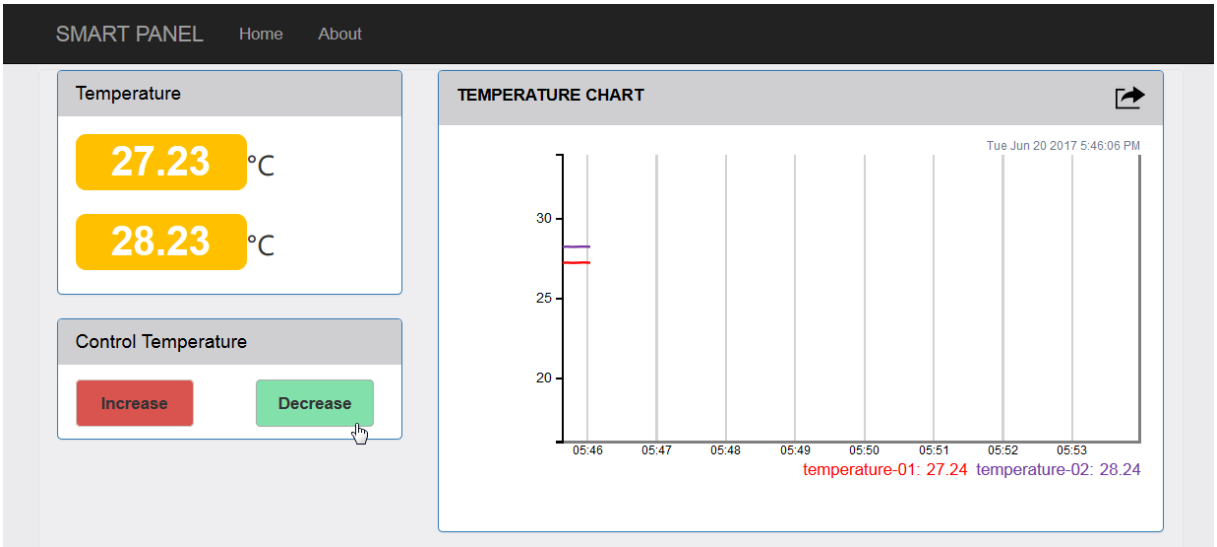
Hình 4-5: Màn hình tăng nhiệt độ

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



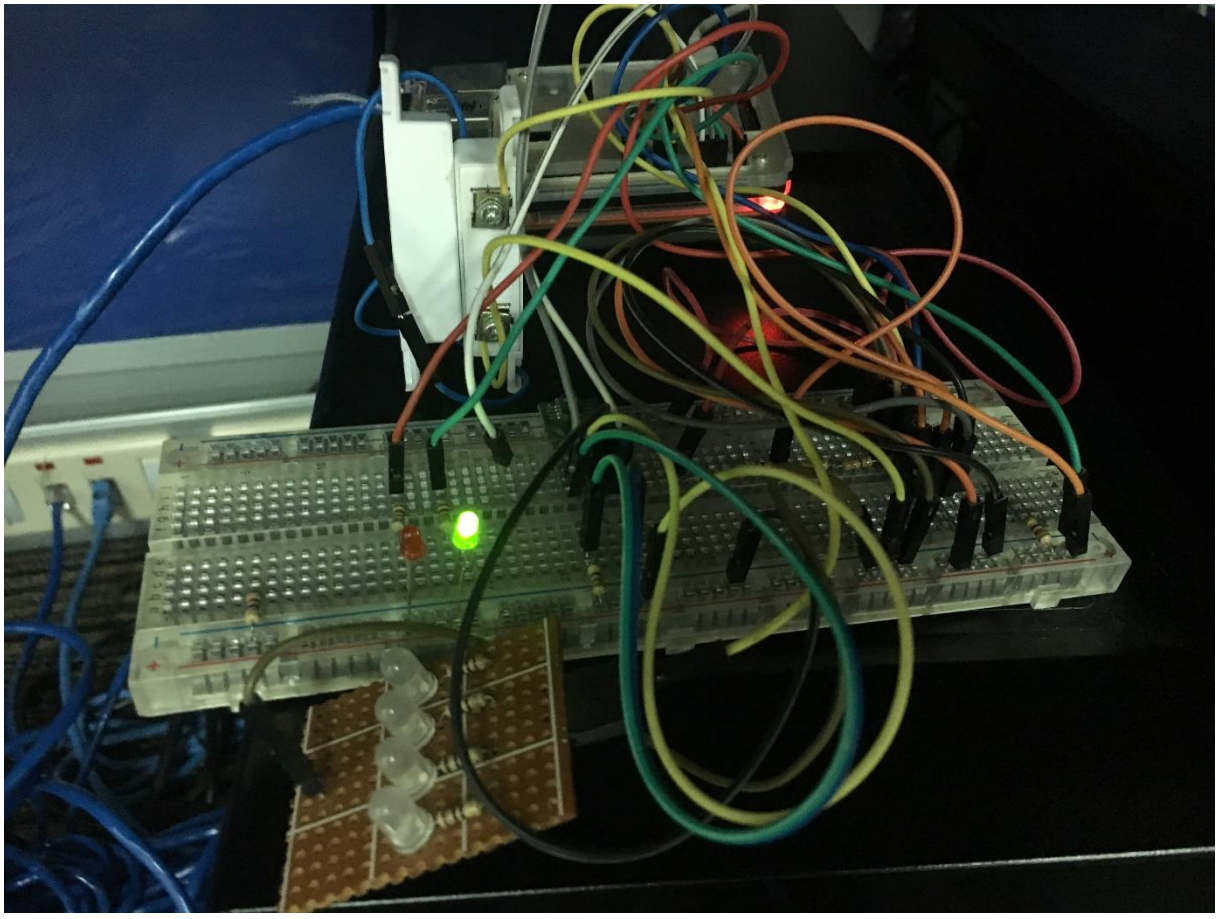
Hình 4-6: Giả lập Raspberry điều khiển tăng nhiệt độ qua đèn LED đỏ

- Giảm nhiệt độ



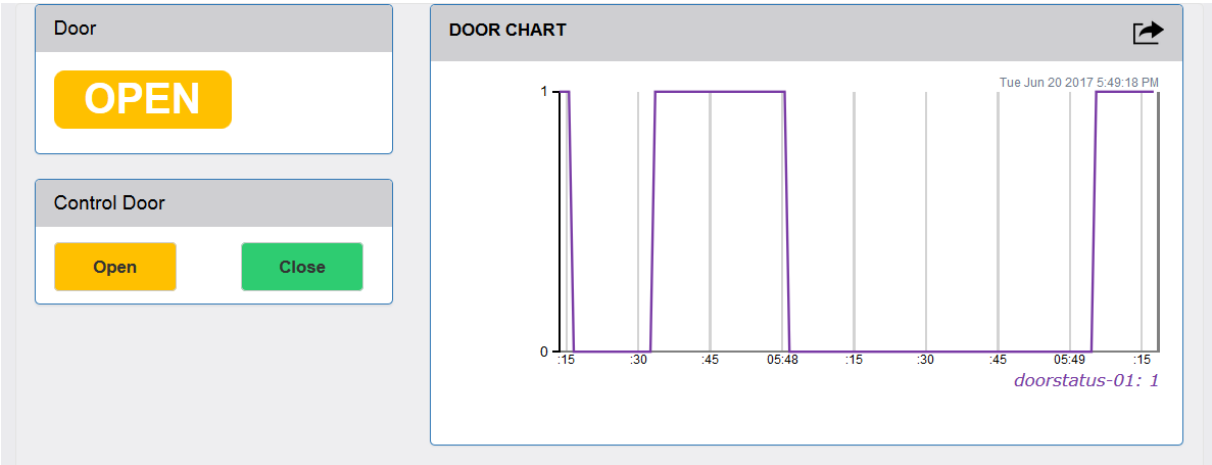
Hình 4-7: Màn hình giảm nhiệt độ

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



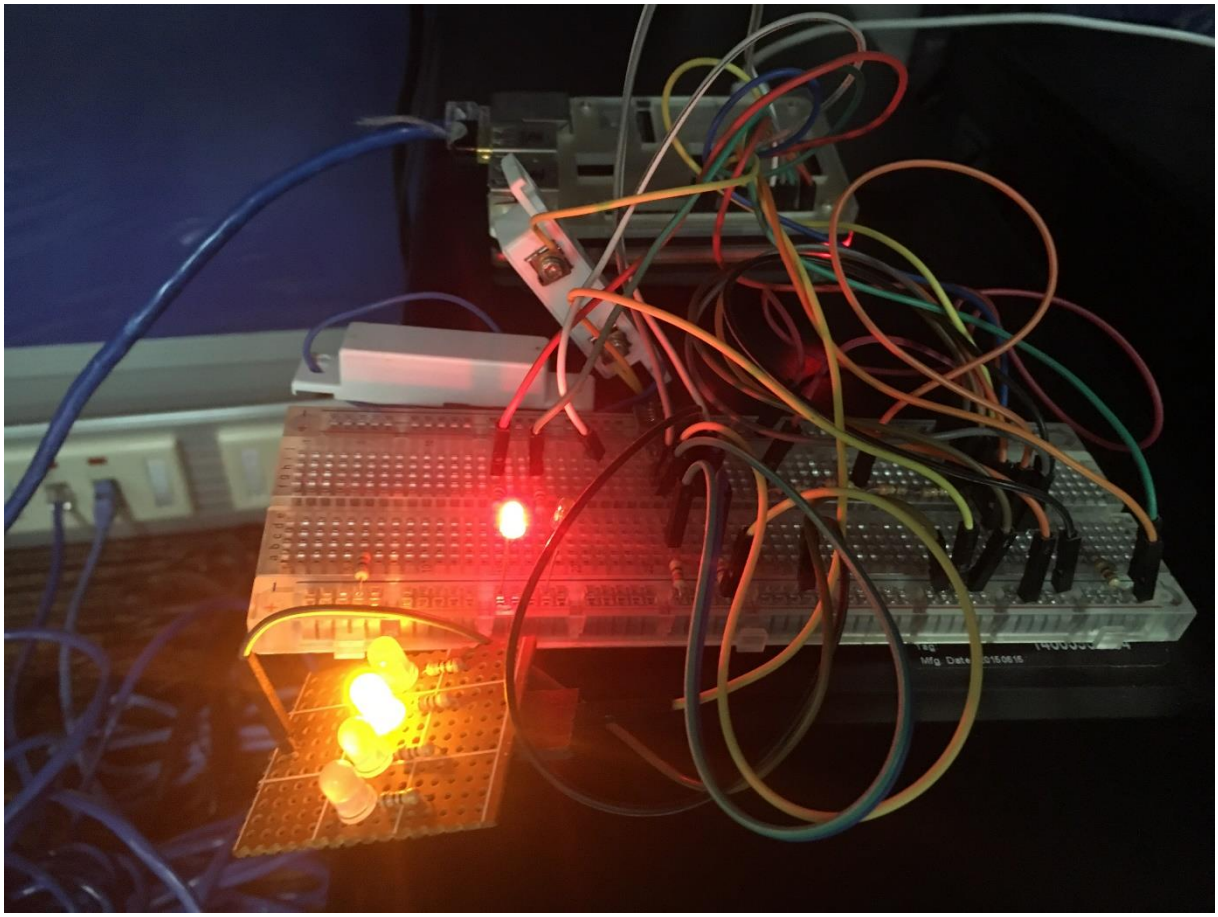
Hình 4-8: Giả lập Raspberry điều khiển giảm nhiệt độ qua đèn LED xanh

- Đóng cửa



Hình 4-9: Màn hình mở cửa

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



Hình 4-10: Giả lập Raspberry điều khiển mở cửa qua đèn LED vàng

4.5. Đánh giá

4.5.1. Đánh giá an toàn

- Kẻ tấn công đứng giữa (Man-in-the-Middle attacks. Với cách thức tấn công kẻ tấn công

đúng giữa, việc có được dữ liệu đã mã hóa cũng sẽ mất thời gian để giải mã. Điều này nhằm hạn chế được việc thông tin bị lộ.

- 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.
- Backdoors: Việc mã hóa sử dụng mã khối đối xứng hạn chế được việc nhà cung cấp dịch vụ có thể mở cửa hậu nhằm thu thập thông tin người dùng. Như vậy dữ liệu người dùng có thể được đảm bảo.

4.5.2. Đánh giá hiệu năng

Hình 4-11: Hiệu năng quá trình mã hóa và giải mã

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

Bảng 4-1: So sánh Grain và một số hệ mã hóa nhẹ khác trên Raspberry

Thuật toán	**Kích thước khóa**	**Kích thước IV**	**GE ***	**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	1362	101.2	59.3	125.5
Grain-128	128	96	1296	109.4	50.2	119.7
Trivium	80	80	2605	102.2	68.9	133.5
AES	256		2478	55.6	51.1	121.2
KATAN64	64		1056	25.2	86.7	157.4

** Giá trị trung bình sau 20 lần thực hiện mỗi thuật toán*

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

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 A5/1, 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. Grain phù hợp với các ứng dụng phần cứng, 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.

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

[2] 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.

[3] Adi Shamir, "Stream Ciphers: Dead or Alive?", ASIACRYPT, 2004, trang 22-41.

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

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

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

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

[8] 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.

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

[10] Côme Berbain, Henri Gilbert, Alexander Maximov (2006-01-02). "Cryptanalysis of Grain" (PDF). eSTREAM.

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

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

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

[14] Masanobu Katagi and Shiho Moriai, "Lightweight Cryptography for the Internet of Things"

[15] Arnault, F., Berger, T., Lauradoux, C., Minier, M., & Pousse, B. (2009, January). "A new approach for FCSRs. In Selected Areas in Cryptography" (trang. 433-448). Springer Berlin Heidelberg.

[16] Hell, M., & Johansson, T. (2008). “Breaking the F-FCSR-H stream cipher in real time”. In Advances in Cryptology-ASIACRYPT 2008 (trang. 557-569). Springer Berlin Heidelberg

[17] Biryukov, A., Shamir, A., & Wagner, D. (2001, January). Real time cryptanalysis of A5/1 on a PC. In Fast Software Encryption (pp. 1-18). Springer Berlin Heidelberg

[18] 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

[19] Tillich, S., et al.: “High-Speed Hardware Implementations of BLAKE, BMW, CubeHash, ECHO, Fugue, Grostl, Hamsi, JH, Keccak, Luffa, Shabal, SHAvite-3, SIMD, and Skein”. In: Cryptography ePrint (November 2009)

[20] Namin, A.H., Hasan, M.A.: “Hardware Implementation of the Compression Function for Selected SHA-3 Candidates”, CACR 2009-28 (July 2009)

[21] Elif Bilge Kavun and Tolga Yalcin, “A Lightweight Implementation of Keccak Hash Function for Radio-Frequency Identification Applications”

[22] Elie Bursztein. Google security blog: Speeding up and strengthening https connections for chrome on android (April 24, 2014), 2014. https://security.googleblog.com/2014/04/ speeding-up-and-strengthening-https.html.