DOI: 10.15625/vap.2017.00066

MỘT PHƯƠNG PHÁP CẢI TIẾN MẬT MÃ KHỐI ÁP DỤNG TRONG MẠNG ĐÒI HỎI THỜI GIAN XỬ LÝ NHANH

Nguyễn Đào Trường[1], Lê Mỹ Tú[1], Nguyễn Doãn Cường[2]

[1] Học viện Kỹ thuật Mật mã

[2] Viện Công nghệ thông tin, Viện KHCN Quân sự

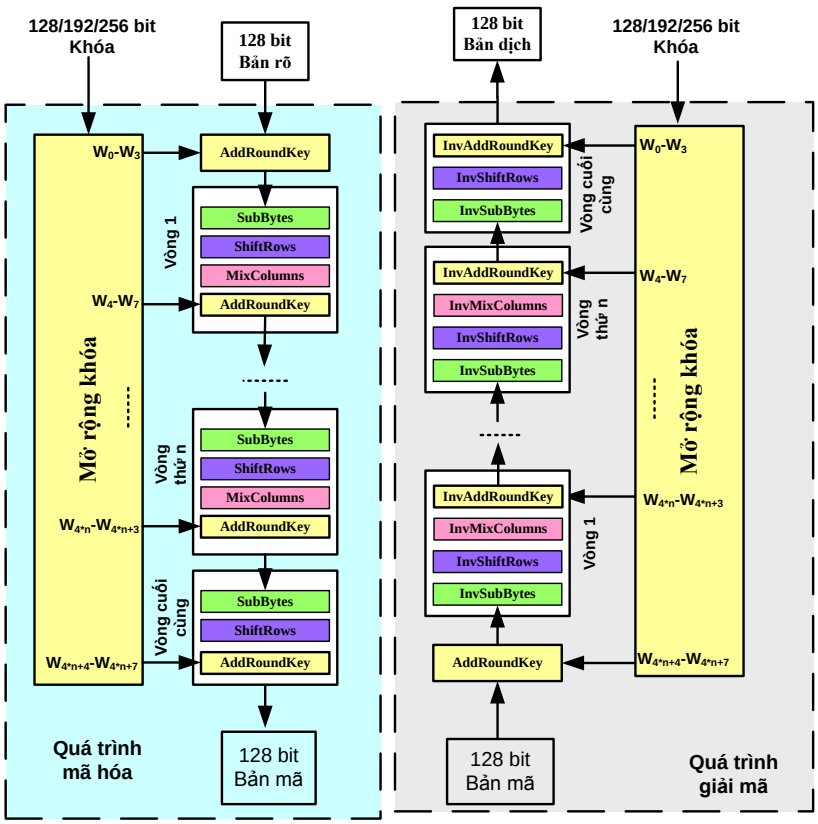
truongnguyendao@gmail.com, tulemy@hotmail.com, cuongvncntt@yahoo.com

TÓM TẮT: *Để bảo mật trong quá trình truyền dữ liệu trên mạng thì việc áp dụng giải pháp ứng dụng các kỹ thuật mã hóa là một điều tất yếu. Tuy nhiên, trong những mạng có tài nguyên hạn chế và yêu cầu thời gian thực hiện nhanh thì cần phải có những cải tiến phù hợp. Bài báo trình bày một phương pháp cải tiến mật mã khối để nâng cao khả năng bảo mật của hệ mật dựa trên sự phụ thuộc khóa của các hàm biến đổi vòng trong quá trình mã hóa và giải mã. Ngoài ra, bài báo cũng đề xuất giải pháp cứng hóa bằng hệ thống mô phỏng trên FPGA để tăng tốc độ mã hóa và giải mã áp dụng trong các mạng có tài nguyên hạn chế.*

Từ khóa: *FPGA (Field Programmable Gate Array), AES (Advanced Encryption Standard), mã hóa, giải mã.*

I. GIỚI THIỆU

AES là thuật toán chuẩn mật mã đối xứng được NIST (National Institute of Standard and Technology) thực hiện năm 2000, sau bốn năm nỗ lực để thay thế cho DES (Data Encryption Standard), NIST đã lựa chọn AES của Rijndael [1], [2] (NIST 2004). Bản FIPS (Federal Information Processing Standard) cho AES được công bố tháng 2 năm 2001 [3]; Việc chuẩn hóa AES được chấp nhận sau khi đã nhận xét và đánh giá công khai, được công bố thành chuẩn FIPS PUB-197 vào tháng 12 năm 2001 [3]. AES thực hiện mã hóa và giải mã theo khối sử dụng khóa đối xứng, độ dài của khóa có thể là 128, 192 hoặc 256 bit, tuy nhiên việc mã hóa và giải mã thực hiện theo các khối 128 bit [4]. Đầu vào của thuật toán mã hóa và giải mã là từng khối 128 bit, khối này được biểu diễn thành một ma trận vuông các byte. Khóa được cung cấp cho quá trình mã hóa và giải mã là qua quá trình mở rộng khóa lưu trữ trong một mảng các word, mỗi word có kích thước 4 byte và tổng cộng có 44, 52 hoặc 60 word tương ứng với độ dài khóa 128, 192, 256. Hình 1 mô tả chi tiết thuật toán mã hóa và giải mã của AES.



n là số lần lặp (9, 11, 13 tương ứng với 128, 192, 256

Hình 1. Mã hóa và giải mã trong AES

Trong quá trình mã hóa và giải mã sử dụng 4 hàm biến đổi khác nhau và các hàm ngược tương ứng sau đây:

A. SubBytes (invSubBytes)

Là hàm thay thế phi tuyến, sử dụng bảng thay thế (S-Box), bảng 1, bảng này được xây dựng bởi nghịch đảo phép nhân và biến đổi affine. Nó cung cấp khả năng xáo trộn và phi tuyến. Mỗi byte trong trạng thái (state) được thay thế bằng byte khác trong bảng S-Box.

B. ShiftRows (invShiftRows)

Là hàm dịch chuyển byte đơn giản, mỗi hàng được dịch chuyển tuần tự với số lượng bước cố định. Đặc biệt, các phần tử của hàng đầu tiên sẽ không thay đổi vị trí, hàng thứ hai dịch sang trái một cột, hàng thứ ba dịch sang trái hai cột, hàng cuối cùng dịch sang trái ba cột. Thao tác này nhằm đảm bảo mỗi cột của bảng đầu ra đều được tạo thành từ các cột của bảng trạng thái đầu vào.

Bảng 1. Bảng thế S-Box

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	63	7C	77	7B	F2	6B	6F	C5	30	01	67	2B	FE	D7	AB	76
1	CA	82	C9	7D	FA	59	47	F0	AD	D4	A2	AF	9C	A4	72	C0
2	B7	FD	93	26	36	3F	F7	CC	34	A5	E5	F1	71	D8	31	15
3	04	C7	23	C3	18	96	05	9A	07	12	80	E2	EB	27	B2	75
4	09	83	2C	1A	1B	6E	5A	A0	3B	52	D6	B3	29	E3	2F	84
5	53	D1	00	ED	20	FC	B1	5B	6A	CB	BE	39	4A	4C	58	CF
6	D0	EF	AA	FB	43	4D	33	85	45	F9	02	7F	50	3C	9F	A8
7	51	A3	40	8F	92	9D	38	F5	BC	B6	DA	21	10	FF	F3	D2
8	CD	0C	13	EC	5F	97	44	17	C4	A7	7E	3D	64	5D	19	73
9	60	81	4F	DC	22	2A	90	88	46	EE	B8	14	DE	5E	0B	DB
A	E0	32	3A	0A	49	06	24	5C	C2	D3	AC	62	91	95	E4	79
B	E7	C8	37	6D	8D	D5	4E	A9	6C	56	F4	EA	65	7A	AE	08
C	BA	78	25	2E	1C	A6	B4	C6	E8	DD	74	1F	4B	BD	8B	8A
D	70	3E	B5	66	48	03	F6	0E	61	35	57	B9	86	C1	1D	9E
E	E1	F8	98	11	69	D9	8E	94	9B	1E	87	E9	CE	55	28	DF
F	8C	A1	89	0D	BF	E6	42	68	41	99	2D	0F	B0	54	BB	16

C. MixColumns (invMixColumns)

Mỗi cột được chuyển đổi tuyến tính bằng cách nhân nó với một ma trận trong trường hữu hạn. Chính xác hơn mỗi cột được xem như một đa thức trong trường $GF(2^8)$ và được nhân modulo x^4+1 với một biểu thức cố định $c(x)=3x^3+x^2+x+2$. Phép nghịch đảo trong biến đổi tuyến tính cùng với hàm *ShiftRow*, để tạo ra các bản tin trong quá trình mã hoá và hàm ngược của nó trong quá trình giải mã.

D. AddRoundKey (invAddRoundKey)

Mỗi byte trong bảng trạng thái được thực hiện phép *XOR* với một khoá vòng, quá trình xử lý AES thu được 11/13/15 khoá vòng từ các khóa mã hoá được phân phối cho quá trình mã hoá và giải mã.

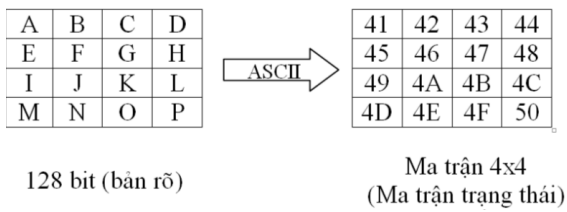
Cả quá trình mã hóa và giải mã đều bắt đầu bằng giai đoạn *AddRoundKey*, tiếp theo là các vòng với mỗi vòng bao gồm cả bốn giai đoạn trên, vòng cuối cùng chỉ còn ba giai đoạn (Hình 1).

II. ĐỀ XUẤT CẢI TIẾN AES

Những thay đổi trong thuật toán AES thể hiện như sau: khóa mã phân bố trong một ma trận 4×4, như thể hiện trong hình 2. Ở đây khóa bí mật có độ dài 128 bit còn với khóa 192 và 256 thì AES cải tiến chỉ lấy 128 bit đầu tiên để thực hiện trong các cải tiến, $k_{i,j}=k_{i,j}^0k_{i,j}^1...k_{i,j}^k$ là phần tử khóa có độ dài 8 bit (*i* là chỉ số hàng, *j* là chỉ số cột, *k* là vị trí các bit trong chuỗi biểu diễn dạng nhị phân của phần tử khóa ở ô (*i, j*)), $0\le i\le 3; 0\le j\le 3; 0\le k\le 7$. Mỗi hàng ma trận này được dùng cho các phép biến đổi khác nhau trong quá trình cải tiến. Hai hàng cuối cùng của ma trận khóa mã được sử dụng trong SubBytes cải tiến. Hàng đầu tiên được sử dụng trong ShiftRows cải tiến. Hàng thứ ba được sử dụng trong MixColumns cải tiến. Bước AddRoundKey giữ nguyên không thay đổi. Bản rõ được phân bố trong một ma trận 4×4 được gọi là ma trận trạng thái State (M) (hình 3).

$$K\Rightarrow\begin{bmatrix} k_{0,0} & k_{0,1} & k_{0,2} & k_{0,3} \\ k_{1,0} & k_{1,1} & k_{1,2} & k_{1,3} \\ k_{2,0} & k_{2,1} & k_{2,2} & k_{2,3} \\ k_{3,0} & k_{3,1} & k_{3,2} & k_{3,3} \end{bmatrix}$$

Hình 2. Phân bổ khóa trong AES cải tiến



Hình 3. Ma trận State trong AES cải tiến

A. Quá trình mã hóa

1. Cải tiến hàm SubBytes

Chuyển hàng thứ 3 của ma trận khóa về dạng nhị phân, bốn nhóm nhị phân 8 bit, 4 bit thứ hai từ mỗi 8 bit được tách ra và tham gia vào việc lựa chọn phần tử thay thế trong bảng S-Box, $g_0 = k_{2,0}^4 k_{2,0}^5, g_1 = k_{2,0}^6 k_{2,0}^7, g_2 = k_{2,1}^4 k_{2,1}^5, g_3 = k_{2,1}^6 k_{2,1}^7,$ $g_4 = k_{2,2}^4 k_{2,2}^5$. Những bit này được nhóm thành 4 nhóm: g_0g_4, g_1g_5, g_2g_6, g_3g_7 trong đó g_0, g_1, g_2 và g_3 lần lượt là số hàng và g_4, g_5, g_6 và g_7 lần lượt là số cột của ma trận trạng thái. Dữ liệu ở vị trí M(g_0, g_4) được thay thế từ S-box. Phép thay thế được thực hiện tương tự như hàm SubBytes của AES. Thực hiện tương tự với các vị trí còn lại M(g_1, g_5), M(g_2, g_6) và M(g_3,g_7). Như vậy, hàng thứ 3 của ma trận mã gồm 4 phần tử được thay thế trong ma trận trạng thái. Tương tự như vậy, quá trình trên sẽ được lặp lại với hàng thứ 4 của ma trận khóa.

2. Cải tiến hàm ShiftRows

Trong bước này, hàng thứ nhất của ma trận khóa được chuyển đổi sang dạng nhị phân tương ứng. Chuỗi nhị phân thu được nhóm thành 8 nhóm, mỗi nhóm có 4 bit bắt đầu từ bit đầu tiên, $a_0 = k_{0,0}^0 k_{0,0}^1 k_{0,0}^2 k_{0,0}^3, a_1 = k_{0,0}^4 k_{0,0}^5 k_{0,0}^6 k_{0,0}^7,$ $a_2 = k_{0,1}^0 k_{0,1}^1 k_{0,1}^2 k_{0,1}^3, a_3 = k_{0,1}^4 k_{0,1}^5 k_{0,1}^6 k_{0,1}^7, a_4 = k_{0,2}^0 k_{0,2}^1 k_{0,2}^2 k_{0,2}^3, a_5 = k_{0,2}^4 k_{0,2}^5 k_{0,2}^6 k_{0,2}^7, a_6 = k_{0,3}^0 k_{0,3}^1 k_{0,3}^2 k_{0,3}^3, a_7 = k_{0,3}^4 k_{0,3}^5 k_{0,3}^6 k_{0,3}^7.$ Các bit của a_0 được XOR với những bit của a_7 để có được một kết quả 4 bit nhị phân của P, $P = a_0 \oplus a_7$. Tương tự như vậy Q, R và S được tạo ra từ các nhóm còn lại [a_1, a_6], [a_2, a_5] và [a_3, a_4] tương ứng, $Q = a_1 \oplus a_6$, $R = a_2 \oplus a_5$, $S = a_3 \oplus a_4$. Các bit của P và R được sử dụng để xác định chỉ số hàng còn các bit của Q và S xác định số lần dịch vòng trái. Hai bit đầu tiên của P cho biết chỉ số hàng được dịch vòng trái. Với hàng số các số 1 trong Q lớn hơn 1. Việc làm này cho phép số lần dịch bit đối với hàng đó được biểu diễn bởi P. Quá trình tương tự được lặp lại cho R. Ở đây, nếu hai bit đầu tiên (tạm gọi b_0 và b_1) biểu diễn cùng một hàng mà đã được dịch trước đó thì b_1 và b_2 được coi là những bit kế tiếp ngay sau. Mặt khác, nếu số hàng giống nhau thì b_2 và b_3 được kiểm tra kế tiếp ngay sau b_3 và b_1. Sau khi kiểm tra tất cả các bit, nếu số hàng giống nhau thì hàng giống nhau đó được dịch đi số các số 1 trong S lớn hơn 1. Vì vậy, số hàng tối đa được dịch là 2.

Thực hiện đối xứng trên các hàng còn lại. Trong thao tác này, dữ liệu riêng được chuyển từ mỗi hàng thành dạng nhị phân và đọc từ phải sang trái, cuối cùng chuyển nó sang dạng thập lục phân. Ví dụ: Xét các phần tử dữ liệu 4A (Hexa) = 0100 1010 (nhị phân). Đọc từ phải sang trái, ta có được 0101 0010 = 52 (Hexa).

3. Cải tiến hàm MixColumns

Ở vòng này, các phần tử của hàng thứ 2 trong ma trận khóa được tách ra thành các khối 4 bit, $b_0 = k_{1,0}^0 k_{1,0}^1 k_{1,0}^2 k_{1,0}^3,$ $b_1 = k_{1,0}^4 k_{1,0}^5 k_{1,0}^6 k_{1,0}^7, b_2 = k_{1,1}^0 k_{1,1}^1 k_{1,1}^2 k_{1,1}^3, b_3 = k_{1,1}^4 k_{1,1}^5 k_{1,1}^6 k_{1,1}^7, b_4 = k_{1,2}^0 k_{1,2}^1 k_{1,2}^2 k_{1,2}^3, b_5 = k_{1,2}^4 k_{1,2}^5 k_{1,2}^6 k_{1,2}^7, b_6 = k_{1,3}^0 k_{1,3}^1 k_{1,3}^2 k_{1,3}^3, b_7 = k_{1,3}^4 k_{1,3}^5 k_{1,3}^6 k_{1,3}^7.$ Sau đó được nhóm lại, $\overline{b_0b_7}, \overline{b_1b_6}, \overline{b_2b_5}, \overline{b_3b_4},$ những nhóm này sau đó chuyển đổi thành giá trị thập phân tương ứng và tính mod với 4 cho mỗi nhóm (ví dụ $\overline{b_0b_7} = 01010010 \Leftrightarrow 82_{10}, r = 82\%4 = 2$ tức là cột thứ ba). Phần dư r sẽ luôn luôn nằm trong phạm vi từ 0 đến 3 (*tức là* $0 \le r \le 3$. *0, 1, 2 và 3*) tương ứng cột đầu tiên, cột thứ hai, cột thứ ba và cột thứ tư. Các biến đổi *MixColumns* của thuật toán ban đầu được thực hiện trên các cột đã chọn. Số cột tối đa được trộn trong bước này là 4.

4. Cải tiến hàm mở rộng khóa

Trong thuật toán AES cải tiến này, sử dụng thuật toán mở rộng khóa dựa trên hàm băm. Trước tiên, tiến hành tính mã băm với khóa được sử dụng trong thuật toán AES. Sau đó mã băm này kết hợp với khóa được nén sao cho có độ dài 16 byte (128 bit). Sử dụng hàm băm SHA-1 để tạo ra mã băm có 128 bit, làm đầu vào cho quá trình mở rộng khóa. Nó sẽ tạo ra thuật toán mã hóa mạnh trong thuật toán AES cải tiến. Tiếp theo giá trị hàm băm 16 byte khóa chuyển thành các word. Sử dụng các hàm sau đây, các word liên tiếp được tạo ra cho addroundkey được sử dụng trong mỗi vòng mã hóa và giải mã.

RotWord thực hiện dịch vòng trái một byte trên một word.

SubBytes thực hiện thay thế một byte trên mỗi byte của word đầu vào sử dụng *S-Box*.

Kết quả của các bước trên được XOR với một hằng số vòng, *Rcon*[j].

B. Quá trình giải mã

Quá trình giải mã diễn ra ngược lại so với quá trình mã hóa. Khi bên nhận, nhận được dữ liệu đã mã hóa, đầu tiên sẽ XOR dữ liệu bố trí trong ma trận 4×4 với ma trận khóa. Kết quả thu được là đối tượng của các vòng biến đổi sau đây:

1. Inverse MixColumns

Quá trình này làm việc giống như MixColumns cải tiến nhưng quá trình thực hiện sẽ ngược lại. Hàng thứ 2 của ma trận khóa sẽ được thực hiện trong bước này. Các thao tác được thực hiện trên ma trận dữ liệu theo thứ tự ngược lại với quá trình mã hóa.

2. Inverse ShiftRows

Ở vòng này, các bit của hàng đầu tiên trong ma trận khóa sẽ nhóm lại như trong quá trình của ShiftRows. Cần lưu ý rằng các bit được nhóm cũng tương tư như trong phần mã hóa. Trong vòng này hàng được biểu diễn bằng bit trong *R* và hàng đó được biểu diễn bởi các bit trong *P* được dịch vòng sang phải tương ứng. Các hàng còn lại được thực hiện đối xứng ngẫu nhiên theo thứ tự bất kỳ.

3. Inverse SubBytes

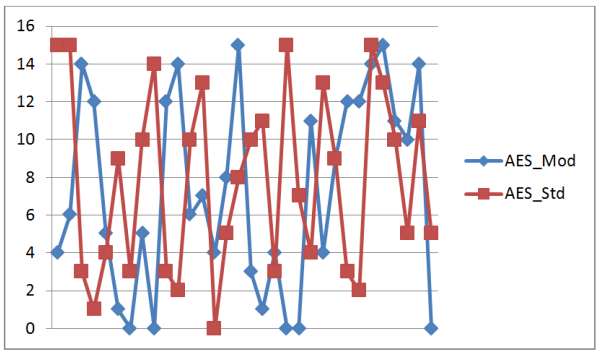
Tại vòng này, sử dụng hai hàng cuối cùng trong ma trận khóa. Các biến đổi được thực hiện trong vòng SubBytes vẫn không thay đổi. Thứ tự thay thế được lấy từ trong ra ngoài có nghĩa là trục thay thế theo thứ tự M(g_3, g_4), M(g_2,g_5), M(g_1,g_6) và M(g_0,g_7). Sử dụng S-Box nghịch đảo cho phép thay thế dữ liệu trong ma trận này.

C. Thử nghiệm và đánh giá AES cải tiến

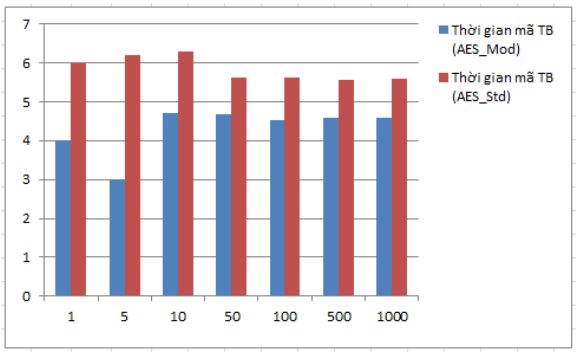
1. Kết quả mã hóa AES cải tiến

Sau khi chạy thử nghiệm mã hóa với AES cải tiến, một số kết quả đánh giá như trong các hình 4.a. Trong thử nghiệm, sử dụng khóa mã (128 bit) và bản rõ đơn giản dạng Hex như sau Key="EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEE", bản rõ Plaintext = "11111111111111111111111111111111". Ngay cả khi bản rõ và khóa mã rất đơn giản thì những cải tiến giúp làm cho bản mã có độ phi tuyến tốt. Điều này càng thể hiện rõ hơn trong phần thay đổi một hoặc hai bit ở một số vị trí nhưng kết quả cho bản mã có những chỉ số về độ bảo mật cao.

Về thời gian thực hiện mã hóa với số lượng các khối 128 bit cũng được cải thiện hơn (hình 4.b), tuy nhiên không đáng kể (AES cải tiến là 4,2979 ms so với 5,8399 ms của AES chuẩn). Để tăng tốc độ mã hóa và giải mã, trong phần III của bài báo sẽ đề xuất giải pháp cứng hóa thuật toán bằng công cụ mô phỏng FPGA.



a. Phân bố bản mã của AES cải tiến (AES_Mod) và AES chuẩn (AES_Std)



b. Thời gian mã hóa trung bình với số block (128 bit) khác nhau

Hình 4. Kết quả thử nghiệm mã hóa với AES cải tiến

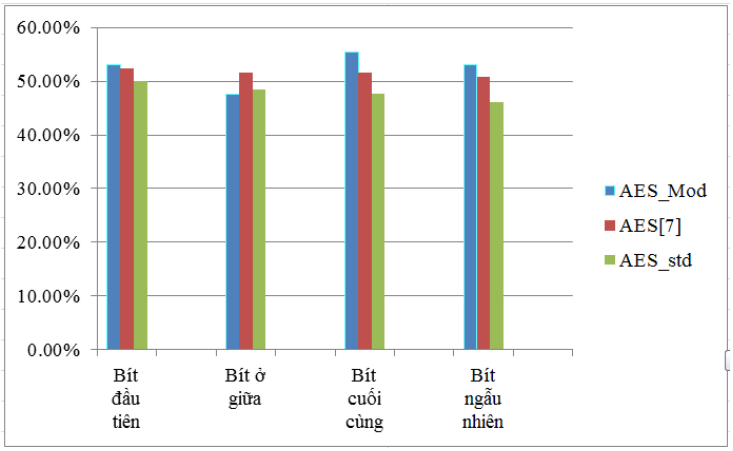
2. Đánh giá an toàn của AES cải tiến

Một tham số quan trọng trong vấn đề đánh giá mức độ bảo mật của một hệ mật mã khối đó là hiệu ứng thác đổ (avalanche effect) [5], đây là đặc điểm quan trọng của các thuật toán mã hóa. Năm 2000, các tác giả P. Barreto và V. Rijmen [6] đã đưa ra tiêu chuẩn về mức độ ảnh hưởng hiệu ứng thác đổ để đánh giá mã khối. Phương pháp này kiểm tra mức độ của tiêu chuẩn hiệu ứng thác đổ AVE, trong đó chỉ ra hàm f: $(GF(2))^n \rightarrow (GF(2))^m$ được gọi là thỏa mãn tiêu chuẩn hiệu ứng thác đổ nếu trung bình có 1/2 các bit ra thay đổi mỗi khi thay đổi một bit đơn đầu vào. Trong mật mã, hiệu ứng thác đổ thể hiện khi chỉ cần thay đổi một chút (một bit đơn ở một vài vị trí) dẫn tới những thay đổi đáng kể trong bản mã (có thể là một nửa số bit trong phần mã). Để xem xét chất lượng của các khối mã, khi thay đổi một chút ở bản mã hoặc ở khóa dẫn đến sự thay đổi lớn trong bản mã. Việc xây dựng một bộ mã cho thấy được sự lan tỏa đáng kể là một mục tiêu quan trọng trong bộ mã. Hiệu ứng thác đổ được tính như sau:

$$AVE = \frac{ChangedNo}{Length} x100\%$$

Trong đó *AVE* là hiệu ứng thác đổ, *ChangedNo* là số bit thay đổi trong bản mã, *Length* là độ dài bản mã theo bit.

Trong thử nghiệm, xét bản rõ và khóa mã có độ dài 128 bit. Với những thay đổi trong khóa mã từ "12" thành "02", "01" thành "11" và "F0" thành "F1" thì kết quả thu được đều có tỷ lệ thay đổi các bit trong bản mã ở chủ yếu trên 50 %, còn AES gốc đạt dưới 50 %. Ngoài ra, so với cải tiến của các tác giả Amish Kumar, Mrs. Namita Tiwari [7] thì những cải tiến của chúng tôi về cơ bản đều cao hơn, chi tiết trong hình 5.



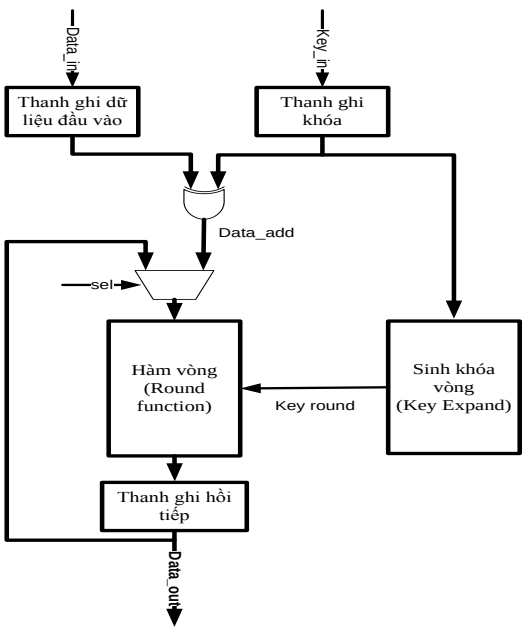
Hình 5. Hiệu ứng thác đổ khi thay đổi các bit đơn

III. ĐỀ XUẤT CỨNG HÓA AES CẢI TIẾN

Không mất tính tổng quát, trong mô hình đề xuất cứng hóa này, chúng tôi triển khai cho AES-128 (128 bit khóa đầu vào). Như vậy, AES sẽ được thực hiện trong 10 vòng lặp.

A. Mô hình thiết kế cứng hóa mã khối AES cải tiến

Thuật toán AES được thực hiện trên 10 vòng lặp, khi thực hiện cứng hóa thuật toán trên phần cứng, thuật toán được thiết kế theo mô hình rút gọn như trong hình 6. Thuật toán mã khối AES thực hiện trên 10 vòng lặp. Dựa trên nguyên tắc này, mô hình rút gọn chỉ sử dụng tài nguyên thực hiện đủ cho một vòng xử lý, sau đó kết quả đầu ra lại được đưa trở lại thành đầu vào của vòng tiếp theo. Như vậy với mô hình này tài nguyên sử dụng để cứng hóa thuật toán được rút gọn đi tối đa.



Hình 6. Mô hình thiết kế thuật toán AES cải tiến rút gọn

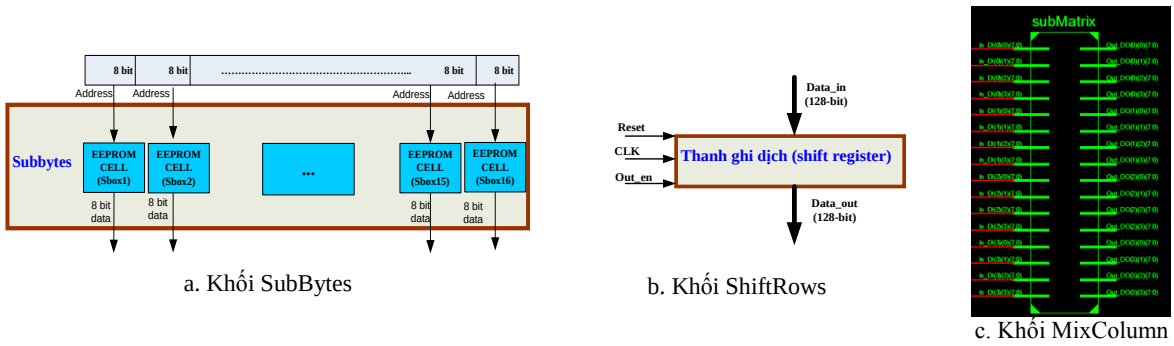
B. Thiết kế các module cơ bản trong AES cải tiến

1. Bảng thế S-Box và SubBytes

Các giá trị (8 bit) của bảng thế sbox1…sbox16 đã được nạp sẵn vào trong ROM của FPGA, quá trình thế là việc áp địa chỉ đầu vào tương ứng với giá trị của bảng thế đầu ra như thể hiện trong hình 7.a.

2. Biến đổi ShiftRows

Phép dịch vòng 8, 16, 24 bit được thực hiện bằng phương pháp dịch vòng thanh ghi dữ liệu đầu vào đi 8, 16, 24 vị trí. Dữ liệu sau khi đã dịch được chuyển sang thanh ghi kết quả tầng tiếp theo như hình 7.b.



a. Khối SubBytes b. Khối ShiftRows c. Khối MixColumn

Hình 7. Các khối cứng khóa các hàm con trong AES cải tiến

3. Phép biến đổi Mixcolumn

Thực hiện phép nhân ma trận dữ liệu với ma trận nhân. Phép Mixcolumn thực hiện nhờ khối dịch vào và XOR với x "1B" trong FPGA như trong hình 7.c.

4. Khối hàm vòng

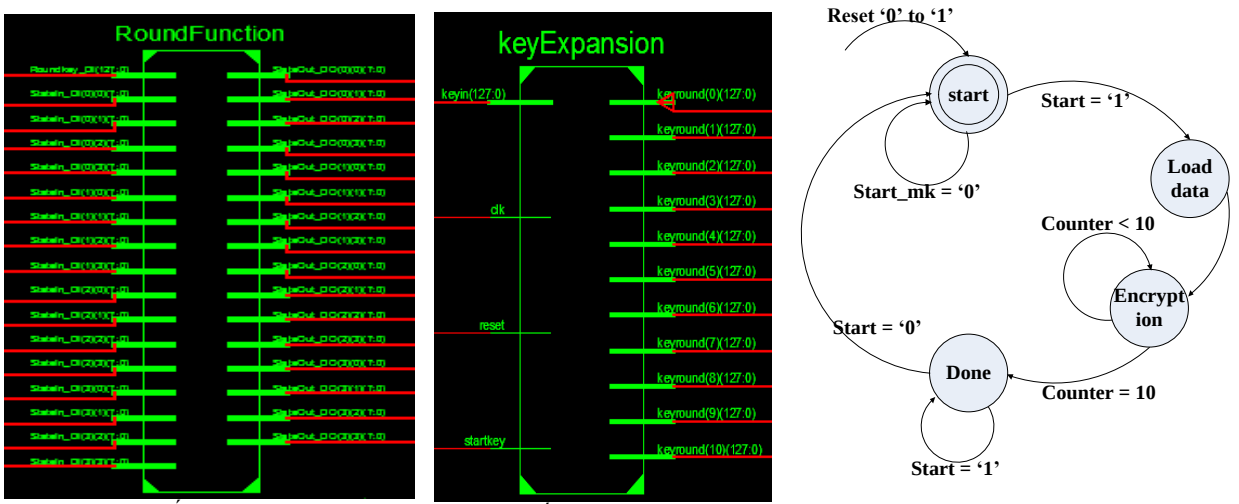
Khối hàm vòng thực hiện các phép tính toán Subbytes, ShiftRows và Mixcolumn. Đây là khối chính trong phần lõi mã khối AES cải tiến như trong hình 8.a.

5. Khối sinh khóa vòng

Khối sinh khóa vòng thực hiện sinh 10 khóa con từ khóa chính. Chi tiết trong hình 8.b.

6. Khối điều khiển quá trình mã hóa và giải mã

Hình 8.c là khối điều khiển toàn bộ module mã hóa/giải mã AES cải tiến được thực hiện theo nguyên lý của FSM (*máy trạng thái*). Các tín hiệu đầu vào điều khiển là "Start", xung nhịp "clk". Các tín hiệu điều khiển đầu ra là "load_d_new", " Valid_out", "Shift".



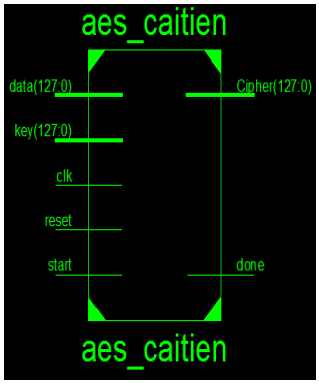
a. Khối hàm vòng b. Khối sinh khóa vòng c. Đồ hình trạng thái FSM module điều khiển

Hình 8. Các khối chức năng của AES cải tiến

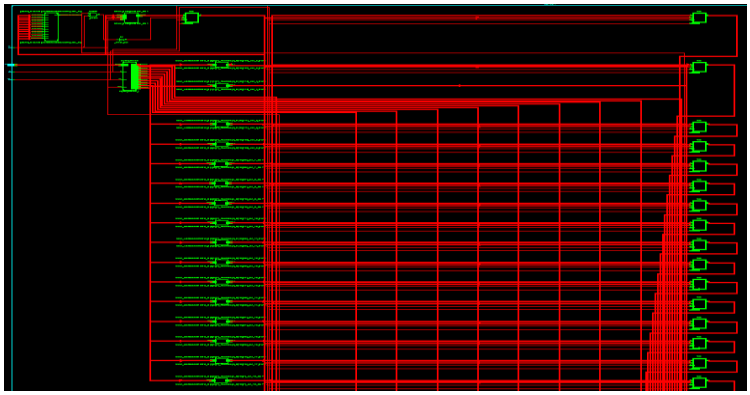
C. Kết quả thiết kế cứng hóa AES cải tiến trên FPGA

Thuật toán AES cải tiến được thực hiện trên FPGA với giao diện phần cứng như trong hình 9.a gồm:

1 cổng dữ liệu đầu vào 128 bit (data).
1 cổng vào khóa 128 bit (key).
Tín hiệu xung nhịp clock (clk), tín hiệu Reset.
Tín hiệu điều khiển bắt đầu quá trình mã hóa/giải mã "Start".
1 cổng dữ liệu mã đầu ra 128 bit (Cipher).
1 cổng báo trạng thái 1 bit (Done).



a. Giao diện Module cứng hóa AES cải tiến



b. Kiến trúc cứng khóa của Module AES cải tiến

Hình 9. Module cứng hóa AES cải tiến

Core AES cải tiến có khả năng hoạt động chính xác ở tốc độ 261 Mhz trên chip FPGA Zynq 7z020 [8], tương ứng với thông lượng (246*128)/(10+2) = 2,624 Gbps.

Thông số kỹ thuật của module:

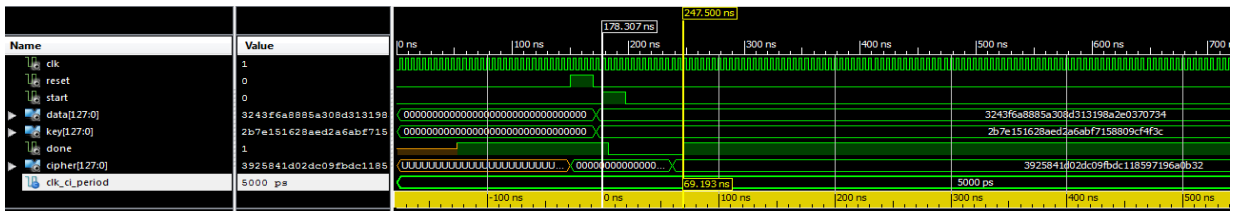
Device utilization summary:

Selected Device : 7z020clg400-3
Slice Logic Utilization:
Number of Slice Registers: 2956 out of 106400 2%
Number of Slice LUTs: 16775 out of 53200 31%
Number used as Logic: 16775 out of 53200 31%
==========
Timing Summary:

Speed Grade: -3
Minimum period: 4.059ns (Maximum Frequency: 246.366MHz)
Minimum input arrival time before clock: 1.238ns
Maximum output required time after clock: 1.575ns
Maximum combinational path delay: No path found

D. Đánh giá thời gian thực thi

Thuật toán mã khối AES cải tiến thực thi trên FPGA với tốc độ khá cao, thời gian tính toán một khối 128 bit tương ứng 69 ns như thể hiện chi tiết trong hình 10. Với tốc độ này hoàn toàn phù hợp trong triển khai cho bảo mật các mạng có tài nguyên hạn chế cũng như yêu cầu tốc độ tính toán nhanh.



Hình 10. Thời gian mã hóa AES cải tiến trên ISIM với 1 block 128 bit

IV. KẾT LUẬN

Thuật toán mã khối AES là một thuật toán mã hóa mạnh được NIST cũng như nhiều tổ chức sử dụng trong quá trình mã hóa bảo vệ dữ liệu trên đường truyền trong các mạng. Trong mỗi môi trường truyền thông cần có những cải tiến cả về mặt nâng cao độ mật cũng như tăng về tốc độ mã hóa và giải mã. Bài báo đã tập trung vào việc thiết kế lại

những hàm con trong các vòng mã hóa và giải mã để cải thiện phần nào độ mật so với AES chuẩn. Từ đó, cũng tăng tốc độ mã hóa và giải mã bằng giải pháp cứng hóa thông qua mô phỏng bằng công cụ lập trình FPGA.

Với thời gian mã hóa và giải mã nhanh hơn rất nhiều và có độ bảo mật được đánh giá thông qua sự xáo trộn của bản mã. Với những cải tiến nhỏ này giúp thuận lợi trong quá trình sử dụng mật mã AES trong các mạng có tài nguyên hạn chế cũng như yêu cầu thời gian thực hiện nhanh như mạng truyền số liệu trong công nghiệp chẳng hạn.

TÀI LIỆU THAM KHẢO

[1] J. Daemen, V. Rijmen, "The block cipher Rijindael", in Proc. Third International Conference on smart card Research and Applications, CARDIS'98, Lecture Notes in computer Science, Berlin, vol.1820, Springer, pp. 277_284, 2000.

[2] J. Daemen and V. Rijmen, "The Design of Rijndael: AES – The Advanced Encryption Standard." Springer-Verlag, 2002.

[3] Federal Information Processing Standards Publications (FIPS 197), "Advanced Encryption Standard (AES)", vol. 26 Nov. 2001.

[4] William Stallings, Cryptography and Network Security, Prentice Hall, 2008.

[5] A. Kumar, "Effective implementation and avalanche effect of AES," International Journal of Security, Privacy and Trust Management (IJSPTM), vol. 1, no. 3/4, pp. 31-35, August 2012.

[6] Barreto, P. S. L. M. and Rijmen, V. "The Anubis Block Cipher. First Open NESSIE Workshop", November 2000.

[7] Amish Kumar, Mrs. Namita Tiwari, "Effective implementation and avalanche effect of AES," International Journal of Security, Privacy and Trust Management (IJSPTM), Vol. 1, 2012.

[8] Zynq-7000 All Programmable SoC Overview, Xilinx, DS190 (v1.10) September 27, 2016.

A METHOD OF IMPROVING BLOCK CIPHER APPLY TO THE NETWORK REQUIRES FAST PROCESSING TIME

Nguyen Dao Truong, Le My Tu, Nguyen Doan Cuong

ABSTRACT: *For security in the process of data transmission on the network, the application of the solution of encryption techniques is indispensable. However, in networks with limited resources and fast response times, improvements are needed. This paper presents a method for improving block ciphers to improve the security of the cryptosystem based on the key dependence of the round transform functions during encryption and decryption. In addition, the paper also proposes a hardening solution by using FPGA simulators to speed up encryption and decryption in limited-resource networks.*