Về một giải pháp cứng hóa phép tính lũy thừa modulo

Trần Văn Thắng, Hoàng Văn Thức

***Tóm tắt*— Phép tính lũy thừa modulo là phép tính cơ bản trong các nguyên thủy mật mã RSA. Đây là phép tính phức tạp, tiêu tốn tài nguyên và thời gian thực hiện (đặc biệt với các số lớn). Thực thi cứng hóa phép tính lũy thừa modulo trên FPGA giúp nâng cao tốc độ xử lý, giảm thời gian cho phép tính, đáp ứng yêu cầu trong bài toán thực tế. Trọng tâm phép tính lũy thừa là phép nhân modulo số lớn. Trong nội dung bài báo chúng tôi trình bày giới thiệu, phân tích, lựa chọn thuật toán lũy thừa modulo và phép nhân modulo Montgomery dựa trên một số công trình nghiên cứu trên thế giới. Phép tính lũy thừa modulo được thực thi bằng ngôn ngữ mô tả phần cứng HDL Verilog số modulo lựa chọn 2048 bit, chip FPGA XC7z045. Kết quả thực hiện phép tính lũy thừa modulo với hai giải pháp thiết kế lõi IP nhân Montgomery khác nhau được tổng hợp ngắn gọn trên Bảng 1 và Bảng 2.**

***Abstract*— Modular exponentiation is the basic operation in cryptography algorithm RSA. This's a complicated algorithm, consuming resource and time to implement (especially with large number). Hardware implementation of modular exponentiation on the FPGA would increase speed, reduce computation time that is required by the practice. The heart of modular operand is modular multiplication of large number. In this paper we presented introduction, the analysis, choosing modular exponentiation algorithm and modular multiplication Montgomery based on several public researchs on the world. Modular exponentiation operation is implemented with hardware language HDL Verilog with the modulus is chosen as 2048 bit, chip FPGA XC7z045. Implementation results of modular exponentiation with two different designs of IPcore Montgomery is briefly presented in Table 1 and Table 2.**

***Từ khóa:* RSA; Montgomery multiplication; FPGA.**

***Keywords*: RSA; Montgomery multiplication; FPGA.**

Bài báo được nhận ngày 01/11/2018. Bài báo được gửi nhận xét và được chấp nhận đăng bởi phản biện thứ nhất vào ngày 02/12/2018 và 20/12/2018. Bài báo được gửi nhận xét và được chấp nhận đăng bởi phản biện thứ hai vào ngày 5/12/2018 và 28/12/2018.

I. GIỚI THIỆU

Có thể nói cho đến thời điểm hiện tại RSA là hệ mật mã khóa công khai đang được ứng dụng rộng rãi nhất trong các sản phẩm bảo mật và an toàn thông tin. Để hệ thống mật mã RSA được ứng dụng một cách hiệu quả và an toàn, trên thế giới đã có nhiều công trình nghiên cứu cải tiến thuật toán tính lũy thừa modulo trên phần cứng và phần mềm, đây là phép tính cơ bản tiêu tốn nhiều thời gian tính toán nhất trong các nguyên thủy mật mã RSA. Trên cơ sở các kết quả đã được công bố trên thế giới, trong bài viết này chúng tôi trình bày việc phân tích, lựa chọn và triển khai cài đặt cứng hóa thuật toán tính lũy thừa modulo với độ dài modulo là 2048 bit, đưa ra một số đánh giá ban đầu khi kết quả cài đặt được thực thi trên Chip Xilinx XC7z045.

Để thực hiện phép lũy thừa modulo, chúng ta có thể sử dụng một trong các thuật toán: phép lũy thừa nhị phân từ trái qua phải (Left-to-right Binary Exponentiation), phép lũy thừa nhị phân từ phải qua trái (Right-to-left Binary Exponentiation) phép lũy thừa cửa sổ trượt (Slide-window Exponentiation). Tùy thuộc vào môi trường cài đặt mà mỗi thuật toán có những điểm mạnh yếu khác nhau. Dưới đây chúng tôi xin giới thiệu một số thuật toán tính lũy thừa modulo đưa ra phân tích, lựa chọn ra một thuật toán hiệu quả thực thi trên nền tảng phần cứng FPGA.

Phần tiếp theo của bài báo sẽ có bố cục như sau: Mục II sẽ trình bày một số thuật toán lũy thừa modulo và một số cải biên cho phép nhân nhanh Montgomery. Tiếp theo Mục III sẽ trình bày cách triển khai thực hiện. Kết quả thực hiện sẽ được trình bày trong Mục IV và cuối cùng là Mục Kết luận.

II. THUẬT TOÁN LŨY THỪA MODULO

A. Một số thuật toán lũy thừa modulo

Thuật toán [1]. *Thuật toán lũy thừa từ trái qua phải k-ary*

Đầu vào: g và $e = (e_t e_{t-1}....e_1 e_0)_b$ ở đây $b = 2^k$ và $k \geq 1$;

Đầu ra: g^e

1. Tính toán trước

1.1. $g_0 \leftarrow 1$

1.2. Cho i chạy từ 1 tới $(2^k - 1)$ thực hiện:

$g_0 \leftarrow g_{i-1}$ do đó, $g_i = g^i$

2. $A \leftarrow 1$

3. Cho i chạy từ t giảm về 0 thực hiện như sau:

3.1. $A \leftarrow A^{2k}$

3.2. $A \leftarrow A.g_i$

4. Trả lại kết quả A

Thuật toán [2]. *Thuật toán lũy thừa sliding-window*

Đầu vào: g và $e = (e_t e_{t-1}....e_1 e_0)_2$ với $e_t = 1$, và một số nguyên $k >> 1$

Đầu ra: g^e

1. Tính toán trước

1.1. $g_1 \leftarrow g$, $g_2 \leftarrow g^2$

1.2. Cho i chạy từ 1 tới $(2^{k-1} - 1)$ thực hiện:

$g_{2i+1} \leftarrow g_{2i-1}.g_2$

2. $A \leftarrow 1$, $i \leftarrow t$

3. Trong khi $i >> 0$ thực hiện:

3.1. Nếu $e_i = 0$ thì $A \leftarrow A^2$, $i \leftarrow i-1$

3.2. *Nếu không* ($e_i \neq 0$), tìm chuỗi bit dài nhất $e_i e_{i-1}..... e_l$ sao cho $i - l+1 \leq k$ và $e_i = 1$ thực hiện như sau:

$A \leftarrow A_{2i-l+1}. g(e_i e_{i-1}....e_1)_2$, $i \leftarrow l-1$

4. Trả lại kết quả A

Thuật toán lũy thừa từ trái qua phải k-ary và thuật toán sliding-window thực hiện phép tính lũy thừa qua bốn bước tính toán và hai vòng lặp thực hiện tại bước 1 và bước 3. Trong đó, Bước 1 thực hiện tính toán trước giá trị g_i và g_{2i+1} làm giá trị đầu vào trong vòng lặp tại bước 3. Việc thực hiện tính toán trước giá trị của g_i và g_{2i+1} giúp quá trình thực hiện thuật toán nhanh, giải pháp này rất phù hợp với việc cài đặt thực trên phần mềm, bằng cách ta lưu trữ các giá trị tính toán trước.

Thuật toán [3]. *Lũy thừa nhị phân từ trái qua phải*

Đầu vào: $g \in G$, A là một số nguyên dương và $e = (e_t e_{t-1}....e_1 e_0)_2$

Đầu ra: g^e

1. $A \leftarrow 1$

2. Cho i chạy từ t giảm dần về 0 thực hiện như sau:

2.1. $A \leftarrow A.A$

2.2. Nếu $e_i = 1$ thì $A \leftarrow A.g$

3. Trả lại kết quả A

Thuật toán [4]. *Lũy thừa nhị phân từ phải qua trái*

Đầu vào: $g \in G$ và một số nguyên dương $e>>1$

Đầu ra: g^e

1. $A \leftarrow 1$, $S \leftarrow g$

2. Trong khi $e \neq 0$ thực hiện như sau:

2.1. Nếu e lẻ thì $A \leftarrow A \,.\, S$

2.2. $e \leftarrow e/2$

2.3. Nếu $e \neq 0$ thì $S \leftarrow S \,.\, S$

3. Trả lại kết quả A

Đối với Thuật toán 3, trong vòng lặp tại bước 2.1 và 2.2 kết quả phép tính A trong bước 2.1 làm giá trị đầu vào cho bước 2.2. Do vậy quá trình thực hiện phải tính toán trình tự nối tiếp kết thúc bước 2.1 rồi mới đến bước 2.2. Trong khi đó, khác với Thuật toán 3, ở Thuật toán 4, trong vòng lặp tại bước 2 kết quả phép tính bước 2.1 và 2.3 có thể thực hiện tính toán song song để nâng cao tốc độ thực thi phép tính. Trọng tâm phép lũy thừa là phép nhân modulo và phép bình phương modulo số lớn. Trên thế giới gần đây đã có nhiều công trình công bố liên quan đến việc nâng cao tính hiệu quả phép nhân và phép bình phương modulo, trong đó phải kể tới thuật toán nhân Montgomery modulo. Trong phần tiếp theo chúng tôi xin giới thiệu một giải pháp cải biên cho phép nhân Montgomery modulo số lớn, đồng thời đưa ra kết quả thực hiện và đánh giá việc cài đặt thuật toán lũy thừa modulo trên ngôn ngữ mô tả phần cứng FPGA.

B. Một số cải biên cho phép nhân nhanh Montgomery

Thuật toán [5]. *Thuật toán nhân Montgomery cơ bản*

Đầu vào: X,Y,N với $0 \le X,Y < N$

Đầu ra: $P = (X*Y(2^{-n})) \bmod N$

N: số modulo

n: độ dài theo bit của N

X_i: bit thứ i của X

1. $P := 0$
2. Với ($i=0$; $i < n$; $i = i++$) thực hiện
   2.1. $P = P + X_i*Y$
   2.2. Nếu $P_0 = 1$ thì $P = P + N$;
   2.3. $P = P/2$;
3. Nếu $P > N$ thì $P = P - N$

Hạn chế của thuật toán nhân Montgomery cơ bản là sử dụng 2 bộ cộng có nhớ trong vòng lặp ở bước 2.1 và 2.2, do đó độ trễ lan truyền trong cài đặt cứng hóa sẽ bằng n độ trễ lan truyền của một FA (Full Adder) một bit. Có rất nhiều phương pháp trong thiết kế để tăng hiệu quả tính toán và giảm thời gian trễ cho bộ cộng trong đó phải kể tới phương pháp sử dụng bộ cộng CSA (Carry Save Adder). Bộ cộng CSA không phụ thuộc vào cờ nhớ lan truyền nên thời gian trễ trong mỗi phép tính đúng bằng độ trễ của một bộ cộng FA một bit. Tuy nhiên kết quả đầu ra của bộ cộng CSA là kết quả trung gian chưa phải là kết quả cuối cùng của phép tính. Để thấy được việc ứng dụng bộ cộng CSA vào thuật toán nhân Montgomery cơ bản như thế nào, chúng ta xem xét các cải biên dưới đây.

Thuật toán [6]. *Thuật toán Faster Montgomery*

Đầu vào: X,Y,N với $0 \le X,Y < N$

Đầu ra: $P = (X*Y(2^{-n})) \bmod N$

N: số modulo

n: độ dài theo bit của N

X_i: bit thứ i của X

1. S = 0, C = 0;
2. Với ($i=0$; $i < n$; $i = i++$) thực hiện
   2.1.
   Nếu $((S_0 = C_0)$ *và* $(X_i))$ thì $I = 0$
   Nếu $((S_0 != C_0)$ *và* $\sim X_i$ thì $I = N$
   Nếu $((S_0 \wedge C_0 \wedge Y_0)\ \&\ X_i)$ thì $I = Y$
   Nếu $(S_0 \wedge C_0 \wedge Y_0)\ \&\ \sim X_i$ thì $I = N + Y$
   2.2. $(S,C) := CSA(S, C, I)$;
   2.3. $S := S$ Div 2 ; $C := C$ Div 2
3. $P = S + C$;
4. Nếu $P > N$ thì $P = P - N$;

Điểm nổi bật của thuật toán Faster Montgomery so với thuật toán Montgomery cơ bản đó là trong vòng lặp thay thế hai bộ cộng có nhớ bằng một bộ cộng CSA, với sự thay thế này giảm được tài nguyên thiết kế, giảm được độ trễ cờ nhớ trong bộ cộng số lớn nâng cao được tần số hoạt động của lõi IP khi thực hiện giải pháp này. Hạn chế trong thuật toán Faster Montgomery vẫn sử dụng một bộ cộng có nhớ ở ngoài vòng lặp trong Bước 3, vì vậy tần số tổng hợp của IPcore Faster Montgomey vẫn phụ thuộc vào tần số bộ cộng này. Tiếp theo chúng tôi sẽ giới thiệu một kết quả được công bố trong bài báo **[**3**]** để cải tiến bộ cộng trên.

Thuật toán [7]. *Thuật toán Semi Carry Save Adder Montgomery*

Đầu vào: X,Y,N với $0 \le X,Y < N$

Đầu ra: $SS[k+2] = (X*Y(2^{-n})) \bmod N$

N: số modulo

n: độ dài theo bit N

1. $(SS,SC) = CSA(Y,N,0)$;
2. Trong khi (SC!= 0)
   $(SS,SC) = CSA(SS,SC,0)$;
3. $D = SS$;
4. $SS[0] = 0$; $SC[0] = 0$;
5. Với (i=0; i<n+1; i++) thực hiện
   $q_i = (S[i]_0 + C[i]_0 + X_i * Y_0) \bmod 2$;
   Nếu ($X_i = 0$ và $q_i = 0$) thì $I = 0$;
   Nếu ($X_i = 0$ và $q_i = 1$) thì $I = N$;
   Nếu ($X_i = 1$ và $q_i = 0$) thì $I = Y$;
   Nếu ($X_i = 1$ và $q_i = 1$) thì $I = D$;
   $(SS[i+1], SC[i+1]) = CSA(SS[i], SC[i], I)/2$;
6. Trong khi ($SC[k+2]$!= 0)
   $(SS[k+2], SC[k+2]) = CSA(SS[k+2], SC[k+2], 0)$;
7. Trả lại kết quả $SS[k+2]$;

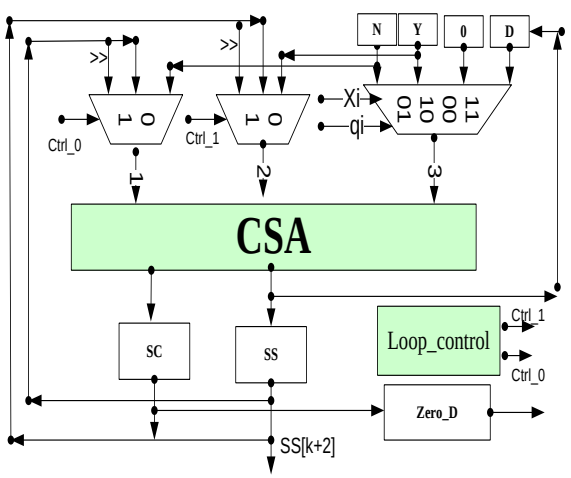
Điểm nổi bật giải pháp Semi Carry Save Adder Montgomery (SCSAMM) là thay thế bộ cộng có nhớ trong Bước 3 của thuật toán Faster Montgomery bằng bộ cộng SCSA tại Bước 6 trong thuật toán SCSA Montgomery. Với giải

pháp thiết kế này thuật toán không còn sử dụng bộ cộng có nhớ. Do đó, sẽ khắc phục được hạn chế trong thuật toán Faster Montgomery, nâng cao tần số hoạt động cho lõi IP nhân Montgomery sau khi tổng hợp, và giảm thời gian thực hiện một phép tính để giải pháp này có thể áp dụng hiệu quả với những số modulo có kích thước lớn như 2048, 3072, hoặc 4096.

III. PHƯƠNG PHÁP TRIỂN KHAI THỰC HIỆN THUẬT TOÁN LỰA CHỌN

A. Thiết kế cứng hóa lõi IP SCSA Montgomery

Lõi IP SCSA được xây dựng thiết kế dựa trên thuật toán [7] với độ dài số modulo 2048 bit, dưới đây là mô hình thiết kế lõi IP SCSA Montgomery trên FPGA.



Hình 1. Mô hình thiết kế lõi
IP SCSA Montgomery trên FPGA

Lõi IP SCSA được thiết kế sử dụng một bộ cộng trung tâm CSA với ba ngõ đầu vào 1,2,3 và hệ thống các thanh ghi lưu trữ *X, SS, SC, N*. Trong đó thanh ghi *X, N* được thiết kế để lưu giá trị đầu vào, *SS* và *SC* lưu kết quả tính toán đầu ra, các thanh ghi được thiết kế có độ dài 2048 bit. Bộ điều khiển vòng lặp đóng vai trò là máy thái, điều khiển quá trình hoạt động của lõi IP đồng thời sinh ra các tín hiệu điều khiển Ctrl_0, Ctrl_1 được sử dụng để lựa chọn đầu vào cho bộ cộng CSA. Kết quả mỗi vòng thực hiện được lưu ở hai thanh ghi *SS*, *SC* đồng thời làm giá trị phản hồi cho vòng lặp tiếp theo. Số vòng lặp phụ thuộc vào kích thước chiều dài chuỗi bit của số modulo. Dưới đây là kết quả tổng hợp thực hiện lõi IP SCSA Montgomery trên chip FPGA.

BẢNG 1. KẾT QUẢ THỰC THI LÕI IP MONTGOMERY TRÊN FPGA

Thuật toán	Chip FPGA	Chiều dài bit (k)	Tần số hoạt động (MHz)	Tài nguyên thiết kế (LUTs)
TT[6] Faster MM	XC7 Z045	2048	194	25.279
TT[7] SCSA MM			284	20.241

Lõi IP SCSA Montgomery được thiết kế theo Thuật toán [7] đã giải quyết được hạn chế trong Thuật toán [6] nâng cao tần số hoạt động của lõi IP từ 194 MHz lên 284 MHz đây là một minh chứng cho thấy tính hiệu quả của lõi IP được thiết kế theo giải pháp SCSA Montgomery.

B. Thực thi phép tính lũy thừa modulo

1. RSA và phép tính lũy thừa modulo

Phép mã hóa giải mã RSA thực hiện tính toán như sau: $C = M^e \pmod n$, $M = C^d \pmod n$ trong đó M là bản rõ, C là bản mã, e là khóa công khai (public key), d là khóa bí mật (private key), n là số modulo có kích thước k bit. Mối quan hệ giữa e,d,n được thể hiện qua biểu thức $e.d = 1 \pmod{(p-1)(q-1)}$, với $n = p.q$, trong đó p và q là hai số nguyên tố lớn có kích thước xấp xỉ $k/2$ bit.

2. Sử dụng lõi IP SCSA Montgomery trong phép tính lũy thừa modulo

Dựa trên việc phân tích ưu nhược điểm các thuật toán cài đặt phép tính lũy thừa modulo, trong bài báo nhóm tác giả thực hiện theo thuật toán [4] sử dụng song song hai lõi IP SCSA Montgomery được ký hiệu SCSA_M1, SCSA_M2 để nâng cao tốc độ cho phép tính lũy thừa modulo. Thuật toán [4] được triển khai như sau:

Thuật toán lũy thừa (C,M,d,n)

$K = 2^{2k} \pmod n$

k : kích thước chiều dài chuỗi bit

n: số modulo

1. $P[0]$ = SCSA_M1(K, C, n);
   $R[0]$ = SCSA_M2(K, 1, n);
2. Vòng lặp cho i chạy từ 0 tới k
   Nếu $d[i] = 1$ thì
   $R[i+1]$ = SCSA_M2($R[i]$, $P[i]$, n);
   Kết thúc nếu;

$P[i+1]$ = SCSA_M1($P[i]$, $P[i]$, n);

Kết thúc vòng lặp

3. M = SCSA_M1(*1*, $R[k]$, n);

Trả lại kết quả M;

Lõi IP phép tính lũy thừa modulo 2048 bit được lập trình bằng ngôn ngữ mô tả phần cứng HDL Verilog, được thực hiện kiểm tra theo bộ tiêu chuẩn tham số NIST FIPS 186-3. Dưới đây là một số kết quả thực hiện lõi IP phép tính lũy thừa modulo:

BẢNG 2. KẾT QUẢ CÀI ĐẶT THUẬT TOÁN [4] LŨY THỪA MODULO SỬ DỤNG CÁC IP BỘ NHÂN MONTGOMERY KHÁC NHAU TRÊN FPGA

Thuật toán [4] sử dụng IPcore	Chip FPGA	Chiều dài bit (k)	Tần số hoạt động (MHz)	Tài nguyên thiết kế (LUTs)
IPcore nhân Faster MM	XC7 z045	2048	194,4	48.740
IPcore nhân SCSA MM			281,2	42.311

Lõi IP phép tính lũy thừa modulo 2048 bit sử dụng lõi IP nhân modulo dựa trên giải pháp mới Semi Carry Save Adder (SCSA) Montgomery. Điều này khắc phục được một số hạn chế trước đó của phép nhân modulo được thiết kế theo giải pháp Faster Montgomery. Kết quả tổng hợp lõi IP trên nền tảng phần cứng chip FPGA XC7z045 như sau: số lượng thanh ghi 57.475 chiếm 13%, số cổng LUTs sử dụng 61.929 chiếm 28% tổng số tài nguyên của chip FPGA, tần số hoạt động lớn nhất của lõi IP đạt 281 MHz, được thể hiện trong Hình 2.

Device Utilization Summary (estimated values)

Logic Utilization	Used	Available	Utilization
Number of Slice Registers	57475	437200	13%
Number of Slice LUTs	61929	218600	28%
Number of fully used LUT-FF pairs	39024	80380	48%
Number of bonded IOBs	0	250	0%
Number of BUFG/BUFGCTRLs	2	32	6%

```
Speed Grade: -2

  Minimum period: 3.556ns (Maximum Frequency: 281.207MHz)
  Minimum input arrival time before clock: 1.091ns
  Maximum output required time after clock: 1.896ns
  Maximum combinational path delay: No path found
```

Hình 2. Kết quả tổng hợp lõi IP trên nền tảng phần cứng chip FPGA XC7z045

BẢNG 3. KẾT QUẢ THỰC HIỆN MỘT SỐ PHÉP TÍNH LŨY THỪA MODULO

Thuật toán [4] sử dụng lõi IP	Chip FPGA	Chiều dài bit (k)	RSA encrypt khóa e (ms)	RSA dencrypt khóa d (ms)
IPcore nhân Faster MM	XC7 z045	2048	0,24	21,1
IPcore nhân SCSA MM			0,22	15,6

Để thấy rõ hiệu năng của thuật toán 6 so với các thuật toán 3 và thuật toán 5, nhóm nghiên cứu đã tổng hợp và nạp lên chip Zynq7z045ffg676-2 thuật toán lũy thà modulo sử dụng các thuật toán nêu trên. Một số tiêu chí kỹ thuật so sánh giữa chúng có thể tóm lược như trong bảng 1 dưới đây (vẫn với các tham số trong [5])

BẢNG 4. SO SÁNH CÁC THUẬT TOÁN NHÂN MODULU TRÊN CHIP Zynq7z045ffg676-2

Thuật toán nhân modulo	Tần số clock MHz	Tài nguyên thiết kế (%)	Thời gian phép mã hóa (ms)	Thời gian phép giải mã (ms)
T.Toán 3	141,36	25%	0,338	42,00
T.Toán 5	225,78	33 %	0,218	21,10
T.Toán 6	284,10	28%	0,24	15,67

Trong bảng Bảng 4 cho ta thấy lõi IP lũy thừa modulo áp dụng giải pháp thiết kế lõi IP bộ nhân modulo Montgomery theo phương pháp cơ bản,tài nguyên chiếm 23%, tần số hoạt động lớn nhất 141 MHz, thời gian thực hiện phép tính lũy thừa 42,0 ms. Áp dụng giải pháp thiết kế lõi IP bộ nhân modulo Faster Montgomery tài nguyên chiếm 33 %, tần số hoạt động 225 MHz, thời gian thực hiện phép tính lũy thừa 21,1 ms. Còn với việc áp dụng thiết kế lõi IP bộ nhân modulo theo giải pháp SCS-based Montgomerytài nguyên chiếm 28% tổng số cell logic, thanh ghi, tần số hoạt động 284 MHz, thời gian thực hiện một phép tính 15,67 ms, đây là một kết quả mới rất ý nghĩa trong bài toán thực thi hiệu năng cao (các kết quả trên được thực hiện trên chip trên chip Zynq7z045ffg676-2)

IV. KẾT QUẢ THỰC HIỆN

Trên cơ sở các kết quả nghiên cứu đã được công bố trên thế giới về tối ưu hóa hiệu năng khi triển khai các thuật toán tính lũy thừa modulo bằng ngôn ngữ mô tả phần cứng, nhóm nghiên cứu đã phân tích, đánh giá và lựa chọn và cài đặt thành công thuật toán lũy thừa nhị phân từ phải qua trái áp dụng giải pháp Semi Carry Save Adder trong bộ nhân Montgomery. Kết quả thực thi trên dòng chip Zynq 7z045ffg676-2 cho thấy các ưu điểm về tài nguyên thiết kế, tần số hoạt động, thời gian thực hiện đã thể hiện kết quả tốt hơn so với một số thuật toán được công bố trước đó. Các kết quả trên cũng đã được chúng tôi tích hợp cho các module phần cứng bảo mật (HSM) do Viện Khoa học Công nghệ mật mã, Ban Cơ yếu Chính phủ chủ trì thiết kế chế tạo.

V. KẾT LUẬN

Trong khuôn khổ bài báo này, nhóm tác giả trình bày và cập nhật một số công trình nghiên cứu khoa học trên thế giới. Từ đó, phân tích một số thuật toán, để lựa chọn ra một phương pháp hiệu quả có thể thiết kế cứng hóa phép tính lũy thừa modulo 2048 bit. Trong phần thực nghiệm, nhóm tác giả triển khai thực thi phương pháp này bằng ngôn ngữ mô tả phần cứng HDL Verilog, sử dụng nền tảng phần cứng có sẵn Kit FPGA ZynqZC706. Sau đó, đưa ra một số kết quả so sánh giữa các giải pháp mà nhóm đã thực hiện với các giải pháp đã có, để làm rõ tính ưu việt của giải pháp nghiên cứu mới mà nhóm lựa chọn. Trong thời gian tới, hướng nghiên cứu tiếp theo nhóm sẽ thực hiện giải pháp trên với phép tính lũy thừa modulo 4096 bit.

TÀI LIỆU THAM KHẢO

[1]. A. Menezes, P. van Oorschot and S. Vanstone.. "Handbook of Applied Cryptography", CRC Press, 1996.

[2]. Ranjeet Behera, Pradhan Abhisek, "FPGA Implementation of RSA algorithm and to develop a crypto based security system", National Institute of Technology, Rourkela, 2012-2013.

[3]. D. J. ANITHA, G. SUJATHA, P. JAYARAMI REDDY, "High Speed Low Cost New Semi Carry Save Adder Montgomery Modular Multiplier", International Journal of Scientific Engineering and Technology Research, 2016.

[4]. Mclvor, Ciaran, Maire McLoone, and John V. McCanny. "Fast Montgomery modular multiplication and RSA cryptographic processor architectures.", Signals, Systems and Computers, 2004. Conference Record of the Thirty-Seventh Asilomar Conference on. Vol. 1. IEEE, 2003.

[5]. FIPS PUB 186-4. "Digital Signature Standard (DSS")*.* July, 2013.

SƠ LƯỢC VỀ TÁC GIẢ

ThS. Trần Văn Thắng

Đơn vị công tác: Viện Khoa học – Công nghệ mật mã, Ban Cơ yếu Chính phủ

Email: vanthang.qsbk@gmail.com

Quá trình đào tạo: Nhận bằng kỹ sư năm 2014 và Thạc sĩ năm 2018 chuyên ngành Kỹ thuật điện tử.

Hướng nghiên cứu hiện nay: Công nghệ vi mạch, FPGA.

TS. Hoàng Văn Thức

Đơn vị công tác: Viện Khoa học – Công nghệ mật mã, Ban Cơ yếu Chính phủ.

Email: thuchv@yahoo.com

Quá trình đào tạo: Nhận bằng kỹ sư năm 1998 và Thạc sĩ năm 2004 chuyên ngành Kỹ thuật mật mã, Học viện Kỹ thuật mật mã. Nhận bằng Tiến sĩ Toán học, Viện Khoa học - Công nghệ quân sự năm 2012.

Hướng nghiên cứu hiện nay: Khoa học - Công nghệ Mật mã.