VỀ MỘT HÀM BĂM MỀM DẺO

Đỗ Thị Bắc[1*], Nguyễn Hiếu Minh[2]
[1]Trường Đại học Công nghệ Thông tin và Truyền thông – ĐH Thái Nguyên,
[2]Học viện Kỹ thuật mật mã

TÓM TẮT

Việc phát triển các hàm băm mới đáp ứng yêu cầu mềm dẻo cho các ứng dụng và đảm bảo tốc độ cao là một yêu cầu tất yếu trong truyền thông. Trong bài báo này sẽ đề xuất một hàm băm được xây dựng riêng có cơ chế xử lý mềm dẻo để phù hợp hơn cho các ứng dụng không dây, đồng thời có khả năng chống đụng độ mạnh, có hiệu quả tích hợp cao trên FPGA. Nó được xây dựng theo mô hình sử dụng các hàm băm lặp quay vòng, phần tử điều khiển được. Các tham số đầu vào của hàm băm có thể được tùy biến bởi người dùng như: độ dài bản tin, độ dài khóa, số lần lặp của hàm quay vòng, đô lớn của kết quả băm. Đây là những vấn đề rất được quan tâm trong bài báo này

Từ khóa: *hàm băm, phần tử điều khiển được, thuật toán MBA, phần tử chuyển mạch phụ thuộc dữ liệu, FPGA*

MỞ ĐẦU

Trong các hệ thống mạng không dây hiện tại, các hàm băm thường được sử dụng là họ MDx, họ SHA-x hay một số các hàm băm chuyên dụng không được công khai [1,2, 3]. Chúng thường được chia thành hai loại cơ bản là MAC (*Message Authentication Codes*) và MDC (*Manipulation Detection Codes*). MDC tập trung vào việc xác thực nguồn gốc dữ liệu và xác thực tính toàn vẹn dữ liệu, còn MAC đảm bảo việc xác thực người gửi và toàn vẹn dữ liệu. Mỗi dòng hàm băm được thiết kế dựa trên một trong các phương pháp: lặp, modulo số học, mật mã khối hay được thiết kế riêng. Tuy nhiên, chúng đều có những hạn chế chung là phải sử dụng thêm hệ thống mã hóa trung gian để hoàn tất việc đảm bảo an toàn xác thực bản tin. Đồng thời kích thước khối dữ liệu đầu vào và đầu ra của các dòng hàm băm này khó có thể biến đổi linh hoạt. Vì vậy các hàm băm trên thiếu tính mềm dẻo khi đưa vào các ứng dụng cụ thể có các yêu cầu khác nhau. Do đó, việc phát triển các hàm băm mới đáp ứng yêu cầu mềm dẻo cho các ứng dụng và đảm bảo tốc độ cao là một yêu cầu tất yếu. Xu hướng xây dựng các hàm băm mới đạt được tốc đô cao kết hợp với mô hình sử dụng các hàm băm vòng và có tính mềm dẻo về các tham số đầu vào có thể được tùy biến bởi người dùng như: độ dài bản tin, độ dài khóa, số lần lặp của hàm quay vòng, đô lớn kết quả băm là những vấn đề đang được quan tâm

Trong bài báo này, chúng tôi đề xuất hàm băm được xây dựng riêng có cơ chế xử lý mềm dẻo để phù hợp hơn cho các ứng dụng không dây. Bài báo có cấu trúc: mở đầu, mô tả thuật toán băm đề xuất, phân tích và đánh giá kết quả, kết luận.

ĐỀ XUẤT THUẬT TOÁN BĂM MỀM DẺO

Thực tế, khi xây dựng hàm băm các yêu cầu ngày càng được nâng cao để phù hợp hơn cho các ứng dụng. Ngoài khả năng chống đụng độ mạnh, dễ ứng dụng, . . thì yếu tố quan trọng khác là tốc đô thực thi và sự mềm dẻo của hàm băm là các yếu tố quan trọng. Chính vì vậy, sử dụng cơ chế phụ thuộc dữ liệu trong quá trình thiết kế hàm băm đã trở thành một xu hướng mới. Điểm này thể hiện rõ nhất trong các thuật toán băm dựa trên mã khối. Tuy nhiên cơ chế phụ thuộc dữ liệu trong các thuật toán này tương đối phức tạp và hoàn toàn dựa trên các hàm tính toán, các phép biến đổi cơ bản hoặc trên mẫu dữ liệu nguyên thủy ở vòng tính toán trước nên làm giảm tốc độ và tiềm ẩn nguy cơ bị tấn công thám mã. Thuật toán băm đề xuất có sử dụng cơ chế phụ thuộc dữ liệu, tuy nhiên quá trình này sẽ được thực hiện thông qua một bảng các số giả

* *Email: dtbac@ictu.edu.vn*

ngẫu nhiên. Bảng này được gọi là "*Bảng truy vấn phụ thuộc dữ liệu*". Sau đây là chi tiết các vấn đề trong thuật toán băm.

Mục tiêu thiết kế

Thuật toán băm đề xuất được thiết kế với mục tiêu

- Khả năng chống đụng độ mạnh,
- Sử dụng để xác thực tính toàn vẹn của bản tin;
- Dùng mô hình hàm băm lặp quay vòng để tăng hiệu ứng lan truyền và độ an toàn, số lần lặp của hàm quay vòng được người dùng tùy biến,
- Kết hợp dùng các phép toán modulo số học để tăng tốc độ trong quá trình xử lý;
- Khóa được dùng như một tham số để tác động vào mỗi khối và mỗi vòng xử lý dữ liệu (khóa này là cơ sở để sinh ra các tập khóa con, không cần một thuật toán phức tạp để sinh khóa con như các thuật toán mật mã khóa đối xứng);
- Khóa con được sinh ra phụ thuộc vào đặc tính dữ liệu tại mỗi lần thực hiện hàm lặp quay vòng,
- Hàm băm đảm bảo tính mềm dẻo và nhiều tham số được tùy biến bởi người dùng (độ dài bản tin, độ dài khóa, số lần lặp của hàm quay vòng,. .);
- Độ dài của giá trị băm có thể tùy biến dễ dàng: 160, 192, 224 bit, . .

Mô hình thiết kế

Mô hình thiết kế thuật toán băm đề xuất được minh họa như trong hình 1. Từ mô hình cho thấy bản tin cần băm M sẽ được chia thành n tin M_i, với mỗi M_i lại chia thành z khối W_j (ở đây j = 1..z) dựa trên phương pháp tạo "*thông điệp đệm*". Tuy nhiên phần quan trọng nhất của hàm băm này lại thể hiện ở các đối tượng sau:

- ***Table Lookup***: là một bảng các số nguyên đủ lớn đóng vai trò như một khóa. Từ bảng này, kết hợp với các tham số phù hợp sẽ đưa ra các khóa con (SubKey: key[j], j=1..z). Table Lookup được gọi là "Bảng truy vấn phụ thuộc dữ liệu", cung cấp các khóa con trong quá trình mã hóa từng khối dữ liệu. Đảm bảo được việc xác thực tính toàn vẹn bản tin mà không cần dùng đến một hệ thống mã hóa trung gian

- Cơ chế sinh key[j] từ Table Lookup đã loại bỏ được nhược điểm phải sử dụng các thuật toán lập lịch sinh khóa con (SubKey) trong mã khối. Bộ giá trị (160, 192 bit,...): đóng vai trò như một *tham số* đầu vào của hàm lặp. Bộ giá trị này chịu tác động từ hàm lặp và khóa SubKey, vì vậy bộ giá trị này cũng bị phụ thuộc vào các biến đổi dữ liệu của các khối trước đó.

- Hàm f: hàm này chính là hàm lặp quay vòng, sử dụng các phép toán modulo số học để biến đổi dữ liệu dựa trên hai tham số truyền vào: khóa (SubKey: key[j], j=1..z) và bộ giá trị (160, 192 bit,..).

- m: là tham số được đưa vào bởi người dùng, m xác định số lần thực hiện lặp lại hàm quay vòng trên cả mảng dữ liệu W_i. Sau khi cả quá trình được lặp lại m lần, hàm băm sẽ trả về hai giá trị: một là thông điệp M' là thông điệp mã hóa của M, hai là bộ giá trị (160, 192 bit,...) chính là giá trị băm (Hash value).

Các thành phần trong thuật toán

Cấu trúc thiết kế của $P^{(L,e)}_{(32/32)}$

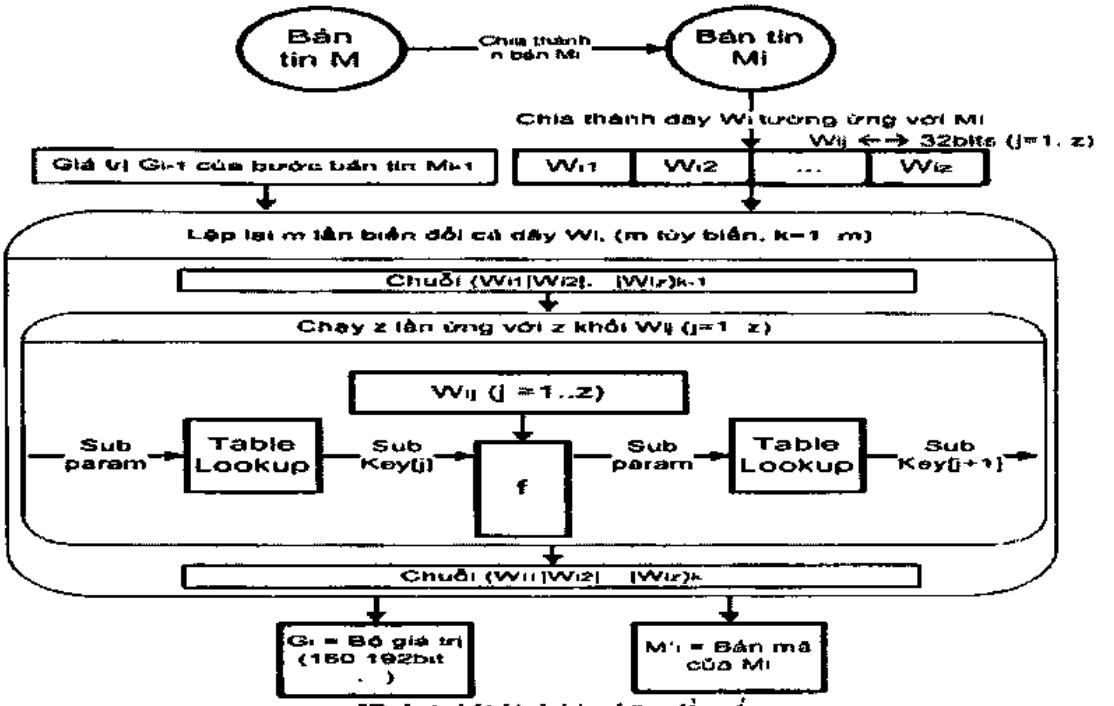
Trong thuật toán băm đề xuất có sử dụng toán tử chuyển mạch phụ thuộc vào dữ liệu $P^{(L,e)}_{(32/32)}$ trong việc tạo ra các cơ chế phụ thuộc vào dữ liệu hiệu quả [3].

Thuật toán băm MBM

Thuật toán băm đề xuất xây dựng trên cơ sở mô hình hàm băm lặp quay vòng. Với mô hình này, đầu vào là bản mã gốc M được chia thành các bản mã M_i, với mỗi bản mã M_i qua hàm băm lặp quay vòng dựa trên bảng truy vấn phụ thuộc dữ liệu. Kết quả thu được là giá trị băm ($G \geq 160$ bit) và bản mã M'_i tương ứng với khối M_i. Thuật toán băm lặp quay vòng đề xuất với tên gọi **MBM** mô tả dưới dạng giả mã như sau.

Procedure *MBM*

Đầu vào: khối dữ liệu vào của vòng thứ i là M_i được miêu tả thành chuỗi của các từ 32 bit $\overline{W_i}$: $(\overline{W_0}, \overline{W_1}, \ldots, \overline{W_{Z-1}})$ và các biến n, R, V, Y, U, N



Hình 1. *Mô hình hàm băm đề xuất*

1 Xác định kích thước của khối dữ liệu vào: $S \leftarrow z$;
2. Thiết lập bộ đếm thứ nhất j: $j \leftarrow 6$;
3. Thực hiện thủ tục ***Initialize*** khởi tạo giá trị ban đầu R, V, N, U, Y,
4 Thiết lập bộ đếm thứ hai k: $k \leftarrow 0$;
5. Thực hiện thủ tục ***ChangeNVUY***;
6. Thay đổi giá trị của từ 32 bit hiện thời. $W_k \leftarrow (W_k +_{32} V) \oplus U$;
7. Thay đổi giá trị của biến R: $R \leftarrow R +_{32} W_k$;
8 Kết thúc sự biến đổi của từ 32 bit hiện thời W_k. $W_k \leftarrow \mathbf{P}^{(V,0)}_{(32/22)}(W$
9 Tăng giá trị của bộ đếm thứ hai k: $k \leftarrow k+1$. Nếu $k \neq S$, thì chuyển đến bước thứ 5,
10 Giảm giá trị của bộ đếm thứ nhất j: $j = j-1$. Nếu $j \neq 0$, thì chuyển tới bước 4, trong trường hợp ngược lại Kết thúc.

Đầu ra:

- Bản mã M', ứng với khối M_i: $h_i := W_{i-1} | .. |W_1| W_0$;
- Giá trị băm: $G_i := R|N|V|Y|U$; trong đó:

a. Hàm băm lặp quay vòng h_i: được xác định theo mô hình hàm lặp quay vòng [1,2], với công thức sau:

$$h_i = E(M_i \oplus h_{i-1}) \oplus M_i, \quad i = 1, 2, \ldots, n$$

ở đây E – hàm băm được xây dựng ở trên; h_0 – giá trị ban đầu nào đó, là tham số của thuật toán; còn h_n – giá trị của hàm băm ở đầu ra (cũng gọi là mã hàm băm, hoặc bản tóm lược) của văn bản M.

b. Thủ tục Initialize: đây là thủ tục dùng để khởi tạo các giá trị ban đầu cho các biến 32 bit R, V, N, U, Y và được mô tả cụ thể như sau:

1 Thiết lập bộ đếm i: $i \leftarrow 0$
2. Gán giá trị ban đầu cho biến R, V, N, U, Y
$R \leftarrow Q_9, V \leftarrow Q_{17}, N \leftarrow Q_{21}, U \leftarrow Q_{33}, Y \leftarrow Q_{25}$,

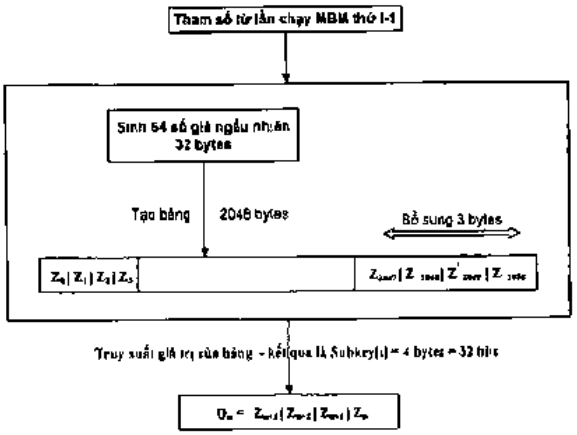
Trong thủ tục này các giá trị Q_i được xác định nhờ thủ tục ***TableQ***. Thủ tục này còn được sử dụng trong thủ tục ***ChangeNVUY***. Cụ thể thủ tục ***TableQ*** này được miêu tả ngay sau đây.

c. Thủ tục TableQ:

Dựa trên các tiêu chí về tiêu chuẩn thiết kế, việc thực hiện truy vấn bảng thông qua truy xuất từng nhóm 4 byte dữ liệu liền kề trên bảng dựa trên chỉ số của byte thứ nhất (gọi n là chỉ số), $0 \leq n \leq 2048$. Mỗi nhóm dữ liệu được truy xuất ra từ bảng sẽ có dạng sau:

$$Q_n = z_{n+3} \mid z_{n+2} \mid z_{n+1} \mid z_n$$

Dễ thấy, cần bổ sung thêm 3 byte vào bảng để với n=2048 vẫn có thể truy xuất được Q_{2048} Do đó bảng truy vấn dữ liệu phụ thuộc có kích cỡ là 2051 byte (hình 2)



Hình 2. *Hoạt động bảng truy vấn phụ thuộc dữ liệu*

Từ ý tưởng xây dựng bảng truy vấn dữ liệu phụ thuộc ở trên, có thể mô phỏng thủ tục tạo ***Table Q*** dưới dạng giả mã như sau.

Thiết lập bộ đếm i $i \leftarrow 0$;
Tính toán số 32 byte
$Q_i' = (a^{23+i} \bmod P)^{17} \bmod G$,
Tăng giá trị biến i: $i \leftarrow i+1$ Nếu $i \neq 64$, thì chuyển tới bước 2,
Tạo số 2051 byte $S' = q_2'||q_1'||q_0'||Q_{63}'||\ldots||Q_0'$ với $q_2'||q_1'||q_0' = Q_0' +_{24} 0$
Chuyển số S thành dãy các byte
$\overline{Q} = \{z_0, z_1, \ldots, z_{2050}\}$

Trong thủ tục ***TableQ*** các giá trị a, P, G được xác định như sau: a=0D; P=B25D28A71A62D775; G= 98915E7EC8265EDFCDA31E88F24809DD B064BDC7285DD50D7289F0AC6F49DD2 D (biểu diễn ở hệ cơ số 16); Ở đây G là tham số tùy chỉnh rất quan trọng trong việc sinh bảng giả ngẫu nhiên theo ý người sử dụng. a, P, G là các hằng số được cho ban đầu với các khung giá trị biểu diễn lần lượt là (1 byte, 8 byte, 32 byte) Khi đó dễ dàng thấy $a^{23+i} \geq 2^{64}$ và $(a^{23+i} \bmod P)^{17} \geq 2^{256}$, vì vậy $(a^{23+i} \bmod P)^{17} \bmod R$ luôn đảm bảo được ý nghĩa về mặt tính toán số học, đồng thời Z'_i sẽ là một số giả ngẫu nhiên

d. Thủ tục ChangeNVUY: thủ tục này có ý nghĩa làm thay đổi giá trị của các biến N,V,U,Y và cụ thể như sau:

1. Thay đổi giá trị N. $N \leftarrow \mathbf{P}_{32/32}^{(Y,0)}(N) \oplus \mathbf{R}$
2. Thiết lập biến trung gian n $n \leftarrow N +_{11} 0$;
3. Thay đổi giá trị của V bằng cách gán: $V \leftarrow \mathbf{P}_{(32/32)}^{(N,1)}(V) +_{32} Q_n$
4. Thay đổi giá trị biến n $n \leftarrow V +_{11} 0$;
5. Thay đổi giá trị của U bằng cách $U \leftarrow \mathbf{P}_{(32/32)}^{(V,1)}(U) \oplus Q_n$
6. Thay đổi giá trị biến n $n \leftarrow U +_{11} 0$;
7. Thay đổi giá trị Y bằng cách: $Y \leftarrow \mathbf{P}_{(32/32)}^{(N,1)}(Y) +_{32} Q_n$

PHÂN TÍCH VÀ ĐÁNH GIÁ HÀM BĂM ĐỀ XUẤT

Với thủ tục ***TableQ***, trọng tâm là việc sinh các số giả ngẫu nhiên. Bước 2, với a, P, G là các hằng số được cho ban đầu với các khung giá trị biểu diễn lần lượt là (1 byte, 8 byte, 32 byte) Đồng thời Q'_i là một số giả ngẫu nhiên. Ngoài ra G ở là tham số tùy chỉnh tạo ra sự mềm dẻo trong việc sinh bảng giả ngẫu nhiên theo người dùng

Với thủ tục ***ChangeNVYU***, việc chọn các giá trị cụ thể của các biến N, V, U, Y và R cho ra trạng thái cụ thể của cơ chế mã hóa hàm băm. Đây là yếu tố giúp cho hàm băm đề xuất có được cơ chế mềm dẻo. Hiệu ứng thác lũ ở đây được mở rộng do việc sử dụng $\mathbf{P}_{(32/32)}^{(V,e)}$, hiệu ứng thác lũ này có được liên quan đến ba yếu tố; 1) mỗi một bit của từ sẽ ảnh hưởng tới việc chọn các phần tử từ chuỗi giả ngẫu nhiên $\overline{Q}$; 2) sự biến dạng của từ W_k dẫn đến sự thay đổi của biến R; 3) sự thay đổi giá trị hiện thời của biến R kéo theo sự thay đổi giá trị tiếp theo của nó. Như vậy, sự thay đổi bất kỳ ở đầu vào của bộ biến đổi được tích lũy vào bộ lưu giữ của biến R, V, Y, U và N.

Trong thủ tục ***ChangeNVYU.*** Các biến N, V, Y, U vừa đóng vai trò là tổ hợp tạo nên giá trị băm, đồng thời là tham số quan trọng tại mỗi lần xử lý của hàm băm lặp quay vòng. Với từng khối dữ liệu cho thấy n, N, V, Q là những tham số đông, biến đổi dựa trên dữ liệu đầu vào tại mỗi vòng của hàm băm và bảng truy vấn phụ thuộc dữ liệu Điều đó tạo ra yếu tố mềm dẻo thực sự cho thuật toán và đóng

vai trò như một tham số độc lập tác động cho mỗi vòng biến đổi của N, V, Q.

Thuật toán băm đề xuất đã được tổ chức sao cho có thể dễ dàng mô phỏng theo kích thước bất kỳ của khối dữ liệu đầu ra. Việc tính toán giá trị băm từ văn bản nào đó $M=(M_1,M_2,...,M_n)$, mà nó được trình bày như các khối M_i nối liên tiếp. Ở đây các khối M_i có kích thước như nhau (khi cần thiết thì khối dữ liệu cuối cùng cần được bổ sung thêm các bit để trở thành khối có kích thước theo tiêu chuẩn).

Hàm băm lặp quay vòng *MBM:* hàm đóng vai trò như một hàm xử lý dữ liệu của các hàm mã hóa hay hàm nén. Hàm băm lặp quay vòng (MBM) sử dụng dữ liệu đầu vào mềm dẻo. Cụ thể, nếu quy định kích cỡ khối dữ liệu đầu vào là m, thì $m = S^*m_0$ với $m_0 = 32$ bit, S là số tự nhiên, $S \geq 4$. Như vậy, $m \geq 128$ để đảm bảo những tiêu chuẩn đầu tiên về an toàn bảo mật trong hàm băm và việc điều chỉnh m trở lên rất thuận tiện đối với người dùng thông qua biến S Ngoài việc quy định kích cỡ mềm dẻo với tham số đầu vào là khối dữ liệu, hàm lặp quay vòng còn phải sử dụng các tham số được sinh ra bởi bảng truy vấn phụ thuộc dữ liệu, cụ thể chính là các khóa con Subkey[i].

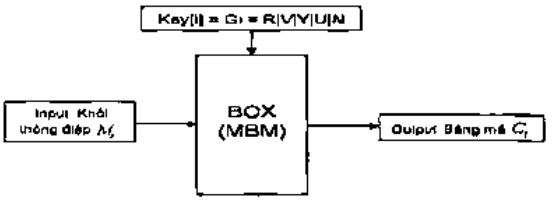
Hàm này tận dụng được các đặc tính về sự phụ thuộc dữ liệu, lặp quay vòng và toán tử số học tốc độ cao. Điều đặc biệt hơn, là sử dụng một bộ tham số R, V, Y, U, N như các khóa con (SubKey) để tính toán ở mỗi lần xử lý hiện tại cũng như tác động lan truyền sang lần tính toán kế tiếp. R, V, Y, U, N là những giá trị 32 bit được người sử dụng chủ động khai báo và khởi tạo ngay từ đầu thông qua hàm ***Initialize*** Đây chính là đặc tính mềm dẻo ở đầu vào của ***MBM***.

Ngoài ra bản thân trong ***MBM*** luôn hướng tới việc cho phép người lập trình can thiệp thông qua các biến nội bộ, cụ thể có thể thấy các biến S, j, v cho phép gán các giá trị phù hợp để tác động tới sự biến đổi và đặc biệt là số lần lặp của hàm. Trong thuật toán này, các bước từ 4 đến 8 được lặp lại 6^*S lần, trong mỗi lần đó lại có z khối W_i được xử lý lần lượt và tương tác lẫn nhau, làm cho $G=R|V|Y|U|N$ được thay đổi liên tục với mỗi khối W_i và cả khối M_i khi được truyền vào. Điều này giúp tăng hiệu ứng lan truyền phụ thuộc dữ liệu lẫn nhau giữa W_i với W_{i+1}, đồng thời sự tác động qua lại giữa V, Y, U với W_i (tại các bước 6, 8 của thuật toán) đã lan truyền tới giá trị của R (tại bước 7 của thuật toán) giúp việc lựa chọn lại các khóa Q_n trên bảng truy vấn phụ thuộc dữ liệu diễn ra liên tục và ngẫu nhiên trong mỗi lần hàm ***ChangeNVYU*** được gọi ở bước 4 của thuật toán.

Cách sử dụng bộ $G=R|V|Y|U|N$ cũng giống như dùng các thanh ghi A, B, C, D trong hàm băm MD5, tuy nhiên G cho phép linh động tùy biến độ dài thành 160, 192, 224 bit, . dễ dàng bằng việc thêm một biến 32 bit ghép vào G mà không thay đổi bản chất của giải thuật chứ không cố định 128 bit như MD5 Đây chính là đặc tính mềm dẻo đầu ra của ***MBM***.

Kết quả của ***MBM*** cho bản mã C_i, vì vậy khi nhìn từ góc độ một thuật toán mã hóa khóa đối xứng, ***MBM*** đóng vai trò là một giải thuật mã hóa khóa bí mật với Key=$G=R|V|Y|U|N$=160 bit. Sẽ không thể tính toán được C_i nếu chỉ biết M_i và ngược lại nếu có C_i thì sẽ vô cùng khó để tìm ra M_i bởi 160 bit là một không gian quá lớn cho việc tấn công vét cạn hoặc thám mã.

Như vậy, hàm băm đề xuất đã ứng dụng được nhiều ưu điểm từ các phương pháp thiết kế hàm băm khác nhau đó là: cơ chế hàm băm lặp, modulo số học và đặc biệt là được xây dựng trên cơ sở bảng truy vấn phụ thuộc dữ liệu, giúp tăng cường hiệu ứng lan truyền, nâng cao độ bảo mật của hàm băm. Hàm băm đề xuất đã giải quyết được vấn đề giảm thời gian xử lý không phải lập lịch sinh các SubKey mà vẫn tận dụng được những ưu điểm tăng cường bảo mật của chúng. Đồng thời việc ứng dụng cơ chế hàm lặp quay vòng đã giúp thuật toán trở lên đơn giản, nhỏ gọn hơn cùng với tận dụng các toán tử số học làm tốc độ hàm băm được cải thiện đáng kể.



Hình 3. *Mô hình mã hóa khóa bí mật trong MBM*

Bằng cách xử lý dữ liệu đầu vào và độ lớn kết quả băm là tùy biến, hàm băm xây dựng đã khẳng định được tính mềm dẻo khi thao tác. Đây sẽ là tiền đề tốt để có thể đưa vào các ứng dụng trên nhiều hệ thống với những yêu cầu khác nhau.

KẾT QUẢ VÀ BÀN LUẬN

Để có thêm minh chứng đánh giá về hàm băm đề xuất, bài báo đã thực hiện thống kê và tổng hợp các thông số kỹ thuật của một số hàm băm hiện đang được sử dụng rộng rãi hiện nay trong Bảng 1.

Bảng 1. *So sánh độ an toàn của thuật toán băm đề xuất với một số thuật toán băm thông dụng*

Hàm băm	Kích thước bản tin	Kích thước khối	Kích thước từ	Số vòng lặp	Kích thước giá trị băm	Độ an toàn
SHA-1	$< 2^{64}$	512	32	80	160	80
SHA-2(256)	$< 2^{64}$	512	32	64	256	128
SHA-2(384)	$< 2^{128}$	1024	64	80	384	192
SHA-2(512)	$< 2^{128}$	1024	64	80	512	256
MBM (160)					160	80
MBM (192)	$< 2^{128}$	32*z	32	6*z	192	96
.....					...	...

Bảng 2. *Kết quả thực hiện hàm băm đề xuất trên FPGA và so sánh*

Hàm băm	Tài nguyên (CLB)	Tần số (MHz)	Thông lượng (Mb/s)
SHA-1[5]	1004	42 9	119
SHA-2(256) [4]	1060	83	326
SHA-2(384) [4]	1966	74	350
SHA-2(512) [4]	2237	75	480
MBM (160) (đề xuất)	1040	72	1240

Thông qua bảng so sánh cho thấy MBM có ưu điểm chính là cơ chế mềm dẻo trong thiết kế đối với kích thước giá trị băm đầu ra như đã trình bày trên, các thông số so sánh khác cho thấy hàm băm đề xuất tương đương các hàm băm đang được sử dụng phổ biến hiện nay. Tính mềm dẻo cho thấy khả năng phù hợp cho nhiều ứng dụng khác nhau với các mức độ an toàn khác nhau.

Ngoài ra, chúng tôi cũng tiến hành thực hiện hàm băm đề xuất MBM trên FPGA với dòng thiết bị XILINX FPGA Virtex Device (v200pq240) để thực hiện so sánh với một số hàm băm khác trên cùng dòng thiết bị. Kết quả thực hiện được trình bày trong Bảng 2.

Thông qua bảng 2 cho thấy hàm băm đề xuất MBM đạt được tốc độ cao so với SHA-1 và họ SHA-2, có chi phí về tài nguyên ngang bằng với SHA-1 và SHA-2 (256) nhưng thông lượng cao hơn gấp 10 lần SHA-1 và 4 lần SHA-2 (256). Cuối cùng, hàm băm MBM đề xuất vượt trội hơn so với SHA-2 (256, 384, 512) cả về chi phí tài nguyên và thông lượng.

Kết quả nghiên cứu trên cho thấy MBM có thể ứng dụng trong việc đảm bảo tính toàn vẹn của thông tin trong các hệ thống an ninh bằng mật mã của mạng không dây. Nó phù hợp trong việc sử dụng để việc tạo ra bản tin tóm lược hiệu quả và sau đó có thể dùng trong thuật toán chữ ký số để ký trên bản tin tóm lược.

Bài báo là sản phẩm của đề tài có mã số T2017-07-06 được tài trợ bởi kinh phí của Trường Đại học Công nghệ Thông tin và Truyền thông – ĐHTN. Nhóm tác giả chân thành cảm ơn quý Trường.

TÀI LIỆU THAM KHẢO

1. Bart Preneel (2008), *Hash Functions Based on Block Ciphers and Modular Arithmetic*, Katholieke Universiteit Leuven

2 Lars R Knudsen (2008), *Hashing from combination of modular arithmetic and symmetric cryptography*, Dakota

3 Moldovyan N A, Moldovyan A A (2008), *Data-driven Ciphers for Fast Telecommunication Systems*, Auerbach Publications Talor & Francis Group, New York

4 N Sklavos and O. Koufopavlou (2003), On the hardware implementations of the SHA-2 (256, 384, 512) hash functions, *Proceedings of IEEE International Symposium on Circuits & Systems (ISCAS'03)*, Vol V, pp 153-156, Thailand, May 25-28, 2003

5 S Dominikus (2002), "A Hardware Implementation of MD4- Family Hash Algorithms", *proc. of IEEE International Conference on Electronics Circuits and Systems (ICECS'02)*, Vol. III, pp.1143-1146, Croatia

ABSTRACT
A FLEXIBLE HASH FUNCTION

Do Thi Bac[1*], Nguyen Hieu Minh[2]
[1]Universityof information and communication technology,
[2]Academy of Cryptography Techniques

Developing new hash functions which meet the flexibility and secure high speed is an essential requirement in mass communication This article will propose a new hash function exclusively built with a flexible processing mechanism to suit the wireless applications, to be able to sustain strong collisions and to have high integrated efficiency on FPGA as well The hash function is constructed according to a model which uses the repeatedly rotating hash functions and the controlled element. The input parameters of the hash function may be customized by users such as the message length, the key length, the number of repetitions of the rotating function, the magnitude of hash results These are the issues of great interest in this article

Keywords: *hash functions, controlled element, algorithm MBM, switchable data dependent operation, FPGA*

Ngày nhận bài: 29/6/2017; Ngày phản biện: 20/7/2017; Ngày duyệt đăng: 30/9/2017

* *Email: dtbac@ictu edu.vn*