TẤN CÔNG TIÊM LỖI TRÊN AES-128 BẰNG PHƯƠNG PHÁP TẤN CÔNG LỖI VI SAI

Bạch Hồng Quyết[*], **Phạm Văn Tới**[+]
[*] Phòng thí nghiệm trọng điểm ATTT
[+] Phòng thí nghiệm trọng điểm ATTT

***Abstract*—** Thuật toán mã hóa Advanced Encryption Standard (AES) là chuẩn mã hóa dữ liệu rất phổ biến, được sử dụng rộng rãi trên thế giới và được chính phủ Mỹ sử dụng để bảo vệ các dữ liệu tuyệt mật. Tuy nhiên, hiện nay thuật toán AES đã xuất hiện lỗ hổng dễ bị tấn công và những kẻ tấn công bằng nhiều phương pháp khác nhau đang tấn công, khai thác lỗ hổng đó. Trong bài báo này sẽ giới thiệu phương pháp tấn công lỗi vi sai (DFA) trên thuật toán mã hóa AES-128. Mục đích của bài báo là nghiên cứu phương pháp tiêm lỗi vào quá trình mã hóa thiết bị, từ đó có thể xây dựng thuật toán trích xuất khóa K và làm vô hiệu hóa hệ thống an toàn của thiết bị.

***Keywords*—** Thuật toán mã hóa AES, tấn công kênh kề, tấn công tiêm lỗi, phân tích lỗi vi sai.

I. MỞ ĐẦU

Thuật toán mã hóa Advanced Encryption Standard (AES) là chuẩn mã hóa dữ liệu rất phổ biến, được sử dụng rộng rãi trên thế giới và được chính phủ Mỹ sử dụng để bảo vệ các dữ liệu tuyệt mật [1, 2].

AES được phát triển từ cuối những năm 90s để thay thế cho chuẩn mã hóa trước đó là Data Encryption Standard (DES) do IBM tạo ra đầu những năm 70s nhằm khắc phục những khuyết điểm của thuật toán mã hóa đó. Tuy nhiên, hiện nay thuật toán AES đã xuất hiện lỗ hổng dễ bị tấn công và những kẻ tấn công bằng nhiều phương pháp khác nhau đang tấn công, khai thác lỗ hổng đó. Một trong những phương pháp tấn công phổ biến nhất hiện nay là tấn công tiêm lỗi – Fault Injection Attack (FIA), một loại tấn công kênh bên rất mạnh hiện nay [3]. Các cuộc tấn công FIA là mối đe dọa số một cho bất kỳ hệ thống nhúng, chip IC an toàn nào trên thế giới. FIA là những cuộc tấn công vật lý logic với mục đích bỏ qua các cơ chế khởi động an toàn, trích xuất khóa bí mật, phá vỡ bộ đếm chương trình và trích xuất chương trình cơ sở hoặc thao túng bất kỳ tài nguyên bảo mật nào bên trong IC. Các cuộc tấn công như vậy khó thực hiện hơn những cuộc tấn công khác, nhưng đổi lại, nó cho phép bỏ qua các phương pháp bảo vệ hoàn toàn, với tác động nghiêm trọng đến các nhà cung cấp và người dùng.

Có rất nhiều phương pháp tấn công sử dụng FIA như: sự cố nguồn (Glitch power); sự cố xung clock (Glitch clock); thay đổi đột ngột nhiệt độ môi trường (Varying the environmental temperature); sử dụng ánh sáng hoặc lasers; tấn công lỗi vi sai (Different Fault Attack), … [4]. Bản chất của các cuộc tấn công FIA nêu trên đều là giống nhau, và là tiêm lỗi vào trong thiết bị mã hóa trong quá trình tính toán. Quá trình tiêm lỗi phải được tính toán kỹ lưỡng vì phương pháp tiêm lỗi có thể ảnh hưởng đến thiết bị hoặc làm hư hỏng thiết bị. Ví dụ nếu tiêm lỗi bằng cách tăng đột ngột điện áp quá mức hoặc ép xung quá cao, từ trường và điện trường quá mạnh vượt quá giới hạn của các linh kiện điện tử bên trong thiết bị sẽ dẫn đến hư hỏng các linh kiện đó và làm cho thiết bị ngưng hoạt động. Trong bài báo này sẽ giới thiệu phương pháp tấn công lỗi vi sai (DFA) trên thuật toán mã hóa AES-128. Mục đích của bài báo là nghiên cứu phương pháp tiêm lỗi vào quá trình mã hóa thiết bị, từ đó có thể xây dựng thuật toán trích xuất khóa K và làm vô hiệu hóa hệ thống an toàn của thiết bị.

II. TỔNG QUAN VỀ THUẬT TOÁN MẬT MÃ AES

AES là một thuật toán mã hóa khối thay thế hoán vị. Đầu vào của nó và đầu ra bao gồm các chuỗi 128 bit, trong khi khóa mật mã của nó là một chuỗi 128, 192 hoặc 256 bit, tùy thuộc vào mức độ bảo mật cần thiết. Đối với mỗi chiều dài khóa, sẽ có 10, 12 hoặc 14 vòng (tương ứng với khóa 128, 192 và 256-bit), mỗi vòng chúng được tham số hóa bằng khóa vòng (Round key) 128 bit được cung cấp bởi lập lịch khóa (Key scheduling) [5, 6]. Tại mỗi thời điểm, kết quả tạm thời có thể được biểu diễn dưới dạng ma trận 4 x 4 byte, gọi là trạng thái (State). Số vòng lặp ký hiệu là N_r, phụ thuộc vào hai đại lượng N_b và N_k. Vì N_b trong AES có giá trị cố định bằng 4 nên N_r chỉ phụ thuộc vào N_k. Giá trị của N_r tương ứng với ba giá trị của N_k là N_r = 10, 12, 14. Cụ thể, giá trị N_r được xác định bởi:

Type AES	Key length	Number of rounds (N_r)	N_k
AES-128	128	10	4
AES-192	192	12	6
AES-256	256	14	8

Bốn byte trên mỗi cột trong mảng trạng thái state tạo thành 1 từ 32 bit, trong đó số thứ tự của hàng r ($0 \le r \le 3$) cho biết chỉ số của bốn byte trong mỗi từ. Từ định nghĩa state ở trên có thể coi state là mảng một chiều chứa các từ 32 bit:

$$\begin{matrix} S_{00} & S_{01} & S_{02} & S_{03} \\ S_{10} & S_{11} & S_{12} & S_{13} \\ S_{20} & S_{21} & S_{22} & S_{23} \\ S_{30} & S_{31} & S_{32} & S_{33} \end{matrix}$$

Thuật toán mã hóa AES tương đối phức tạp, được mô tả khái quát thành 3 bước như sau:

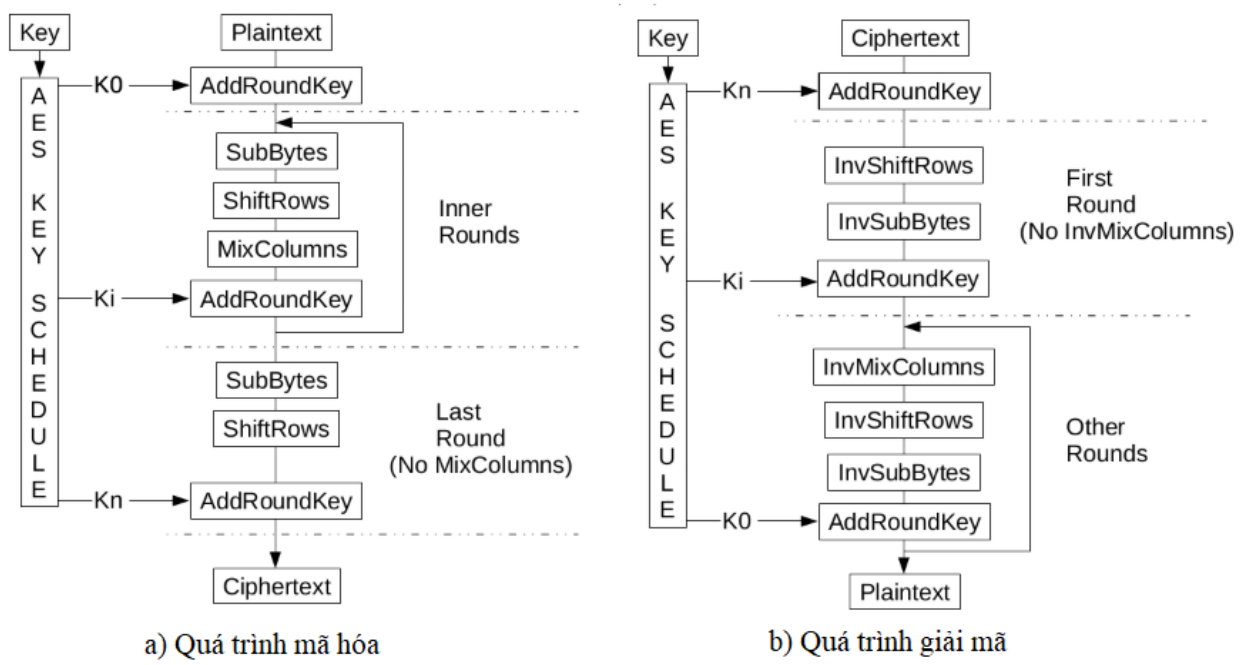
- 1 vòng khởi tạo chỉ bao gồm phép Addroundkey;
- $N_r - 1$ vòng lặp gồm 4 phép biến đổi lần lượt là: SubBytes, ShiftRows, MixColumns, AddRoundKey;
- 1 vòng cuối gồm các phép biến đổi giống vòng lặp và không có phép MixColumns.

Trong giai đoạn khởi tạo AES, chỉ có một bước được thực hiện: AddRoundKey. Kế tiếp, thuật toán sẽ chuyển sang vòng hoạt động đầu tiên và thực hiện bước đầu tiên: SubBytes.

Trong bước Final Round sẽ không có phép MixColumns mà chỉ có các phép: SubBytes, ShiftRows, AddRoundKey [7, 8].

Thuật toán giải mã AES khá giống với thuật toán mã hóa AES về mặt cấu trúc nhưng 4 hàm sử dụng là 4 hàm ngược của quá trình mã hóa.

Một điểm khác biệt nữa trong hai cấu trúc giải mã ngược và giải mã xuôi đó là: Trong giải mã ngược khóa vòng giải mã chính là khóa vòng mã hóa với thứ tự đảo ngược. Còn trong giải mã xuôi thì khóa giải mã ngoài việc đảo ngược thứ tự khóa vòng mã hóa còn phải thực hiện phép InvMixColumns đối với các khóa vòng của vòng lặp giải mã [9]. Sơ đồ khối của thuật toán mã hóa và giải mã hóa AES – 128 được biểu diễn ở hình 1.



Hình 1. Sơ đồ khối thuật toán mã hóa và giải mã hóa AES – 128

Key scheduling xây dựng khóa vòng K trong thuật toán AES bằng cách sử dụng 2 chức năng: khóa vòng (Key Expansion) và lựa chọn phím vòng (Round Key Selection). Thuật toán key AES scheduling được mô tả ở thuật toán 1.

Thuật toán 1. AES key Scheduling

Input *K: secret key,*
L: key length in words,
N_r: *number of rounds*;
Output *W: array containing round keys*
Begin
for i =l **to** $4 \cdot (N_r + 1) - 1$ **do**
if $i \equiv 0 \ mod \ l$ **then**
$W[i] = W[i-l] \oplus SubByte[W[i-1] \ll< 8] \oplus Rcon[\frac{i}{l}]$;
elsif $l = 8 \ and \ i \equiv 4 \ mod \ l$ **then**
$W[i] = W[i-l] \oplus S[W[i-1]]$;
else
$W[i] = W[i-l] \oplus W[i-1]$;
return *W*;
End

Chức năng khóa vòng: Chức năng này tính toán từ khóa AES, khóa vòng có độ dài bằng chiều dài khối tin nhắn nhân với số vòng cộng với *1*, tức là $N_r + 1$. Khóa vòng là một mảng tuyến tính gồm các từ 4 byte và được ký hiệu là $EK[4 \cdot (N_k + 1)]$. Khóa vòng được mô tả ở thuật toán 2.

Thuật toán 2: KeyExpansion

Input: N_k, N_r;
Output *EK*: *array containing round key*;
Begin
$EK \leftarrow temp$;
for *i = 0* **to** $N_k - 1$ **do**
$EK[i] = (K[4 \cdot i], K[4 \cdot i + 1], K[4 \cdot i + 2], K[4 \cdot i + 3])$;
$i = i + +$;
for $i = N_k$ **to** $4 \cdot (N_r + 1)$ **do**
temp = EK[i − 1];
if $i \ mod \ N_k = 0$ **then**
$temp = SubWord(RotWord(temp)) \oplus Rcon[i/N_k]$;
elsif $((N_k > 6) \ and \ (i \ mod \ N_k = 4))$
temp = SubWord(temp);

$EK[i] = EK[i - N_k] \oplus temp;$
$i = i + +;$

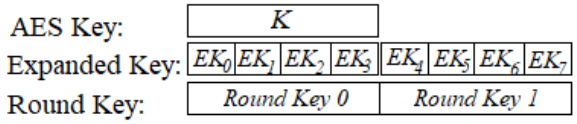
return *EK*;
End

Trong đó:

- *SubWord()* là một hàm áp dụng S-box trong thuật toán mã hóa AES ở mỗi byte của 4 byte đầu vào để tạo ra một từ đầu ra;
- *RotWord ()* là một phép quay tuần hoàn sao cho đầu vào 4 byte (a, b, c,d) tạo ra đầu ra 4 byte (b, c, d, a);
- $Rcon[i] = (x^{i-1}, \{00\}, \{00\}, \{00\})$ với x^{i-1} là giới hạn của *x* trong trường F_{2^8}.

Chức năng lựa chọn khóa vòng: chức năng này trích xuất các khóa vòng 128 bit từ Khóa được mở rộng.

Ví dụ về Key scheduling trong thuật toán mã hóa AES-128:



III. TẤN CÔNG DFA TRÊN THUẬT TOÁN MÃ HÓA AES-128

Các cuộc tấn công phá khóa mã hóa đã được giới thiệu trong [10] dựa trên các lỗi có thể được tạo ra trên các thiết bị thẻ thông minh trong quá trình tính toán thuật toán mã hóa, từ đó nhiều nghiên cứu đã được tiến hành và được thực hiện nghiêm túc hơn. Trong [11] đã thành công trong việc phá vỡ mã hóa RSA CRT với một chữ ký bị lỗi trong cùng một tin nhắn. Sau đó [12] đã cải tiến thuật toán phá khóa bằng cách chỉ tìm một trong các yếu tố của module khóa công khai chỉ sử dụng một chữ ký bị lỗi trong một tin nhắn đã biết. Các cuộc tấn công này là tiền đề cho cuộc tấn công DFA sau này.

DFA là một loại tấn công kênh bên trong lĩnh vực mật mã, cụ thể là phân tích mật mã. Nguyên tắc của cuộc tấn công DFA là tạo ra các lỗi ví dụ như lỗi điện áp, điều kiện môi trường không mong muốn… Các lỗi này được thực hiện trong quá trình triển khai mật mã, điều này sẽ giúp tiết lộ trạng thái bên trong của chúng.

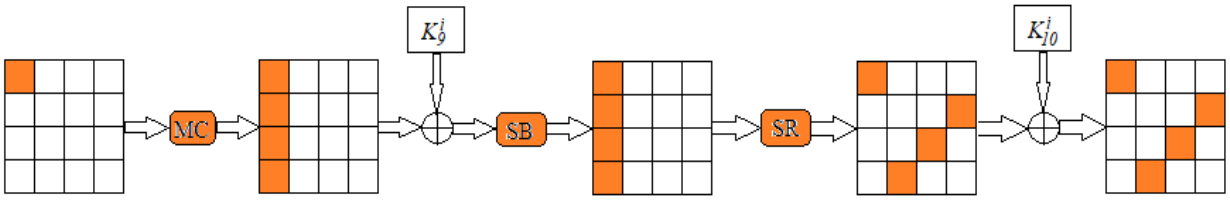
Ví dụ thẻ thông minh chứa bộ xử lý nhúng có thể phải chịu nhiệt độ quá cao, điện áp hoặc dòng điện cung cấp không được hỗ trợ, ép xung quá cao, điện trường hoặc từ trường mạnh có thể ảnh hưởng đến hoạt động của bộ xử lý. Bộ xử lý có thể bắt đầu xuất ra các kết quả không chính xác do lỗi dữ liệu vật lý, điều này có thể giúp kẻ tấn công suy ra được các lệnh mà bộ xử lý đang chạy hoặc trạng thái dữ liệu bên trong của nó [13].

DFA được dùng để kiểm tra tính bảo mật của thuật toán mã hóa DES. Tuy nhiên, ngày nay thuật toán AES đã được thay thế cho DES trong các thuật toán mã hóa. Thật không may, các cuộc tấn công DFA hiện có trên các hệ thống mật mã đối xứng không hoạt động trên AES. Đây là lý do tại sao chúng tôi nghiên cứu, phát triển để tìm cách tấn công AES bằng cách sử dụng DFA.

Trong phần này, chúng tôi sẽ thực hiện một cuộc tấn công DFA với quy mô trên 1 bit của kết quả bản mã tạm thời trước khi kết thúc vòng cuối. Từ đó có thể truy xuất được toàn bộ khóa vòng cuối cùng tức là khóa K trên thuật toán mã hóa AES-128.

Nếu một lỗi gây ra trong một byte của ma trận trạng thái, sau đó được đưa vào vòng thứ 8, thì hoạt động của MixColumn ở vòng cuối sẽ truyền lỗi này đến toàn bộ cột của ma trận trạng thái sau đó. Hoạt động ShiftRow ở đầu vòng sau sẽ chuyển các byte lỗi này sang các cột khác nhau của ma trận trạng thái. Và sau đó, hoạt động MixColumn sẽ lan truyền lỗi đến các byte còn lại.

Quá trình này được mô tả ở hình 2, trong đó hiển thị sự khuếch tán của một lỗi byte gây ra ở đầu vào của vòng thứ 8. Kết quả của thuật toán XOR vi sai ma trận trạng thái sẽ cho ra hai giá trị, một là đúng, hai là sai. Từ đó chúng ta có thể xác định được cơ sở cho một phân tích lỗi vi sai.



Hình 2. Sơ đồ sự khuếch tán lỗi vi sai trên ma trận trạng thái

Để thuận lợi cho quá trình theo dõi và tính toán, chúng tôi sẽ đưa ra một số ký hiệu sau:

1. *S* – bản rõ tương ứng với nó là khóa *K* trên AES;
2. S^i – bản rõ tạm thời lượt thứ *i* và $S_j{}^i$ – byte thứ *j* trên bản rõ S^i;
3. K^i – khóa AES tạm thời lượt thứ *i* và $K_j{}^i$ – byte thứ *j* trên khóa AES K^i;
4. *T* – bản mã chính xác và T_j – byte thứ *j* của bản mã chính xác *T*;
5. *F* – bản mã lỗi và F_j – byte thứ *j* của bản mã lỗi *F*.

Theo định nghĩa, ta có:

$$T = ShiftRows(SubBytes(S^9)) \oplus K^{10} \quad (1)$$

Cũng theo định nghĩa, ta có thể biểu thị $SubByte(S_j^i)$ là kết quả của phép biến đổi byte S_j^i bởi phép $ShiftRows(j)$ tại vị trí thứ *j* trên bản rõ tạm thời. như vậy ta có:

$$T_{ShiftRow(j)} = SubByte(S_j^9) \oplus K_{ShiftRow(j)}^{10}, \quad \forall j \in \{0, \dots, 15\} \quad (2)$$

Nếu chúng ta thực hiện cuộc tấn công DFA e_j trên 1 bit thứ *j* của bản rõ tạm thời S^9 ngay trước vòng cuối của thuật toán, chúng ta sẽ thu được bản mã lỗi *F*:

$$F_{ShiftRow(j)} = SubByte(S_j^9 \oplus e_j) \oplus K_{ShiftRow(j)}^{10} \quad (3)$$

Và:

$$F_{ShiftRow(i)} = SubByte(S_i^9) \oplus K_{ShiftRow(i)}^{10},$$

$$i \in \{0, \dots, 15\} \backslash \{j\} \quad (4)$$

Từ (2) và (4) chúng ta có:

$$T_{ShiftRow(i)} \oplus F_{ShiftRow(i)} = 0, i \in \{0, \dots, 15\} \backslash \{j\} \quad (5)$$

Từ (2) và (3) ta được:

$$T_{ShiftRow(j)} \oplus F_{ShiftRow(j)} = SubByte(S_j^9) \oplus SubByte(S_j^9 \oplus e_j) \quad (6)$$

Từ những công thức trên, chúng ta có thể xây dựng một thuật toán đầy đủ để truy xuất ra được khóa K của thuật toán mã hóa AES-128.

Thuật toán 3. Tìm khóa K của thuật toán mã hóa AES-128.

Input T, S^9, F: *array containing value*
Output K: *array containing key* $AES - 128$
Begin
  For $i = 0$ **to** 15 **do**
    Get $DFA \leftarrow e_j$;
    Get $S_i^9 \oplus e_j$;
    if $S_i^9 \oplus e_j \neq 0$ **then**
      $j = i$;
      Save S_j^9;
      $i = i ++$;
  Get S^9;
  Get K^{10} *from equation (1);*
    Get K *using applying the inverse AES key Scheduling* K^{10};
  return K;
End

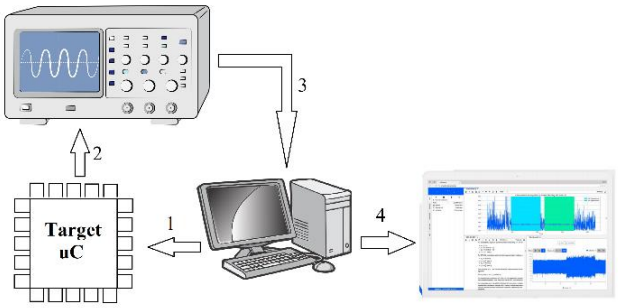
Tấn công DFA hoạt động độc lập trên mỗi bit, vì vậy nếu chúng ta thành công trong việc tạo ra chỉ một lỗi trên một vài byte của bản rõ tạm thời S^9, chúng ta sẽ giảm được số lượng bản mã bị lỗi cần thiết để có được khóa K của thuật toán mã hóa AES-128.

Thuật toán tấn công DFA này cũng áp dụng được trên thuật toán mã hóa AES-192 và AES-256. Trong các trường hợp như vậy, mức độ bảo mật của thuật toán AES-192 sẽ giảm từ 24 byte xuống 8 byte và AES-256 sẽ giảm từ 32 byte xuống 16 byte.

Tấn công DFA này rất mạnh nhưng đòi hỏi phải gây ra lỗi chỉ 1 bit đúng thời điểm là vòng cuối cùng của thuật toán mã hóa AES. Do đó sẽ khó khăn trong việc thực hiện.

IV. ĐÁNH GIÁ THỰC NGHIỆM

Trong phần này, chúng tôi sẽ thực hiện các thí nghiệm để đánh giá độ chính xác của thuật toán đối với kiểu tấn công phân tích lỗi vi sai trên thuật toán mã hóa AES-128. Thực nghiệm được thực hiện trên SoC Kirin 620. Sơ đồ thực nghiệm được mô tả ở hình 3.



Hình 3. Sơ đồ thực nghiệm tấn công phân tích lỗi vi sai trên AES-128

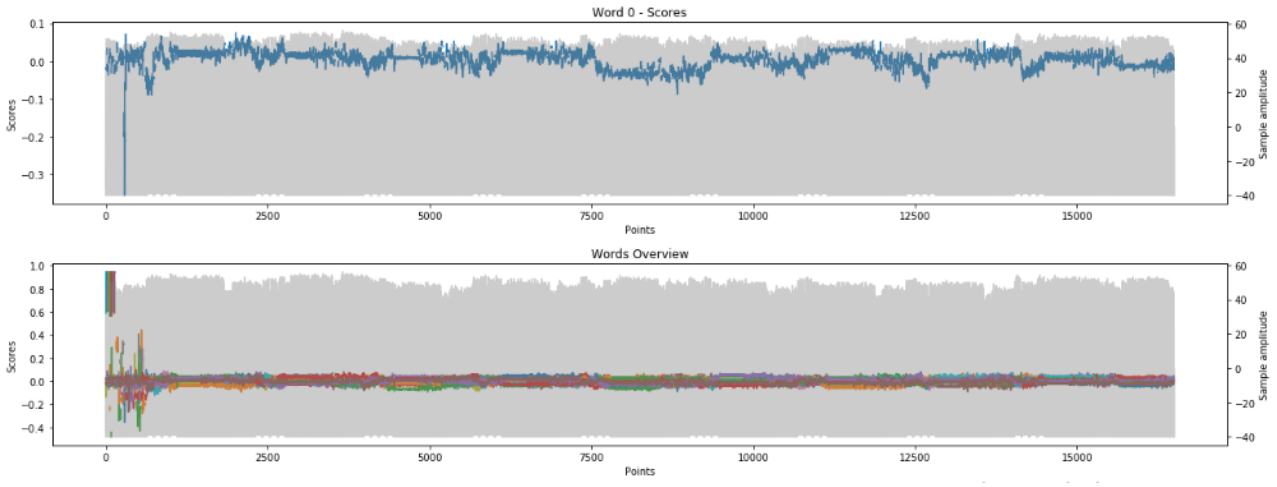
1) Một lệnh được gửi từ máy tính đến thiết bị bị tấn công thông qua giao tiếp nối tiếp báo hiệu cho nó chạy bước AddRoundKey và SubBytes đối với giá trị bản rõ đã biết. Các bước AddRoundKey và SubBytes được chạy 10 lần cho mỗi giá trị bản rõ riêng lẻ. Điều này cho phép oscilloscope có được kết quả trung bình, do đó giúp giảm nhiễu nền và tăng độ chính xác cho kết quả.

2) Oscilloscope được thiết lập để ghi lại từng traces bằng cách sử dụng hàm trung bình (Average function) sao cho giá trị của 10 traces được tính cho mỗi bước duy nhất của AddRoundKey và SubBytes.

3) Khi 10 dấu vết đã được ghi lại bằng oscilloscope, máy tính sẽ truy xuất và lưu kết quả trung bình dưới nhiều định dạng khác nhau trong ổ cứng. Quá trình 1 đến 3 được lặp lại cho mỗi giá trị văn bản gốc từ 00 đến FF.

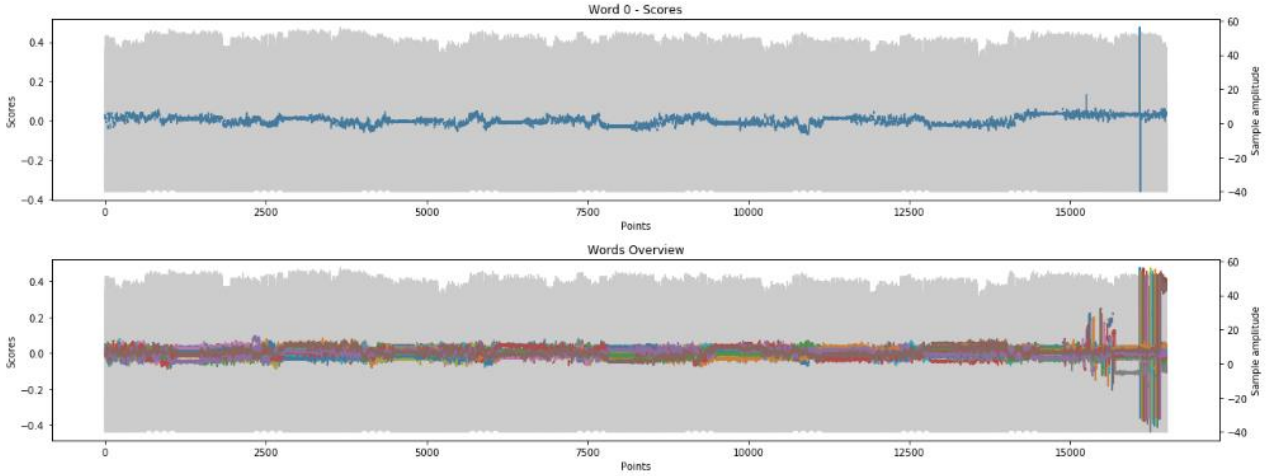
4) Sau khi thu thập dữ liệu giá trị trung bình của 256 traces của bản rõ, được thực hiện ở các bước AddRoundKey và SubBytes trên thiết bị bị tấn công, chương trình có thể phân tích, tính toán kết quả bằng phương pháp phân tích ngoại tuyến. Các kết quả này có thể dự đoán chính xác bằng cách sử dụng Means vi sai.

Sau đây là kết quả thu được từ thực nghiệm.



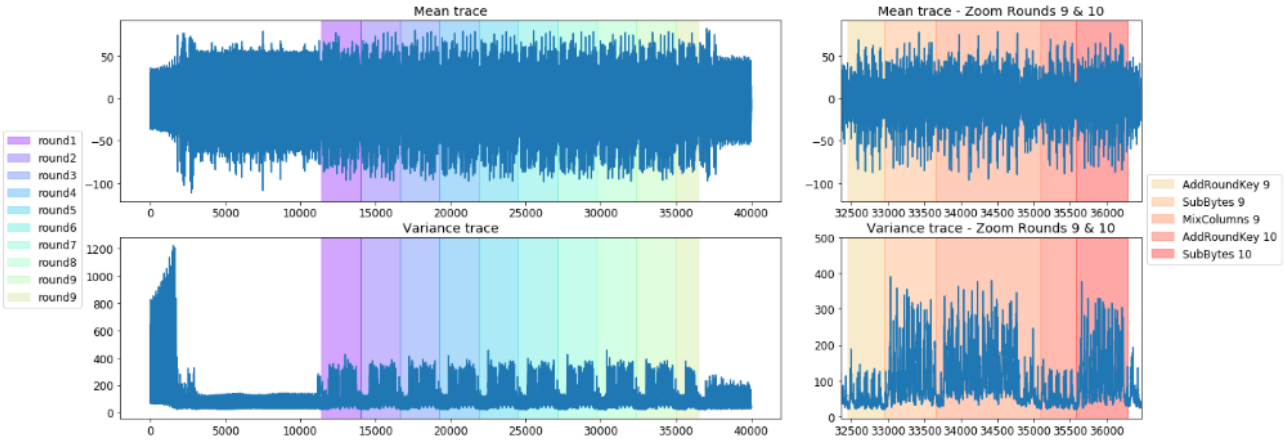
Hình 4. Phát hiện rò rỉ dữ liệu trên bản rõ

Hình 4 mô tả các traces được phát hiện ở đầu vào các bytes của bản rõ. Điều này chứng tỏ bản rõ đã được tải vào thanh ghi hoặc RAM của máy tính. Tương tự như vậy, chúng ta cũng có thể tìm thấy rò rỉ ở bản mã như trong hình 5. Chỉ khác là các traces được phát hiện ở đầu ra các bytes của bản mã.



Hình 5. Phát hiện rò rỉ dữ liệu trên bản mã

Hình 6 mô tả các Means trace và Variance trace của từng bước AddRoundKey, SubBytes, ShiftRows, MixColumns trong quá trình mã hóa. Từ đó có thể tấn công chính xác vào đúng vị trí, thời điểm và thu được minimum các giá trị bản lỗi cần thiết để truy xuất ra khóa *K*.



Hình 6. Means trace và Variance trace của thuật toán mã hóa AES-128

Sử dụng thuật toán đã trình bày ở trên, chúng ta thu được khóa *K* ban đầu.

```
# First retrieve the found key corresponding to the last round
FoundK9 = A.scores.argmax(0).squeeze()

# Then recover the master key
MasterKey = scared.aes.inv_key_schedule(FoundK10.astype('uint8'))[0, 0]

print("The recovered key is: ", MasterKey)
print("The secret key is:    ", ths.key[0])
```

```
The recovered key is:  [143   3 231 122 236 102  88  10  19  43 210  59 245  99  78  15]
The secret key is:     [143   3 231 122 236 102  88  10  19  43 210  59 245  99  78  15]
```

V. KẾT LUẬN

Trong bài báo này đã giới thiệu một phương pháp tấn công DFA trên thuật toán mã hóa AES-128. Trong bài báo này cũng chỉ ra phương pháp để có thể truy xuất được khóa *K* của thuật toán mã hóa AES-128 bằng cách sử dụng các thuật toán đã nêu ở trên. Vì cuộc tấn công này rất mạnh, sử dụng ít bản lỗi để có thể lấy được khóa *K* nên tiêu tốn ít thời gian để thực hiện. Nhược điểm của phương pháp này là cách thực hiện tấn công. Cuộc tấn công phải đúng thời điểm và đúng vị trí, từ đó mới có thể thu được những bản lỗi chính xác, phù hợp, làm dữ liệu để phân tích tìm khóa *K*.

Trong tương lai, chúng tôi sẽ phân tích và áp dụng đề xuất của bài báo này để làm cơ sở để tối ưu hóa kỹ thuật tấn công kênh kề bằng phương pháp phân tích lỗi vi sai, từ đó có thể phát triển một chương trình truy xuất ra khóa K của thuật toán mã hóa AES-128 và hơn nữa trên AES-192 và AES-256.

REFERENCE

[1] J. Bl¨omer, J. G. Merchan, and V. Krummel. Provably Secure Masking of AES. In SAC 2004, vol. 3357 of LNCS, pages 69–83. Springer, 2004.

[2] Heron S. Advanced encryption standard (AES). Netw Secur. 2009; 2009(12):8–12. DOI:10.1016/ S1353-4858(10)70006-4.

[3] A. Berzati, C. Canovas, and L. Goubin. (In) security Against Fault Injection Attacks for CRT-RSA Implementations. In L.

Breveglieri, S. Gueron, I. Koren, D. Naccache, and J.-P. Seifert, editors, Fault Diagnosis and Tolerance in Cryptography – FDTC 2008, pages 101–107. IEEE Computer Society, 2008.

[4] V. Lomne, T. Roche, and A. Thillard. On the Need ´ of Randomness in Fault Attack Countermeasures – Application to AES. In G. Bertoni and B. Gierlichs, editors, Fault Diagnosis and Tolerance in Cryptography – FDTC 2012, pages 85–94. IEEE Computer Society, 2012.

[5] S. Chari, C. Jutla, J. Rao, and P. Rohatgi. A Cautionary Note Regarding Evaluation of AES Candidates on Smart-Cards. In AES 2, March 1999.

[6] J. Goli´c and C. Tymen. Multiplicative Masking and Power Analysis of AES. In CHES 2002, vol. 2523 of LNCS, pages 198–212. Springer, 2002.

[7] E. Oswald, S. Mangard, N. Pramstaller, and V. Rijmen. A Side-Channel Analysis Resistant Description of the AES S-box. In FSE 2005, vol. 3557 of LNCS, pages 413–423. Springer, 2005.

[8] E. Oswald and K. Schramm. An Efficient Masking Scheme for AES Software Implementations. In WISA 2005, vol. 3786 of LNCS, pages 292–305. Springer, 2006.

[9] V. Rijmen. Efficient Implementation of the Rijndael S-box, 2000. Available at http://www.esat.kuleuwen.ac.be/~rijmen/rijndael/sbox.pdf.

[10] D. Boneh, R. A. Demillo and R. J. Lipton, On the Importance of Checking Cryptographic Protocols for Faults, Springer, Lecture Notes in Computer Science vol. 1233, Advances in Cryptology, proceedings of EUROCRYPT'97, pp. 37-51, 1997.

[11] D. Boneh, R. A. DeMillo and R. J. Lipton, On the Importance of Eliminating Errors in Cryptographic Computations, Springer, Journal of Cryptology 14(2), pp. 101-120, 2001.

[12] M. Joye, A. K. Lenstra and J.-J. Quisquater, Chinese Remaindering Based Cryptosystems in the Presence of Faults, Springer, Journal of Cryptology 12(4), pp. 241-246, 1999.

THE FAULT INJECTION ATTACK ON THE AES-128 BY THE METHOD OF DIFERENCE FAULT ATTACK

Abstract: Andvanced Encryption Standard is the popular data standardization, used globally and the US government uses to secure extreme data. However, the algorithm of AES currently provides holes for hackers to attack and exploit with different methods. This article will introduce the DFA based on the encryption algorithm AES-128. Its purpose is to analyze the method of injecting errors in the device encryption process in order to build the K extraction algorithm and disable device's security system.

Keywords— AES encryption algorithm, Side-channel attack, Fault injection attack, Differential analysis.

ThS. Bạch Hồng Quyết

Cơ quan: Phòng thí nghiệm trọng điểm ATTT

Email: bachhongquyet@gmail.com

Quá trình đào tạo: Nhận bằng kỹ sư tại trường Học viện Kỹ thuật Quân sự năm 2001; nhận bằng thạc sỹ tại trường Học viện Kỹ thuật Quân sự năm 2009. Hướng nghiên cứu hiện nay: An ninh mạng, bảo mật phần cứng.

TS. Phạm Văn Tới

Cơ quan: Phòng thí nghiệm trọng điểm ATTT

Email:phamvantoi141090@gmail.com

Quá trình đào tạo: Nhận bằng kỹ sư tại trường ĐHQG Kỹ thuật Vô tuyến Ryazan, LB Nga; Nhận bằng tiến sỹ tại trường ĐHQG Kỹ thuật Vô tuyến Ryazan, LB Nga. Hướng nghiên cứu hiện nay: An ninh mạng, bảo mật phần cứng.