Tạp chí An toàn thông tin, Ban Cơ yếu Chính phủ

GIỚI THIỆU VỀ THUẬT TOÁN MÃ HÓA CÓ XÁC THỰC HẠNG NHẸ ACORN TRONG CUỘC THI CAESAR

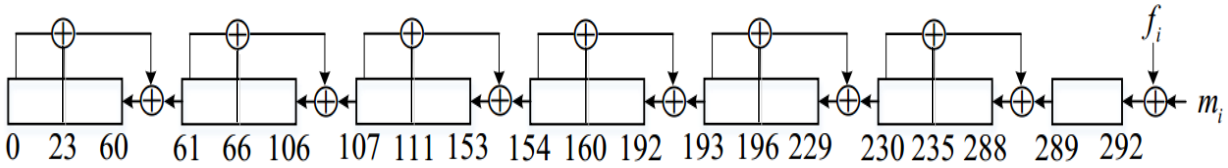
Đinh Tiến Thành, Nông Ngọc Hoàng

Tóm tắt: ACORN [4] là thuật toán mã hóa có xác thực hạng nhẹ. Thuật toán có cấu trúc mã dòng và được thiết kế dựa trên các thanh ghi dịch phản hồi tuyến tính. Bài báo này sẽ trình bày tổng quan về thuật toán, một số đặc trưng an toàn và hiệu suất thực thi của thuật toán ACORN.

I. GIỚI THIỆU CHUNG

Một hướng đi mới trong khoa học mật mã hiện nay là xây dựng các thuật toán mã hóa có xác thực nhằm cung cấp tính bí mật, xác thực và toàn vẹn cho thông báo trong một bước duy nhất, loại bỏ sự không cần thiết của một thuật toán MAC riêng biệt, có lợi về hiệu suất trên các nền tảng tham chiếu khác nhau như FPGA, vi điều khiển [5].

Thuật toán mã hóa có xác thực hạng nhẹ ACORN được giới thiệu lần đầu tiên vào năm 2014 bởi Hongjun Wu, nhà mật mã người Singapore. ACORN là ứng viên tại vòng chung kết của cuộc thi CAESAR [1]. Thuật toán được thiết kế dựa trên cấu trúc mã dòng. Trạng thái của ACORN được tạo thành từ 6 thanh ghi dịch phản hồi tuyến tính (LFSR) có độ dài lần lượt là 61, 46, 47, 39, 37 và 59 kèm 4-bit đệm được mô tả như Hình 1. Như vậy, trạng thái của thuật toán có kích thước là 293 bit. Kích thước khóa và véc-tơ khởi tạo là 128 bit. Thuật toán được thiết kế để mã hóa các bản rõ P có độ dài nhỏ hơn 2^{64} bit cùng với *adlen* bit dữ liệu liên kết $(0 \le adlen < 2^{64})$. Đầu ra của thuật toán là bản mã C có độ dài nhỏ hơn 2^{64} bit và một thẻ xác thực T có độ dài t $(64\text{bit} \le t \le 128\text{bit})$. ACORN cung cấp cả tính bí mật và xác thực cho bản rõ đầu vào.



Hình 1. Tổ chức các thanh ghi dịch trong thuật toán ACORN

Ba phiên bản của thuật toán là ACORNv1, ACORNv2, ACORNv3 lần lượt được giới thiệu tại ba vòng của cuộc thi CAESAR. Bài báo này sẽ phân tích ACORNv3 sử dụng khóa 128 bit, phiên bản tại vòng chung kết của cuộc thi CAESAR và được cải tiến từ hai phiên bản trước. Sau đây bài báo sử dụng tên gọi ACORN để chỉ phiên bản ACORNv3.

Bố cục: Sau mục giới thiệu chung, mục II trình bày về thuật toán ACORN. Mục III trình bày về một số đặc trưng an toàn trong thiết kế của thuật toán ACORN. Mục IV đánh giá hiệu suất thực thi của thuật toán ACORN và cuối cùng là kết luận.

II. THUẬT TOÁN MÃ HÓA CÓ XÁC THỰC ACORN

2.1. Định nghĩa các ký hiệu sử dụng trong thuật toán

Kí hiệu	Chú thích
$\oplus$	Phép toán XOR 2 bit.
&	Phép toán AND 2 bit.
~	Phép toán NOT bit.
$\|$	Phép ghép bit.
AD	Dữ liệu liên kết (Associated Data).
ad_i	Bit thứ i của dữ liệu liên kết.
$adlen$	Chiều dài bit của dữ liệu liên kết, với $0 \le adlen < 2^{64}$.
C	Bản mã.
c_i	Bit thứ i của bản mã.
ca_i, cb_i	2 bit điều khiển tại bước thứ i.
IV_{128}	Véc-tơ khởi tạo 128 bit của ACORN.
$IV_{128,i}$	Bit thứ i của véc-tơ khởi tạo 128 bit.
K_{128}	Khóa 128 bit của ACORN.
$K_{128,i}$	Bit thứ i của khóa 128 bit.
ks_i	Bit khóa dòng tạo ra tại bước thứ i.
m_i	Bit dữ liệu thứ i.
P	Bản rõ.
p_i	Bit thứ i của bản rõ.
$plen$	Chiều dài bit của bản rõ, với $0 \le plen < 2^{64}$.
S_i	Trạng thái tại bước thứ i.
$S_{i,j}$	Bit thứ j của trạng thái S_i, với $0 \le j \le 292$.
T	Thẻ xác thực.
t	Chiều dài bit của thẻ xác thực, với $64 \le t \le 128$.

2.2. Các hàm sử dụng trong ACORN

ACORN sử dụng 3 hàm để thực hiện các quá trình hoạt động: một hàm tạo dòng khóa từ các bit trạng thái, một hàm phản hồi phi tuyến và một hàm cập nhật trạng thái của thuật toán.

a. Hàm tạo dòng bit khóa *(KSG)*

Hàm này được sử dụng để tạo ra một bit khóa dòng tại mỗi bước của thuật toán. Đầu vào của hàm là 8 bit trạng thái. Dòng bit khóa được tính toán theo công thức sau: $ks_i = KSG(S_i)$:

$$ks_i = S_{i,12} \oplus S_{i,154} \oplus maj(S_{i,235}, S_{i,61}, S_{i,193}) \oplus ch(S_{i,230}, S_{i,111}, S_{i,66})$$

với *maj(x,y,z)* và *ch(x,y,z)* là hai hàm Boolean được định nghĩa như sau:

+) $maj(x, y, z) = (x \,\&\, y) \oplus (x \,\&\, z) \oplus (y \,\&\, z)$;

+) $ch(x, y, z) = (x \,\&\, y) \oplus ((\sim x) \,\&\, z)$;

b. Hàm phản hồi *(FBK)*

Hàm phản hồi sử dụng các bit lấy từ trạng thái hiện tại S_i, dòng bit khóa ks_i, các bit điều khiển ca_i, cb_i làm đầu vào và trả về bit phản hồi ở mỗi bước. Các bit phản hồi được tính toán theo công thức sau: $f_i = FBK(S_i, ca_i, cb_i, ks_i)$:

$$f_i = S_{i,0} \oplus (\sim S_{i,107}) \oplus maj(S_{i,244}, S_{i,23}, S_{i,160}) \oplus (ca_i \,\&\, S_{i,196}) \oplus (cb_i \,\&\, ks_i)$$

c. Hàm cập nhật trạng thái *(StateUpdate)*

Đầu vào của hàm cập nhật trạng thái là trạng thái hiện tại S_i, bit dữ liệu m_i, hai bit điều khiển ca_i, cb_i để tạo ra trạng thái S_{i+1} tiếp theo cho thuật toán theo các bước sau:

$$S_{i+1} = StateUpdate(S_i, m_i, ca_i, cb_i)$$

Bước 1: Cập nhật trạng thái sử dụng của 6 thanh ghi dịch

$S_{i,289} = S_{i,289} \oplus S_{i,235} \oplus S_{i,230}$;

$S_{i,230} = S_{i,230} \oplus S_{i,196} \oplus S_{i,193}$;

$S_{i,193} = S_{i,193} \oplus S_{i,160} \oplus S_{i,154}$;

$S_{i,154} = S_{i,154} \oplus S_{i,111} \oplus S_{i,107}$;

$S_{i,107} = S_{i,107} \oplus S_{i,66} \oplus S_{i,61}$;

$S_{i,61} = S_{i,61} \oplus S_{i,23} \oplus S_{i,0}$;

Bước 2: Tạo bit khóa dòng

$ks_i = KSG(S_i)$

Bước 3: Tạo bit phản hồi phi tuyến

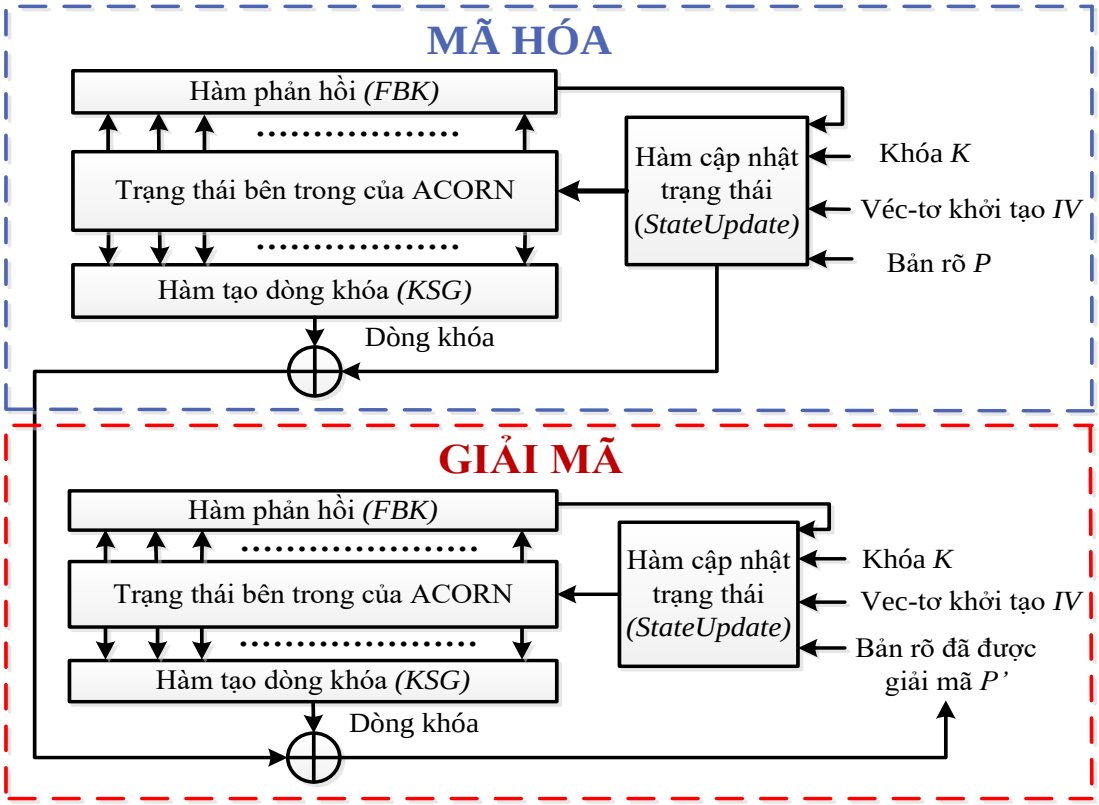
$f_i = FBK(S_i, ca_i, cb_i, ks_i)$

Bước 4: Dịch thanh ghi dịch 293-bit với bit phản hồi f_i:

for j=0 to 291 do $S_{i+1,j} = S_{i,j+1}$;

$S_{i+1,292} = f_i \oplus m_i$;

2.3. Quá trình thực hiện của thuật toán ACORN



Hình 2. Hoạt động của thuật toán ACORN

Các hoạt động trong thuật toán ACORN chia thành bốn quá trình: quá trình khởi tạo, quá trình mã hóa, quá trình tạo thẻ xác thực và quá trình giải mã và xác thực. Hình 2 mô tả hoạt động của ACORN.

a. Quá trình khởi tạo

Quá trình khởi tạo sử dụng khóa K 128 bit, véc-tơ khởi tạo IV 128 bit và dữ liệu liên kết làm đầu vào cho việc khởi tạo trạng thái của ACORN. Quá trình khởi tạo được thực hiện qua hai giai đoạn: giai đoạn nạp khóa, véc-tơ khởi tạo và giai đoạn nạp dữ liệu liên kết.

Giai đoạn nạp khóa và véc-tơ khởi tạo gồm 1792 bước:

1. Khởi tạo trạng thái $S_{-1792} = 0$.
2. *for i=0 to 127 do* $m_{-1792+i} = K_{128,i}$;
   for i=0 to 127 do $m_{-1792+128+i} = IV_{128,i}$;
   for i=0 do $m_{-1792+256} = K_{128,i \bmod 128} \oplus 1$;
   for i=0 to 1535 do $m_{-1792+256+i} = K_{128,i \bmod 128}$;
3. *for i =0 to 1791 do* $ca_{-1792+i} = 1$;
   for i =0 to 1791 do $cb_{-1792+i} = 1$;
4. *for i = -1792 to -1 do* $S_{i+1} = StateUpdate(S_i, m_i, ca_i, cb_i)$

Giai đoạn nạp dữ liệu liên kết:

1. *for i=0 to (adlen – 1) do* $m_i = ad_i$;

$m_{adlen} = 1;$

for i=1 to 255 do $m_{adlen+1} = 0;$

2. *for i=0 to (adlen + 127) do* $ca_i = 1;$

for i=(adlen +128) to (adlen+255) do $ca_i = 0;$

for i=0 to (adlen + 255) do $cb_i = 1;$

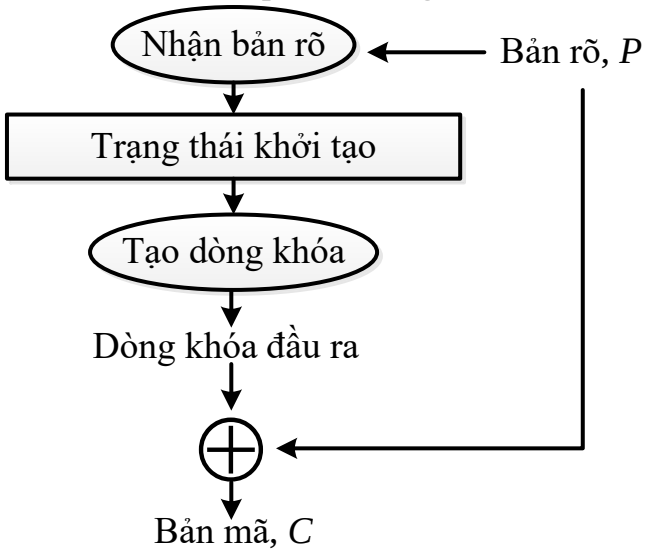
3. *for i=0 to (adlen + 255)*

$S_{i+1} = StateUpdate(S_i, m_i, ca_i, cb_i);$

Nếu không có dữ liệu liên kết, ta vẫn cần thực hiện 256 bước để nạp các bit đệm. Trong quá trình khởi tạo, dòng bit được sử dụng để cập nhật trạng thái khi $cb_i = 1$. Đặc tính của thuật toán được thay đổi qua 128 bước (khi giá trị của ca_i được đặt bằng 0 qua 128 bước) để tách dữ liệu liên kết từ bản rõ/bản mã. Ở đây việc tách dữ liệu từ bản rõ/bản mã có nghĩa là ngăn việc sử dụng một phần dữ liệu liên kết như là bản mã hoặc sử dụng một phần bản mã như là dữ liệu liên kết.

b. Quá trình mã hóa

Quá trình mã hóa sử dụng bản rõ *P* làm đầu vào và tính toán đầu ra là bản mã *C*. Bản rõ *P* được nạp vào trong các ô nhớ của thanh ghi bằng hàm cập nhật trạng thái. Trong hàm cập nhật trạng thái, các bit bản rõ *P* sẽ được sử dụng như các bit dữ liệu m_i trong suốt quá trình mã hóa. Bản mã *C* được tính toán bằng phép XOR giữa các bit khóa được sinh ra từ hàm cập nhật trạng thái với bản rõ đầu vào *P*.



Hình 3. Quá trình mã hóa

1. *for i=0 to (plen – 1) do* $m_{adlen+256+i} = p_i;$

$m_{adlen+256+plen} = 1;$

for i=1 to 255 do $m_{adlen+256+plen+i} = 0;$

2. *for i=(adlen + 256) to (adlen + plen + 383) do* $ca_i = 1;$

for i=(adlen + plen + 384) to (adlen + plen + 511) do $ca_i = 0;$

for i=(adlen + 256) to (adlen + plen + 511) do $cb_i = 0;$

3. *for i=(adlen + 256) to (adlen + plen + 511) do*

$S_{i+1} = StateUpdate(S_i, m_i, ca_i, cb_i);$

$c_{i-adlen-256} = p_{i-adlen-256} \oplus ks_i;$

end for;

c. Quá trình tạo thẻ xác thực

Sau khi tất cả bản rõ đã được mã hóa, thuật toán ACORN thực hiện quá trình tạo thẻ xác thực. Qua 768 bước, thuật toán tạo ra thẻ xác thực T là t bit cuối cùng của dòng khóa với $64 \le t \le 128$.

1. *for i=0 to 767 do* $m_{adlen+plen+512+i} = 0$

2. *for i=(adlen + plen + 512) to (adlen + plen + 1279) do* $ca_i = 1;$

for i= (adlen + plen + 512) to (adlen + plen + 1279) do $cb_i = 1;$

3. *for i= (adlen + plen + 512) to (adlen + plen + 1279) do*

$S_{i+1} = StateUpdate(S_i, m_i, ca_i, cb_i);$

Thẻ xác thực T là t bit cuối cùng của dòng khóa ks_i được sinh ra trong hàm cập nhật trạng thái, tức là:

$T = ks_{adlen+plen+1279-t+1} \| ks_{adlen+plen+1279-t+2} \| \ldots \| ks_{adlen+plen+1279}$

d. Quá trình giải mã và xác thực

Để thực hiện giải mã, đầu tiên, bên giải mã tiến hành khởi tạo trạng thái với khóa K, véc-tơ khởi tạo IV, dữ liệu liên kết AD. Trong quá trình giải mã, các bit dòng khóa ks_i được tạo tại mỗi bước của thuật toán. Các bit này được XOR với bản mã C_i tại thời điểm tương ứng để giải mã tạo ra các bit bản rõ. Các bit bản rõ đã giải mã được cập nhật liên tục vào trạng thái để thực hiện giải mã các bit tiếp theo. Quá trình được lặp lại đến khi các bit bản mã được xử lý xong.

Sau khi xử lý xong các bit bản mã, thẻ xác thực được tạo ra ở phía người nhận tương tự cách thẻ xác thực tạo ra ở phía người gửi. Cuối cùng, thực hiện so sánh thẻ xác thực giữa bên gửi và bên nhận. Nếu như quá trình xác thực là thành công thì bản rõ giải mã được là giống so với bản rõ bên phía người gửi.

III. MỘT SỐ ĐẶC TRƯNG AN TOÀN TRONG THIẾT KẾ CỦA THUẬT TOÁN ACORN

3.1. Mục tiêu an toàn của ACORN

Thuật toán ACORN được thiết kế để đảm bảo mức độ an toàn 128 bit trong cả tính bí mật và xác thực.

Bảng 1. Mục tiêu an toàn của thuật toán ACORN

	Mã hóa	Xác thực
ACORN	128 bit	128 bit

Để đảm bảo tính an toàn của thuật toán ACORN theo đúng mục tiêu, việc sử dụng khóa, véc-tơ khởi tạo và thẻ xác thực cần phải thực hiện các yêu cầu sau:

- Khóa phải được tạo ngẫu nhiên và phân phối tới người gửi và người nhận một cách bí mật.

- Mỗi cặp khóa và véc-tơ khởi tạo (hoặc giá trị Nonce) chỉ được sử dụng để mã hóa một thông báo.

- Nếu việc xác thực thẻ là thất bại, không nên đưa ra bản rõ đã được giải mã thất bại và thẻ xác thực sai.

3.2. Sự an toàn của quá trình khởi tạo

Quá trình khởi tạo có thể bị tấn công bởi việc phân tích sự liên quan giữa véc-tơ khởi tạo và dòng khóa. Trong thuật toán ACORN, véc-tơ khởi tạo được biến đổi qua 1792 bước trước khi tác động đến bản mã. Theo [4], tác giả H. Wu đã tiến hành phân tích sự lan truyền sai khác trong trạng thái dựa trên sự khác biệt của véc-tơ khởi tạo đầu vào, tính toán xác suất lượng sai và nhận thấy như sau:

Trong quá trình khởi tạo, $f_i \oplus ks_i$ được sử dụng như là bit phản hồi phi tuyến. Trong thử nghiệm đầu tiên, giả sử rằng sau khi véc-tơ khởi tạo được nạp vào trạng thái, có sự khác biệt tại $S_{i,292}$ tại thời điểm bắt đầu của bước thứ i và tất cả các bit trạng thái tại bước này là bí mật. Tác giả ước tính xác suất lượng sai là 2^{-227} sau 400 bước.

Để cải thiện xác suất lượng sai, tiến hành kiểm tra sự sai khác tại $S_{i,230}$, $S_{i,235}$ và $S_{i,289}$. Xác suất lượng sai là 2^{-293} sau 400 bước, không hơn đáng kể so với thử nghiệm đầu tiên. Một ví dụ khác, kiểm tra sự khác biệt tại $S_{i,61}$, $S_{i,66}$ và $S_{i,107}$. Xác suất lượng sai là 2^{-289} sau 400 bước, cũng không lớn hơn đáng kể so với thử nghiệm ban đầu.

Từ những thử nghiệm trên, có thể thấy rằng quá trình khởi tạo của ACORN có sự an toàn chống lại tấn công lượng sai.

3.3. Sự an toàn của quá trình mã hóa

Do ACORN là một thuật toán mã dòng với trạng thái lớn được cập nhật liên tục, nên các tấn công lên mã khối không thể áp dụng trực tiếp lên quá trình mã hóa của ACORN. Các tấn công lên quá trình mã hóa của ACORN hầu hết đều có mục đích là tìm ra trạng thái bí mật của thuật toán.

Nếu véc-tơ khởi tạo được sử dụng một lần cho mỗi khóa, thì việc thực hiện tấn công lượng sai và tấn công tuyến tính để tìm ra trạng thái bí mật của thuật toán là không thể vì trạng thái của ACORN được cập nhật bằng hàm cập nhật phi tuyến, mỗi bit đều tác động đến cả trạng thái.

Các tấn công truyền thống lên mã dòng như tấn công tương quan nhanh (fast correlation attack [9]), tấn công đại số nhanh (fast algebraic attack [7]) khai thác đặc tính tuyến tính trong hàm cập nhật trạng thái cũng không thực hiện được do hàm cập nhật trang thái được thực hiện theo phương pháp phi tuyến, tất cả các bit của trạng thái tác động đến cả trạng thái.

Một tấn công khác rất hiệu quả với mã dòng trạng thái nhỏ là tấn công cân bằng thời gian - dữ liệu - bộ nhớ [8]. Tuy nhiên, nếu trạng thái của thuật toán mã dòng lớn hơn 2 lần kích thước khóa thì có thể kháng được tấn công cân này. Ví dụ, với 1 thuật toán mã dòng với *n*-bit khóa và *2n*-bit trạng thái, tấn công cân bằng thời

gian - dữ liệu - bộ nhớ yêu cầu 2^n dữ liệu và 2^n bộ nhớ để phá vỡ thuật toán với thời gian 2^n. Kích thước trạng thái của ACORN là 293 bit, lớn hơn 2 lần so với kích thước khóa (128 bit), nên ACORN là an toàn trước tấn công cân bằng thời gian - dữ liệu - bộ nhớ.

Ngoài ra, tấn công dự đoán và xác định [10] có thể được áp dụng để phân tích mã dòng dựa trên tính phi tuyến hoặc tuyến tính của hàm cập nhật trạng thái. Trong tấn công dự đoán và xác định, một phần của thông tin bí mật được dự đoán để tìm ra thêm thông tin về trạng thái.

Trong thuật toán ACORN, mỗi bit của dòng khóa được tính bởi 1 phương trình phi tuyến liên quan đến các bit trạng thái:

$$ks_i = S_{i,12} \oplus S_{i,154} \oplus maj(S_{i,235}, S_{i,61}, S_{i,193}) \oplus ch(S_{i,230}, S_{i,111}, S_{i,66})$$

Có thể dự đoán rằng hàm $maj(S_{i,235}, S_{i,61}, S_{i,193})$ luôn cho đầu ra $S_{i,61}$ (với xác suất $3/4$) và hàm $ch(S_{i,230}, S_{i,111}, S_{i,66})$ luôn cho đầu ra $S_{i,66}$ (giả sử $S_{i,230} = 0$), do đó đạt được phương trình tuyến tính sau:

$$ks_i = S_{i,12} \oplus S_{i,154} \oplus S_{i,61} \oplus S_{i,66}$$

Xác suất cho hàm tuyến tính trên là $3/4 \times 1/2 = 3/8$. Có thể đạt được 2 phương trình tuyến tính liên quan đến các bit trạng thái (phương trình tuyến tính khác là $S_{i,230} = 0$). Cần 293 phương trình tuyến tính để tìm ra được trạng thái, và xác suất để đạt được 293 phương trình tuyến tính là $(3/8)^{292/2} \approx 2^{-206}$. Kết quả phân tích trên đây chỉ ra rằng ACORN là an toàn trước tấn công dự đoán và xác định.

3.4. Sự an toàn của quá trình xác thực

Một cách tiếp cận để tấn công tính xác thực của ACORN là đưa một sai khác vào trạng thái trong bằng cách sửa đổi bản mã hoặc dữ liệu tương ứng. Một đặc điểm quan trọng của ACORN là thiết kế liền kề của 6 thanh ghi dịch LFSR. Thiết kế này đảm bảo rằng khi một bit sai khác được nạp vào trạng thái thì sẽ tác động đến rất nhiều bit trong trạng thái trước khi chúng bị loại bỏ.

Tại mỗi bước, có 3 hàm phi tuyến được sử dụng trong thuật toán: 2 hàm *maj* và 1 hàm *ch*. Tính chất lượng sai của hàm *maj(x,y,z)* được hiểu là nếu có 1 hay 2 bit đầu vào khác nhau thì sự sai khác đầu ra là 1 với xác suất là 0.5; nếu đầu vào có 3 bit khác nhau, thì sự sai khác đầu ra là 1 với xác suất là 1. Tính chất lượng sai của hàm *ch(x,y,z)* được hiểu là nếu đầu vào có sự khác nhau của cả *y* và *z* thì sự sai khác đầu ra là 1 với xác suất 1; ngược lại, nếu có bất kỳ sự khác biệt ở đầu vào, đầu ra khác nhau là 1 với xác suất 0.5.

Với sự sai khác đầu vào của các thanh ghi và tính chất lượng sai của các hàm phi tuyến, xác suất để loại bỏ sự khác biệt trong trạng thái là 2^{-181}. Với xác

suất lượng sai này, ACORN có thể cung cấp sự an toàn tương đương với mã xác thực thông báo 128-bit.

Đối với tấn công giả mạo, một kẻ tấn công chỉ có thể sửa đổi bản mã hoặc dữ liệu liên kết. Nếu véc-tơ khởi tạo được sử dụng lại, kẻ tấn công có thể sửa đổi bản rõ khiến cho 2 bản rõ khác nhau có thể có chung 1 thẻ xác thực. Với ACORN, tỷ lệ thành công của tấn công giả mạo với véc-tơ khởi tạo được dùng lại là 2^{-120}. Tuy nhiên, nếu véc-tơ khởi tạo được dùng lại trong thuật toán thì sẽ có nhiều tấn công khôi phục trạng thái nghiêm trọng hơn và thông báo có thể bị làm giả một khi trạng thái đã bị tìm ra. Nên loại tấn công giả mạo này không gây ra nhiều nguy hiểm cho ACORN.

3.5. Một số tấn công lên ACORN

Thuật toán ACORN được thiết kế theo cấu trúc mã dòng và là ứng cử viên tại vòng chung kết của cuộc thi CAESAR nên chưa có nhiều nghiên cứu được công bố đối với thuật toán này. Tuy nhiên, trong quá trình diễn ra cuộc thi CAESAR, các nhà mật mã học đã thực hiện một số tấn công lên thuật toán ACORN nhằm đánh giá sự an toàn cũng như khả năng chống lại các tấn công của nó.

Sau đây là một số tấn công nổi bật lên thuật toán ACORN đã được công bố:

Tên tấn công	Tác giả	Kết quả
Tấn công khối lập phương (Cube Attack) [7]	- Md Iftekhar Salam - Harry Bartlett - Ed Dawson - Josef Pieprzyk - Leonie Simpson - Kenneth Koon-Ho Wong	- Tấn công khối lập phương lên quá trình khởi tạo với số vòng rút gọn 477 với độ phức tạp 2^{35} . - Tấn công khối lập phương lên quá trình mã hóa với độ phức tạp $2^{72,8}$ trong trường hợp khóa và véc-tơ khởi tạo được dùng để mã nhiều bản rõ.
Tấn công khôi phục khóa (Full key-recovery) [2]	- Colin Chaigneau - Thomas Fuhr - Henri Gilbert	- Khôi phục lại khóa được sử dụng với độ phức tạp $2^{235-42\times(n-1)}$ (n là số bản rõ được chọn để áp dụng tấn công) trong trường hợp sử dụng lại véc-tơ khởi tạo (hoặc giá trị Nonce)
Tấn công gây lỗi vào trạng thái (Fault Attack) [11]	- Xiaojuan Zhang - Xiutao Feng - Dongdai Lin	- Khôi phục được trạng thái ban đầu với độ phức tạp $c\times 2^{146,5-3,52\times n}$ Trong đó: c là độ phức tạp thời gian để giải các phương trình tuyến tính và n là số lỗi thực nghiệm ($26 < n < 43$).

a. Tấn công khối lập phương

Tấn công khối lập phương được giới thiệu bởi Dinur và Sharmir tại Eurocrypt 2009 [3]. Đây là kỹ thuật tấn công được sử dụng rộng rãi để thám hệ mật và đặc biệt hiệu quả với các hệ mã dòng dựa trên các thanh ghi dịch phi tuyến bậc thấp. Tấn công khối lập phương là một dạng tấn công đại số với mục đích là tìm ra các giá trị bí mật của hệ mật bằng việc vận dụng và giải các phương trình đa

thức của hệ mật. Ý tưởng của tấn công khối lập phương là cấu trúc các giá trị công khai (như bản mã, véc-tơ khởi tạo, dữ liệu liên kết) thành các khối lập phương Boolean n-chiều có kích thước $l_c = d - 1$ (d là bậc của đa thức đầu ra) để tạo ra đủ số lượng các phương trình và giải các phương trình này để tìm ra các giá trị bí mật của hệ mật.

Tấn công khối lập phương đối với thuật toán ACORN được Salam và các cộng sự thực hiện trong quá trình khởi tạo, quá trình mã hóa hoặc quá trình giải mã [6]. Thuật toán ACORN không nhận bất kỳ đầu vào bên ngoài trong suốt quá trình tạo thẻ xác thực nên tấn công khối lập phương không thể thực hiện trong giai đoạn này.

Quá trình khởi tạo của ACORN có số vòng lớn: $adlen + 2048$. Bậc đa thức đầu ra của ACORN tăng khi số lượng vòng tăng và kích thước khối lập phương sẽ rất lớn nếu chọn khối lập phương sau 2048 vòng. Điều này yêu cầu một lượng lớn thời gian tính toán. Do đó tấn công thử nghiệm được thực hiện trên thuật toán ACORN với số vòng quá trình khởi tạo giảm. Cụ thể số vòng là 500 và tính toán ra được 21 phương trình tuyến tính. Với tổng cộng 21 phương trình này, cần phải đoán ra 107 bit khóa và 21 bit còn lại có thể được tìm ra bằng việc giải các phương trình tuyến tính. Điều này có thể giúp cải thiện việc tìm kiếm vét cạn. Để rút ngắn việc tìm kiếm vét cạn hơn nữa, thực hiện đã giảm thiểu số vòng xuống 477 mà vẫn tìm ra được 21 phương trình tuyến tính. Sau khi đã có được các phương trình tuyến tính, tiến hành giải các phương trình này và tính toán độ phức tạp của tấn công là khoảng 2^{35} với số vòng quá trình khởi tạo là 477.

Tấn công có thể được mở rộng với số vòng lớn hơn trong quá trình khởi tạo, nhưng điều này đòi hỏi nghiên cứu trên các không gian lập phương rất lớn. Đây là vấn đề khó khăn để đánh giá hiệu suất của tấn công khối lập phương với số vòng mở rộng của ACORN mà không biết kích thước khối lập phương phù hợp. Do vậy, các nhà nghiên cứu chỉ tiến hành tấn công trên khối lập phương với kích thước nhỏ. Nếu thực hiện tiếp tục với số vòng lớn hơn thì quá trình khởi tạo của thuật toán là an toàn.

Tấn công khối lập phương áp dụng lên quá trình mã hóa nhằm tìm ra trạng thái của thuật toán. Tấn công sử dụng bản rõ làm đầu vào cho thuật toán. Tại mỗi vòng nạp bản rõ, xem xét hàm đầu ra để xác định xem nó có trở thành tuyến tính khi phân biệt nó với các bản rõ khác. Đây có thể coi là tấn công lựa chọn bản rõ. Bằng phương pháp này, các tác giả đã xác định được 245 phương trình tuyến tính với 293 biến. Sau đó tiến hành giải các phương trình để tìm ra các bit trạng thái. Quá trình này có thể tìm ra được các bit trạng thái với độ phức tạp khoảng $2^{72,8}$. Điều này chỉ ra rằng việc thực hiện tấn công nhằm tìm ra được trạng thái của thuật toán có độ phức tạp nhỏ hơn so với việc thực hiện tìm kiếm vét cạn. Tuy nhiên, điều này chỉ đúng khi người gửi dùng cùng một khóa và véc-tơ khởi tạo để mã hóa hoặc giải mã các tập bản rõ đầu vào. Nếu sử dụng đúng theo đề nghị của người thiết kế, tấn công này không đe dọa đến sự an toàn của thuật toán ACORN.

b. Tấn công khôi phục khóa

Tấn công khôi phục khóa lên thuật toán ACORN được Chaigneau và các cộng sự thực hiện với giả định là véc-tơ khởi tạo (hay giá trị Nonce) được sử dụng lặp lại [2]. Tấn công được mô tả như sau:

- Nếu một véc-tơ khởi tạo (hay giá trị Nonce) được dùng để mã hóa một vài thông báo mà có cùng dữ liệu liên kết, trạng thái của ACORN là giống nhau sau quá trình xử lý dữ liệu liên kết. Do đó có thể thu thập thông tin về giá trị của trạng thái.

- Để đạt được điều đó, thực hiện khai thác dòng bit khóa dựa vào bản rõ sau một vài vòng. Điều này cho phép có được một hệ phương trình tuyến tính liên quan đến trạng thái của thuật toán. Hệ phương trình này có khả năng có một số cách giải sau khi được kiểm tra kỹ lưỡng.

- Tiếp theo, một khi trạng thái được khôi phục hoàn toàn, thực hiện các bước ngược lại của thuật toán để tìm ra toàn bộ khóa, điều này cho phép giải mã bất kỳ bản mã và véc-tơ khởi tạo (hay giá trị Nonce) nào.

Với cách thức tấn công như trên, độ phức tạp của tấn công là khoảng $2^{235-42\times(n-1)}$ với n là số bản rõ được chọn để thực hiện tấn công trên cùng một véc-tơ khởi tạo (hay giá trị Nonce) và cùng dữ liệu liên kết. Nếu áp dụng tấn công với 7 bản rõ được lựa chọn trở lên, thì tính an toàn của thuật toán sẽ mất.

Tuy nhiên, tấn công chỉ thực hiện được nếu như người dùng sử dụng lại véc-tơ khởi tạo và dữ liệu liên kết với cùng nhiều bản mã. Nếu sử dụng đúng thuật toán theo khuyến nghị của nhà thiết kế thì tấn công này không đe dọa đến sự an toàn của thuật toán ACORN.

c. Tấn công gây lỗi

Tấn công gây lỗi là một loại tấn công kênh kề thực hiện trên các thiết bị vật lý. Đây là một loại hình tấn công mạnh để tìm ra khóa bí mật của nhiều nguyên thủy mật mã. Tấn công cho phép kẻ tấn công đưa các lỗi bằng laser (hoặc các xung không đều) nhằm thay đổi một hoặc nhiều bit trạng thái của thuật toán. Tiếp theo phân tích sự khác biệt giữa trạng thái lỗi và trạng thái không lỗi, từ đó có thể suy luận ra một số thông tin về trạng thái bí mật hoặc khóa bí mật.

Tấn công gây lỗi lên ACORN được Zhang và các cộng sự thực hiện gồm 2 bước chính: xác định lỗi và giải các phương trình tìm được từ các lỗi [11].

Tại bước xác định lỗi, khi có 1 lỗi được đưa ngẫu nhiên vào trong trạng thái, có thể có được 1 chuỗi các bit khác nhau giữa dòng bit lỗi và dòng bit không lỗi. Với các chuỗi bit này, có thể xác định vị trí các lỗi.

Tại bước thứ 2, một khi các lỗi được xác định, có thể tìm được đủ các phương trình liên quan trạng thái ban đầu. Ý tưởng chính để tìm được các phương trình là quan sát sự khác biệt của dòng bit khóa đầu tiên và mô tả sự khác biệt này như một hàm tuyến tính đối với trạng thái ban đầu. Sau đó, sử dụng phương pháp dự đoán và xác định để tìm được trạng thái ban đầu.

Với trung bình một lỗi được thực nghiệm, có thể có trung bình 7,03 phương trình tuyến tính và 4,23 phương trình phi tuyến. Thực nghiệm trên 27 lỗi đạt được 304 phương trình trong đó có 190 phương trình phi tuyến. Sử dụng phương pháp

dự đoán và xác định để giải các phương trình này, độ phức tạp về thời gian để tìm được trạng thái ban đầu là:

$$c \times 2^{146,5-3,52 \times n}$$

trong đó: c là độ phức tạp thời gian của việc giải các phương trình tuyến tính, n là số lỗi thực nghiệm ($26 < n < 43$)

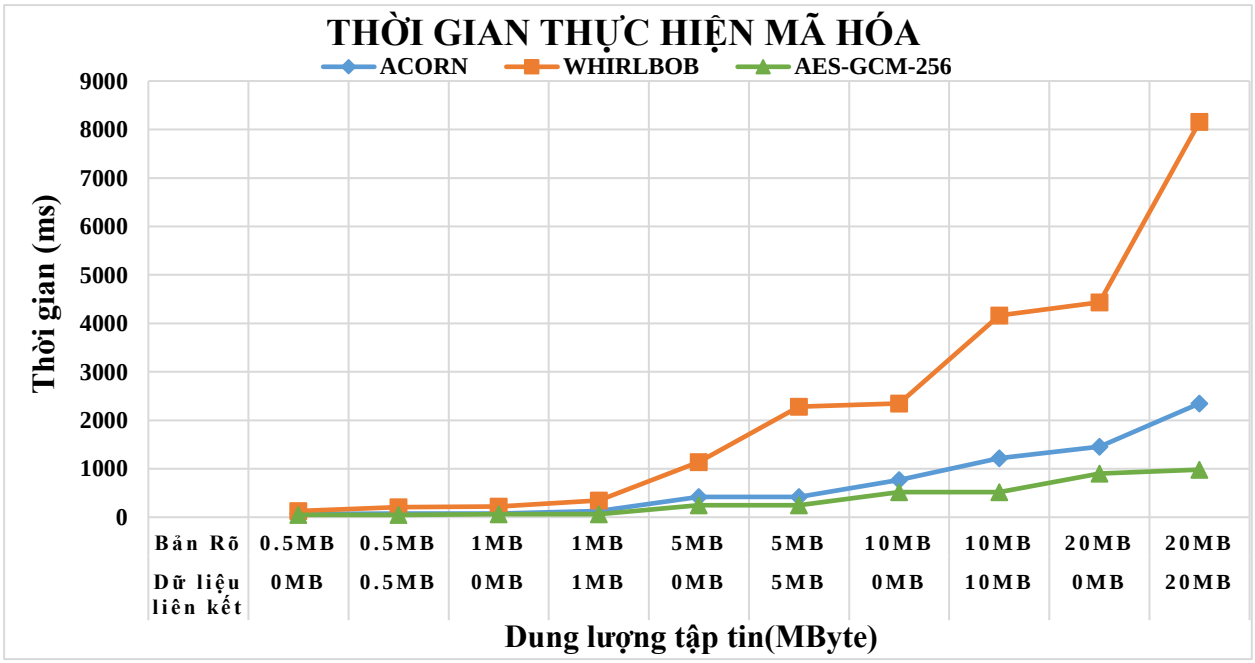
Để thực hiện tấn công, kẻ tấn công cần có khả năng khởi động lại thiết bị vật lý với cặp khóa – véc-tơ khởi tạo ban đầu cũng như có thể chạy lại thuật toán nhiều lần. Khi đưa các lỗi vào trên trong trạng thái, các lỗi này được đưa một cách ngẫu nhiên, kẻ tấn công không thể chọn được vị trí.

IV. HIỆU SUẤT THỰC THI CỦA THUẬT TOÁN ACORN

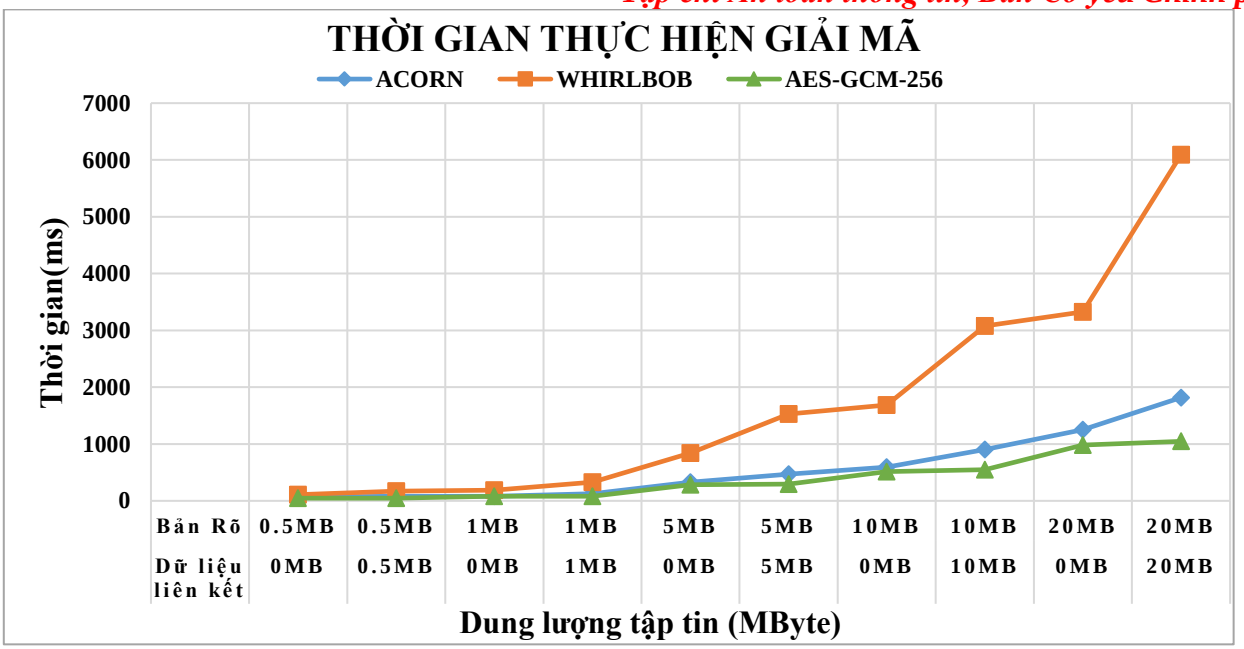
Để đánh giá hiệu suất thực thi về mặt thời gian của thuật toán ACORN, ta so sánh nó với 2 thuật toán mã hóa có xác thực khác là WHIRLBOB và AES-GCM-256.

Thuật toán WHIRLBOB là một ứng viên tại vòng hai của cuộc thi CAESAR. Đây là thuật toán có nguồn gốc từ hàm băm Whirlpool và thuật toán mã hóa có xác thực Stribobr1.

Còn AES-GCM là một thuật toán mã hóa có xác thực được sử dụng rộng rãi trước khi cuộc thi CAESAR được tổ chức. Đây là thuật toán sử dụng mã khối AES ở chế độ GCM, có thể cung cấp cả tính bảo mật và xác thực. Cài đặt thuật toán AES-GCM-256 do Romain Dolbeau thực hiện sử dụng thư viện Crypto++.



Hình 4. Thời gian thực hiện mã hóa của 3 thuật toán ACORN, WHIRLBOB, AES-GCM-256.



Hình 5. Thời gian thực hiện giải mã của 3 thuật toán ACORN, WHIRLBOB, AES-GCM-256.

Việc đánh giá hiệu suất thực thi về mặt thời gian của ba thuật toán ACORN, WHIRLBOB, AES-GCM-256 được thực hiện trên cùng một nền tảng phần cứng là máy tính xách tay với vi xử lý Intel Core i7-4510U, xung nhịp 2.0 ~ 2.6 GHz, bộ nhớ RAM 4GB. Đánh giá quá trình mã hóa và giải mã với các file có kích thước khác nhau và các trường hợp sử dụng dữ liệu liên kết và không sử dụng dữ liệu liên kết. Kết quả đánh giá được thể hiện ở Hình 4 và Hình 5. Kết quả trên cho thấy rằng hiệu suất thực thi quá trình mã hóa và giải mã của thuật toán ACORN là nhanh hơn so với WHIRLBOB và chậm hơn so với AES-GCM-256.

V. KẾT LUẬN

ACORN được thiết kế dựa trên 6 thanh ghi dịch phản hồi tuyến tính để mã hóa bản rõ P và dữ liệu liên kết AD có độ dài nhỏ hơn 2^{64} bit. Đầu ra của thuật toán là bản mã C có độ dài nhỏ hơn 2^{64}bit và một thẻ xác thực T có độ dài t $(64\text{bit} \le t \le 128\text{bit})$. Các tham số an toàn của ACORN lần lượt là khóa 128 bit, vec-tơ khởi tạo 128 bit, giá trị ngẫu nhiên 128 bit và thẻ xác thực từ 64 đến 128 bit đã thể hiện nó là một thuật toán mã hóa có xác thực mạnh có khả năng kháng được các tấn công phổ biến lên mã dòng như tấn công khối lập phương, tấn công cân bằng thời gian – dữ liệu – bộ nhớ, tấn công dự đoán và xác định,… Hiện nay cuộc thi CAESAR đã đến vòng chung kết, việc nghiên cứu về các tính chất mật mã của thuật toán ACORN vẫn đang được tiếp tục, hứa hẹn sẽ có nhiều phân tích lên thuật toán, nhằm hoàn thiện và có thể ứng dụng được trong tương lai.

TÀI LIỆU THAM KHẢO

[1]. CAESAR: https://competitions.cr.yp.to/caesar.html.

[2]. Colin Chaigneau, Thomas Fuhr and Henri Gilbert, *Full Key-recovery on ACORN in Nonce-reuse and Decryption-misuse settings*, 2015.

[3]. I. Dinur, A. Shamir, *Cube Attacks on Tweakable Black Box Polynomials*, In Advances in Cryptology - EUROCRYPT 2009, pp. 278-299, 2009.

[4]. Hongjun Wu, *ACORN: A Lightweight Authenticated Cipher (v3)*. Submission to CAESAR, 2016.

[5]. M. Bellare and C. Namprempre, *Authenticated encryption: Relations among notions and analysis of the generic composition paradigm*. July 2007.

[6]. Md Iftekhar Salam, Harry Bartlett, Ed Dawson, Josef Pieprzyk, Leonie Simpson and Kenneth Koon-Ho Wong. *Investigating Cube Attacks on the Authenticated Encryption Stream Cipher ACORN*. Singapore, 2016.

[7]. N. T. Courtois, *Fast Algebraic Attacks on Stream Ciphers with Linear Feedback*, CRYPTO 2003.

[8]. S. Babbage, *A Space, Time Trade in Exhaustive Search Attacks on Stream Ciphers*, May 1995.

[9]. W. Meier and O. Staffelbach, *Fast Correlation Attacks on Certain Stream Ciphers*, Journal of Cryptography, Page159-176, 1989.

[10]. X. Feng, J. Liu, Z. Zhou, C. Wu and D. Feng, *A Byte-Based Guess and Determine Attack on SOSEMANUK*, 2010.

[11]. Xiaojuan Zhang, Xiutao Feng, Dongdai Lin, *Fault Attack on ACORN v3*, China. 2016.