TẠP CHÍ KHOA HỌC VÀ CÔNG NGHỆ ĐẠI HỌC DUY TÂN
DTU Journal of Science and Technology 4(71) (2025) 14-24
ĐẠI HỌC DUY TÂN
DUYTAN UNIVERSITY

Nghiên cứu, thiết kế và kiểm tra thuật toán mã hóa GIFT-128-bit

Research, design and testing of the GIFT-128-bit encryption algorithm

Phạm Anh Tuấn[a,b], Trần Lê Thăng Đồng[a], Nguyễn Ngô Anh Quân[c*], Nguyễn Thị Bích Hạnh[d]
Pham Anh Tuan[a,b], Tran Le Thang Dong[a], Nguyen Ngo Anh Quan[c*], Nguyen Thi Bich Hanh[d]

[a]Viện Công nghệ Kỹ thuật Hàng không Vũ trụ, Đại học Duy Tân, Đà Nẵng, Việt Nam
[a]Institute of Aerospace Engineering and Technology, Duy Tan University, Da Nang, 550000, Viet Nam
[b]Trung tâm Tiêu chuẩn - Đo lường - Chất lượng 3, Đà Nẵng, Việt Nam
[b]Center for Standards - Metrology - Quality 3, Da Nang, 550000, Viet Nam
[c]Trung tâm Điện - Điện tử, Trường Công nghệ và Kỹ thuật, Đại học Duy Tân, Đà Nẵng, Việt Nam
[c]Center of Electrical - Engineering (CEE), School of Engineering and Technology, Duy Tan University, Da Nang, 550000, Viet Nam
[d]Khoa Điện - Điện tử, Trường Công nghệ và Kỹ thuật, Đại học Duy Tân, Đà Nẵng, Việt Nam
[d]Faculty of Electrical - Electronic Engineering (FEEE), School of Engineering and Technology, Duy Tan University, Da Nang, 550000, Viet Nam

(Ngày nhận bài: 06/05/2025, ngày phản biện xong: 19/07/2025, ngày chấp nhận đăng: 29/07/2025)

Tóm tắt

Đề tài trình bày cơ sở lý thuyết về thuật toán mã hóa xác thực dựa trên cấu trúc mã khối nhẹ GIFT-COFB 128-bit, đảm bảo kết hợp được tính bảo mật và tính xác thực trong quá trình truyền dữ liệu, phù hợp với các hệ thống, thiết bị ứng dụng IoT nhỏ gọn, có khả năng xử lý tính toán và tài nguyên hạn chế, chi phí triển khai thấp. Xây dựng và kiểm tra thuật toán mã hóa GIFT-COFB 128-bit, triển khai giải pháp mã hóa này trên nền tảng ngôn ngữ mô tả phần cứng Verilog HDL và công nghệ FPGA. Thiết kế hệ thống dựa trên cấu trúc mã khối nhẹ GIFT-COFB 128-bit với các tham số an toàn như sử dụng nguyên lý mạng hoán vị thay thế 128-bit, độ dài khóa 128-bit cùng cơ chế mã hóa xác thực COFB đã thu được kết quả bước đầu đáng tin cậy trong quá trình truyền nhận dữ liệu.

Từ khóa: GIFT-COFB; 128-bit; khối mã hóa; thuật toán mã hóa.

Abstract

The project presents the lightweight block cipher-based authenticated encryption mode GIFT- COFB 128-bit, ensuring confidentiality and authenticity in the data transmission process, which is suitable for compact IoT application systems and devices with limited computational capabilities and resources, and low-cost deployment. The process of building and testing the encryption algorithm GIFT-COFB 128-bit is described, including its deployment of this encryption solution on the hardware description language Verilog HDL and FPGA hardware platform. The system design, based on the lightweight block cipher structure GIFT-COFB 128-bit, employs security parameters including a 128-bit substitution-permutation network, a 128-bit key length, and the COFB authenticated encryption mechanism. This design has demonstrated reliable preliminary evaluation results during the data transmission process.

Keywords: GIFT-COFB; 128-bit; block cipher; encryption algorithm.

[*]*Tác giả liên hệ:* Nguyễn Ngô Anh Quân
Email: nguyennanhquan@dtu.edu.vn

1. Giới thiệu

Cùng với sự phát triển của công nghệ Internet vạn vật (IoT) và Trí tuệ nhân tạo (AI), ngày càng có nhiều hệ thống, thiết bị thông minh tham gia vào tất cả các hoạt động khoa học kỹ thuật - công nghệ, sản xuất công nghiệp, cho đến các tiện ích, ứng dụng phục vụ cuộc sống hàng ngày của chúng ta, từ quá trình đo lường tự động, giám sát, điều khiển đến trao đổi thông tin, giao tiếp [1, 2]. Trong quá trình này, dữ liệu được trao đổi giữa các hệ thống, thiết bị này tăng lên với quy mô ngày càng lớn. Cùng với đó, đặt ra nhiều thách thức cho việc bảo vệ dữ liệu của chúng ta một cách an toàn hơn, tối ưu hơn, đặc biệt là đối với các hệ thống, thiết bị ứng dụng IoT, có tài nguyên, khả năng xử lý tính toán giới hạn, cũng như chi phí triển khai thấp [3].

Hiện nay, một trong các giải pháp hiệu quả giải quyết vấn đề này là ứng dụng quá trình mã hóa xác thực (Authenticated Encryption - AE). GIFT-COFB sử dụng mã khối GIFT-128 là một trong những thuật toán mã hóa xác thực AE dựa trên mã khối (Block Cipher) được lựa chọn hàng đầu, đã được Viện Kỹ sư Điện và Điện tử (Institute of Electrical and Electronics Engineers - IEEE), Viện Tiêu chuẩn và Công nghệ Quốc gia (National Institute of Standards and Technology – NIST) của Mỹ công nhận là một trong các giải pháp bảo mật cho các hệ thống IoT [4, 5]. Trong đó, cấu trúc COFB (Combined Feedback) trong thuật toán này giúp giảm kích thước bộ nhớ trạng thái tổng thể bằng cách kết hợp đầu ra mã khối với các khối dữ liệu và sử dụng kỹ thuật che giấu phụ thuộc dữ liệu. Điều này làm giảm đáng kể chi phí phần cứng, đồng thời đảm bảo tính bảo mật cao. Các công trình [6-10] đã nghiên cứu đề xuất nhiều giải pháp cải tiến cấu trúc COFB, khối S-box, giải thuật bảo mật dạng khóa công khai với đặc điểm nổi trội về bảo mật đối với các cấu trúc phần cứng IoT có tài nguyên hạn chế.

Quy trình giải thuật GIFT-COFB sử dụng mã khối GIFT-128 kết hợp được cả tính bảo mật (Confidentiality) và tính xác thực (Authenticity). Ngoài ra, một trong những đặc tính quan trọng nữa của GIFT-COFB là khả năng hoạt động với tỉ lệ mã hóa cao, chỉ cần một lần mã hóa cho mỗi khối dữ liệu, cũng như việc không yêu cầu tính nghịch đảo của mã khối trong quá trình giải mã, tối ưu hóa quá trình mã hóa/giải mã giúp đạt hiệu quả cao về bộ nhớ trạng thái, với kích thước được tối ưu cho các ứng dụng phần cứng hạn chế làm giảm thêm độ phức tạp và chi phí tính toán [7-9]. Trên cơ sở nghiên cứu lý thuyết về kỹ thuật mã hóa khối GIFT-128, kết hợp cấu trúc COFB, nhóm tác giả đề tài đề xuất phương án xây dựng và kiểm tra thuật toán mã hóa GIFT-COFB 128-bit, triển khai giải pháp mã hóa này trên nền tảng ngôn ngữ mô tả phần cứng Verilog HDL và công nghệ FPGA. Kết quả nghiên cứu và triển khai thành công quy trình giải thuật GIFT-COFB sử dụng mã khối GIFT-128 trên phần cứng FPGA cho thấy khả năng ứng dụng của giải thuật vào các thiết kế hệ thống, thiết bị ứng dụng IoT với tài nguyên, khả năng xử lý tính toán giới hạn, mà có yêu cầu về tính bảo mật, xác thực về thông tin, dữ liệu trong đời sống kinh tế, xã hội cũng như trong lĩnh vực quốc phòng, an ninh.

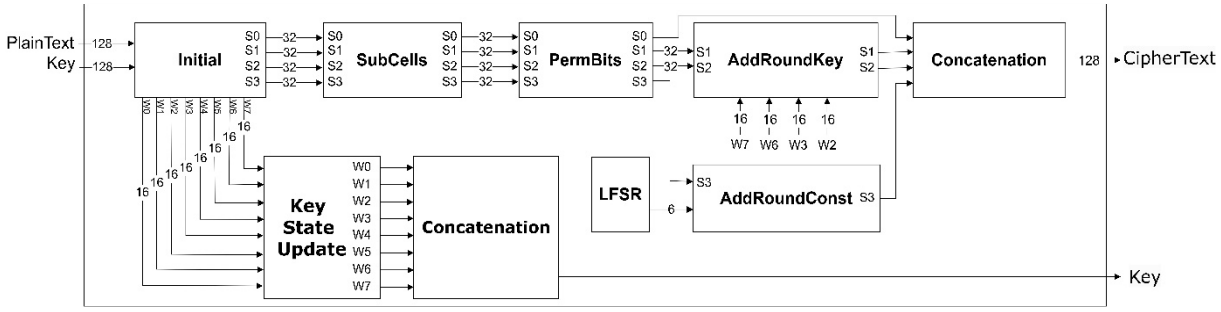
2. Mô tả thuật toán mã hóa GIFT-COFB 128-bit

2.1. Mã hóa khối GIFT-128

Hoạt động mã hóa theo thuật toán GIFT-COFB 128-bit sẽ biến đổi dữ liệu ban đầu (Plaintext) với kích thước lớn nhất là 128-bit để thành bản tin mã hóa có cùng kích thước và thẻ xác thực (Tag) với kích thước cố định 128-bit. Trong đó, mã hóa khối GIFT-128 đóng vai trò chính trong thuật toán GIFT-COFB, thực hiện mã hóa dữ liệu trong từng khối của quá trình mã hóa - xác thực.

Mã hóa khối GIFT-128 được thiết kế dựa trên nguyên lý mạng hoán vị thay thế (Substitution-Permutation Network) 128-bit với độ dài khóa 128-bit. Đây là một mã khối lặp 40 vòng với hàm vòng giống nhau, trong đó mỗi vòng của GIFT-

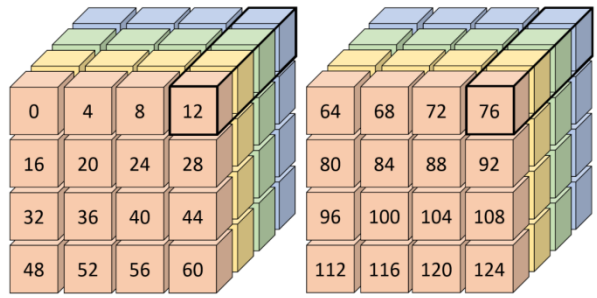
128 bao gồm 3 bước: Thay thế ô con (SubCells), Hoán vị bit (PermBits) và Cộng khóa vòng (AddRoundKey). Dựa trên việc phân tích chi tiết cấu trúc bên trong của khối mã hóa GIFT-128, sơ đồ khối mã hóa GIFT-128 (Hình 1) dưới đây được xây dựng để minh họa toàn bộ quy trình xử lý, từ dữ liệu ban đầu (Plaintext) và khóa (Key) cho đến dữ liệu đầu ra (Ciphertext), gồm nhiều bước xử lý tuần tự, nối tiếp (Concatenation), mỗi bước đảm nhận một chức năng quan trọng nhằm đảm bảo tính bảo mật và hiệu quả của thuật toán mã hóa/giải mã và xác thực [7, 8].



Hình 1. Sơ đồ quá trình mã hóa khối GIFT 128-bit

*** Khởi tạo (Initial)**: Dữ liệu ban đầu (Plaintext) 128-bit được chia thành 4 phần 32-bit và đưa vào trạng thái mã (Cipher State) S. S là mảng 2 chiều, thứ tự bit là từ trên xuống dưới, sau đó từ trái sang phải.

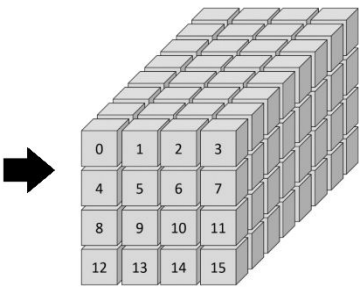
$$S = \begin{bmatrix} S_0 \\ S_1 \\ S_2 \\ S_3 \end{bmatrix} \leftarrow \begin{bmatrix} b_{124} & \dots & b_8 & b_4 & b_0 \\ b_{125} & \dots & b_9 & b_5 & b_1 \\ b_{126} & \dots & b_{10} & b_6 & b_2 \\ b_{127} & \dots & b_{11} & b_7 & b_3 \end{bmatrix}$$



Hình 2. Quá trình khởi tạo dữ liệu Plaintext

Khóa bí mật (Key Secret) 128-bit được chia thành 8 phần 16-bit và đưa vào trạng thái khóa (Key State) KS. KS là một mảng 2 chiều, thứ tự bit là từ phải sang trái, sau đó từ dưới lên.

$$KS = \begin{bmatrix} W_0 \,\|\, W_1 \\ W_2 \,\|\, W_3 \\ W_4 \,\|\, W_5 \\ W_6 \,\|\, W_7 \end{bmatrix} \leftarrow \begin{bmatrix} b_{127} & \cdots & b_{112} & \| & b_{111} & \cdots & b_{98} & b_{97} & b_{96} \\ b_{95} & \cdots & b_{80} & \| & b_{79} & \cdots & b_{66} & b_{65} & b_{64} \\ b_{63} & \cdots & b_{48} & \| & b_{47} & \cdots & b_{34} & b_{33} & b_{32} \\ b_{31} & \cdots & b_{16} & \| & b_5 & \cdots & b_2 & b_1 & b_0 \end{bmatrix}$$

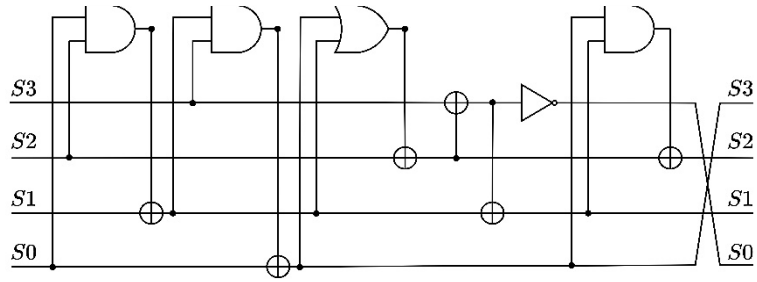


Hình 3. Quá trình khởi tạo Key 128-bit

Thay vì xử lý một khối dữ liệu 128-bit tuần tự, sử dụng kỹ thuật bit-slice chia khối dữ liệu đó thành 4 khối 32-bit cho phép thực hiện tính toán song song, nhằm tối ưu hóa quá trình khởi tạo dữ liệu ban đầu (Plaintext), trước khi đưa dữ liệu vào quá trình xử lý thay thế ô con (SubCells), hoán vị bit (Permutation), cộng khóa vòng (AddRoundKey).

*** Thay thế ô con (SubCells)**: Cập nhật trạng thái mã (Cipher State) S theo cấu trúc hộp thay thế S-Box (Substitution Box) sau:

$$
\begin{aligned}
S_1 &\leftarrow S_1 \oplus (S_0 \& S_2)\\
S_0 &\leftarrow S_0 \oplus (S_1 \& S_3)\\
S_2 &\leftarrow S_2 \oplus (S_0 \mid S_1)\\
S_3 &\leftarrow S_3 \oplus S_2\\
S_1 &\leftarrow S_1 \oplus S_3\\
S_3 &\leftarrow \sim S_3\\
S_2 &\leftarrow S_2 \oplus (S_0 \& S_1)\\
\{S_0, S_1, S_2, S_3\} &\leftarrow \{S_3, S_1, S_2, S_0\}
\end{aligned}
$$



Hình 4. Mô tả cấu trúc S-Box

*** Hộp thay thế S-Box (Substitution Box)**: là một thành phần cơ bản của hầu hết các thuật toán mã hóa khối. S-Box hoạt động trên các khối dữ liệu 4-bit. S-Box là một hàm phi tuyến được sử dụng để thực hiện hoán vị trên các bit hoặc byte của dữ liệu đầu vào, khi đó, dữ liệu đầu ra nhận được là một giá trị 4-bit (0000 đến 1111), tương ứng với kết quả ánh xạ dữ liệu đầu vào (một giá trị 4-bit, từ 0000 đến 1111) theo cấu trúc S-Box như trên.

*** Hoán vị bit (PermBits)**: Phép hoán vị trong GIFT-128 là hoán vị ở mức bit (bit-Level Permutation), thực hiện việc di chuyển các bit đến các vị trí mới theo nguyên tắc nhất định. PermBits thực hiện di chuyển mỗi bit tại vị trí `i` trong khối 128-bit đến một vị trí mới theo một ánh xạ định trước, từ đó mỗi khối trạng thái mã (Cipher State) S_0, S_1, S_2, S_3 được hoán vị độc lập. PermBits được thiết kế để hoán đổi các bit nhằm đảm bảo tính bảo mật và khuếch tán dữ liệu đầu vào qua các vòng mã hóa. Đây là một bước phi tuyến tính**,** kết hợp với cấu trúc S-Box, phép XOR cùng khóa Key để làm tăng độ khó của việc phân tích thuật toán giúp tăng tính phức tạp toán học cho dữ liệu trước khả năng tấn công tuyến tính **(**Linear Attacks) và tấn công sai lệch (Differential Attacks), đảm bảo tính bảo mật của thuật toán cũng như tính toàn vẹn của dữ liệu.

*** Cộng khóa vòng (AddRoundKey)**: Phép biến đổi này bao gồm bước thêm khóa vòng (Round Key) và hằng số vòng (Round Constant). Đầu tiên, hai khối 32-bit U, V được trích xuất từ trạng thái khóa (Key State):

$$U \leftarrow W_2 \,||\, W_3,\ V \leftarrow W_6 \,||\, W_7,\ RK = U \,||\, V$$

Tiếp theo, để bổ sung khóa vòng, U và V được XOR với S_1 và S_2 như sau:

$$S_2 \leftarrow S_2 \oplus U,\ \ S_1 \leftarrow S_2 \oplus V$$

Bổ sung hằng số vòng, S_3 được cập nhật như sau:

$S_3 \leftarrow S_3 \oplus 0\text{x}800000\text{XY}$, trong đó $\text{XY}=00C_5C_4C_3C_2C_1C_0$.

Trạng thái khóa sau khi được cập nhật như sau:

$$\begin{bmatrix} W_0 & || & W_1 \\ W_2 & || & W_3 \\ W_4 & || & W_5 \\ W_6 & || & W_7 \end{bmatrix} \leftarrow \begin{bmatrix} W_6 >>> 2 & || & W_7 >>> 12 \\ W_0 & || & W_1 \\ W_2 & || & W_3 \\ W_4 & || & W_5 \end{bmatrix}$$

Hằng số vòng được tạo ra bằng cách dùng thanh ghi dịch phản hồi tuyến tính (Liner Feedback Shift Register - LFSR), được ký hiệu là $C_5C_4C_3C_2C_1C_0$. Hàm cập nhật giá trị của nó được định nghĩa như sau:

$$c_5 \,||\, c_4 \,||\, c_3 \,||\, c_2 \,||\, c_1 \,||\, c_0 \leftarrow c_4 \,||\, c_3 \,||\, c_2 \,||\, c_1 \,||\, c_0 \,||\, c_5 \oplus c_4 \oplus 1$$

Bước AddRoundKey bao gồm việc kết hợp khóa bí mật và hằng số vòng vào trạng thái mã (Cipher State) trong mỗi vòng lặp.

2.2. Chế độ mã hóa xác thực COFB

Quy trình giải thuật GIFT sử dụng mã khối GIFT-128 kết hợp chế độ mã hóa xác thực COFB, cho phép nâng cao tính bảo mật và tính xác thực [9]. Khi đó, các lệnh gọi khối mật mã (Block Cipher) E_K được thực hiện xen kẽ với lệnh gọi của các hàm thành phần, và mỗi nonce (với $N \in \{0,\ 1\}^n$), được mã hóa bằng E_K để tạo ra giá trị nối chuỗi nội bộ đầu tiên. Quá trình này sử dụng một thủ tục cắt bớt Trunc k(x) để truy xuất k bit quan trọng nhất của đầu vào n-bit và một hàm đệm (Padding Function) mở rộng các đầu vào có độ dài không phải là bội số của *n*, để đảm bảo dữ liệu đầu vào có kích thước phù hợp với yêu cầu của thuật toán, thường là một bội số của kích thước mã hóa khối [10].

* Hàm đệm (Padding Function) được mô tả như sau:

$$Pad(x) = \begin{cases} x & \text{, nếu } x \neq \varepsilon \text{ và} \\ x \| 10^{(n-(|x| \bmod n)-1)} & |x| \bmod n = 0 \\ & \text{, ngược lại} \end{cases}$$

Khi đó, nếu x không rỗng và độ dài của x chia hết cho n thì chuỗi x được giữ nguyên, ngược lại thì sẽ nối các bit 0 vào sau *x*.

*** Hàm phản hồi (Feedback Function)** – được định nghĩa để xử lý các phép biến đổi dữ liệu trong giải thuật GIFT-COFB, đảm bảo tính ngẫu nhiên và an toàn trong quá trình mã hóa/giải mã và xác thực. Với $Y \in \{0,\ 1\}^n$ và chuỗi Y được chia thành 2 phần $(Y[1],\ Y[2]) \leftarrow Y$, thì hàm phản hồi G được định nghĩa như sau:

$$G(Y) = (Y[2],\ Y[1] <<< 1)$$

Hàm G(Y) thực hiện biến đổi dữ liệu dựa trên việc kết hợp *Y*[2] và *Y*[1] (đã xoay trái) để tạo ra một đầu ra mới. Hàm G cũng có thể được biểu diễn như một ma trận n×n, gọi là ma trận không suy biến (Non-singular Matrix). Điều này đảm bảo *G* có tính chất đảo ngược, giúp thực hiện được cả mã hóa và giải mã.

Khi đó, với $M \in B$ và $Y \in B$, hàm phản hồi trong quá trình mã hóa dữ liệu ρ_1 thực hiện phép kết hợp giữa *Y* và *M* để tạo một giá trị mới, được định nghĩa là $\rho_1(Y, M) = G \cdot Y \oplus M$. Thì hàm mã hóa $\rho(Y, M)$ sử dụng $\rho_1(Y, M)$ để kết hợp dữ liệu gốc *M* và trạng thái trước đó *Y* nhằm tạo ra dữ liệu đầu ra (Ciphertext) *C* được định nghĩa như sau:

$$\rho(Y, M) = (\rho_1(Y, M), Y \oplus M)$$

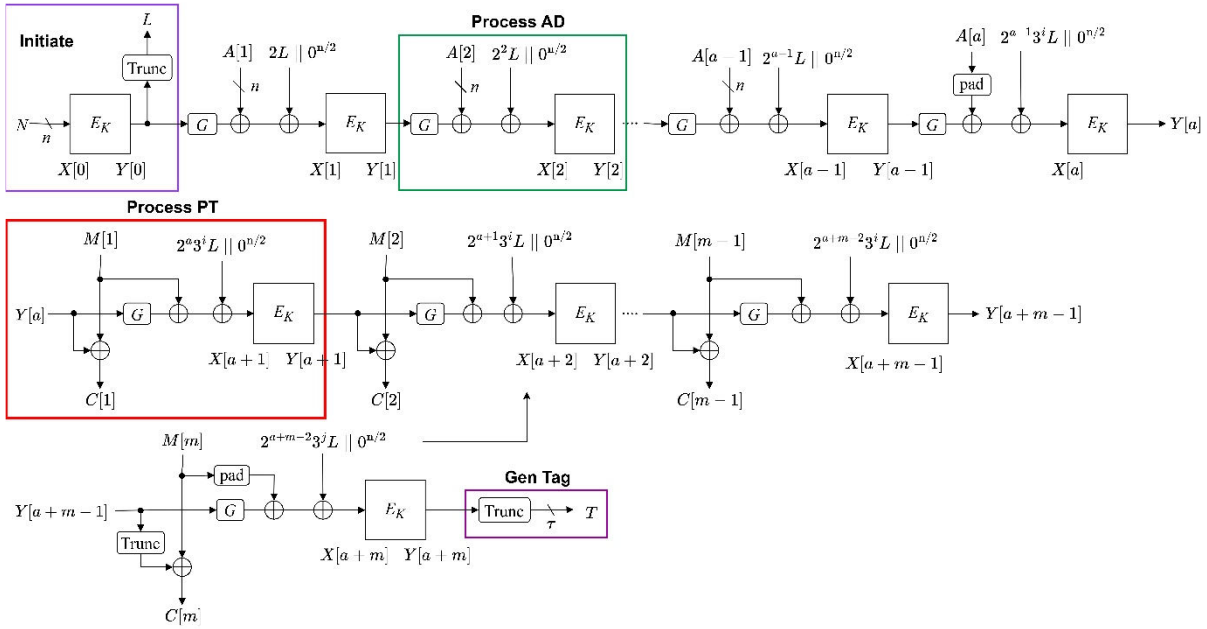
Hàm giải mã ρ'(Y, C) dùng phép XOR với C để truy xuất M được định nghĩa như sau:

$$\rho'(Y, C) = (\rho_1(Y, Y \oplus C), Y \oplus C)$$

Trong đó: Y - giá trị chuỗi ngẫu nhiên đầu vào, M - dữ liệu đầu vào, C - dữ liệu sau khi được mã hóa, và G.Y - phép nhân ma trận giữa G và Y.

2.3. Thuật toán mã hóa/giải mã và xác thực GIFT-COFB 128-bit

Quá trình mã hóa/giải mã và xác thực GIFT-COFB 128-bit được xây dựng theo quy trình được chuẩn hóa, thực hiện tuần tự theo các bước: Khởi tạo (Initiate), xử lý dữ liệu liên kết (Process AD), xử lý bản tin đầu vào (Process PT), tạo thẻ Tag (Gen Tag) [11, 12], chi tiết như Hình 5.



Hình 5. Sơ đồ khối quá trình mã hóa/giải mã và xác thực của thuật toán GIFT-COFB 128-bit.

*** Quá trình mã hóa:**

Bước 1: Khởi tạo. $Y[0] \leftarrow E_K(N)$, $L \leftarrow Trunc_{(n/2)}(Y[0])$

+ $Y[0]$: Giá trị nonce, là một giá trị ngẫu nhiên hoặc duy nhất được sử dụng một lần trong các giao thức mã hóa hoặc giải mã, được mã hóa bằng hàm khối E_K.

+ L: Giá trị được lấy từ $Y[0]$, sử dụng hàm cắt $Trunc_{(n/2)}$ để lấy n/2 bit đầu tiên.

Bước 2: Đệm dữ liệu liên kết (A). Sử dụng hàm Pad(A) để biến đổi dữ liệu liên kết A thành các khối $A[1], A[2], \ldots, A[a]$.

Bước 3: Đệm cho dữ liệu đầu vào (M). Nếu $M \neq \epsilon$ (chuỗi rỗng), thì sử dụng hàm Pad(M) để biến đổi dữ liệu đầu vào M thành các khối $M[1], M[2], \ldots, M[m]$.

Bước 4: Xử lý dữ liệu liên kết (A).

Thực hiện vòng lặp từ i=1 đến a−1:

+ $L \leftarrow 2 \cdot L$ (*tính toán L nhân với 2*);

+ $X[i] \leftarrow A[i] \oplus G \cdot Y[i-1] \oplus L \| 0^{n/2}$ (*phép XOR kết hợp khối dữ liệu liên kết, hàm phản hồi và giá trị L*);

+ $Y[i] \leftarrow E_K(X[i])$ (*mã hóa X[i] để sinh Y[i]*);

+ Nếu |A| không chia hết cho n, thì: $L \leftarrow 3 \cdot L$ (*điều chỉnh giá trị L để xử lý khối cuối*).

Bước 5: Xử lý dữ liệu đầu vào (M).

Thực hiện vòng lặp từ $i = 1$ đến $m - 1$:

+ $L \leftarrow 2 \cdot L$;

+ $C[i] \leftarrow M[i] \oplus Y[i+a-1]$ (*sinh khối Ciphertext*);

+ $X[i+a] \leftarrow M[i] \oplus G \cdot Y[i+a-1] \oplus L \| 0^{n/2}$;

+ $Y[i+a] \leftarrow E_K(X[i+a])$.

Xử lý khối cuối $M[m]$:

+ Điều chỉnh L nếu |M| không chia hết cho n;

+ C[m] được tính từ $M[m] \oplus Y[a+m-1]$;

+ Tạo giá trị tag T: $T \leftarrow Trunc_{Y[a+m]}$.

Bước 6: Trả về kết quả mã hóa: Xử lý, và trả về kết quả đầu ra gồm: (C, T).

*** Quá trình giải mã:**

Bước 1: Khởi tạo: $Y[0] \leftarrow E_K(N)$, $L \leftarrow Trunc_{(n/2)}(Y[0])$

Bước 2: Thêm đệm (Padding) cho dữ liệu liên kết (A). Sử dụng hàm Pad(A) để biến đổi dữ liệu liên kết A thành các khối $A[1], A[2], \ldots, A[a]$.

Bước 3: Thêm đệm (Padding) cho Ciphertext (C). Nếu $C \neq \epsilon$, thì sử dụng hàm Pad(C) để biến đổi Ciphertext C thành thành các khối $C[1],C[2],\ldots,C[c]$.

Bước 4: Xử lý dữ liệu liên kết (A).

Thực hiện vòng lặp từ i=1 đến a−1:

+ $L \leftarrow 2 \cdot L$

+ $X[i] \leftarrow A[i] \oplus G \cdot Y[i-1] \oplus L \| 0^{n/2}$

+ $Y[i] \leftarrow E_K(X[i])$

Xử lý khối cuối của A nếu $|A|$ không chia hết cho n.

Bước 5: Xử lý Ciphertext (C).

Thực hiện vòng lặp từ i=1 đến c−1:

+ $L \leftarrow 2 \cdot L$;

+ $M[i] \leftarrow C[i] \oplus Y[i+a-1]$ (tính lại khối dữ liệu gốc từ Ciphertext);

+ $X[i+a] \leftarrow M[i] \oplus G \cdot Y[i+a-1] \oplus L \| 0^{n/2}$;

+ $Y[i+a] \leftarrow E_K(X[i+a])$.

Xử lý khối cuối cùng:

+ Điều chỉnh L nếu $|M|$ không chia hết cho n;

+ $M[c]$ được tính từ $C[c]$ và các giá trị trước đó;

+ Tính lại tag T' và so sánh với T.

Bước 6: Trả về kết quả giải mã. Nếu $T' = T$, đầu ra là M (dữ liệu gốc), ngược lại nếu $T' \neq T$, đầu ra là giá trị lỗi $\perp$ (không hợp lệ).

3. Thử nghiệm và đánh giá kết quả

Trong nghiên cứu này, giải thuật GIFT-COFB được triển khai trên nền tảng ngôn ngữ mô tả phần cứng Verilog HDL kết hợp với công nghệ FPGA. Môi trường phát triển sử dụng phần mềm Quartus II Version 4.2 của Altera để lập trình và cấu hình FPGA EP20K100TC144-3. Các chân I/O của FPGA được cấu hình theo chuẩn LVTTL/LVCMOS với điện áp hoạt động 2.5 V. Tín hiệu đầu vào và đầu ra của bộ mã hóa/giải mã được kết nối đồng bộ qua các chân GPIO nhằm đảm bảo sự ổn định và đồng bộ trong quá trình xử lý dữ liệu. Bảng phân bổ chân (Pin Assignment) chi tiết được trình bày trong Bảng 1.

Để đáp ứng yêu cầu xử lý song song và tối ưu hóa phần cứng, dữ liệu bản tin gốc (Plaintext) và dữ liệu liên kết (Associated Data) đều được cố định ở độ dài 128-bit. Thiết kế tích hợp đồng thời chức năng mã hóa và giải mã trên cùng một FPGA, cho phép đánh giá toàn diện hiệu năng và tính bảo mật của giải pháp. Quy trình thực hiện bao gồm lập trình, mô phỏng, và kiểm tra trên thiết bị thực tế nhằm hướng tới khả năng ứng dụng trong các hệ thống IoT đòi hỏi bảo mật cao và hiệu suất xử lý nhanh.

Bảng 1. Phân bổ chân và chuẩn tín hiệu trên FPGA

Pin Name/Usage	Location	Dir.	I/O Standard	Voltage	I/O Bank	User Assignment
VCC_INT	1	power		2.5V		
DataIn[0]	2	input	LVTTL/LVCMOS			Y
DataIn[1]	3	input	LVTTL/LVCMOS			Y
GND	4	gnd				
DataIn[2]	5	input	LVTTL/LVCMOS			Y
DataIn[3]	6	input	LVTTL/LVCMOS			Y
DataIn[4]	7	input	LVTTL/LVCMOS			Y
DataIn[5]	8	input	LVTTL/LVCMOS			Y
DataIn[6]	9	input	LVTTL/LVCMOS			Y
DataIn[7]	10	input	LVTTL/LVCMOS			Y
cs[0]	11	output	LVTTL/LVCMOS			Y

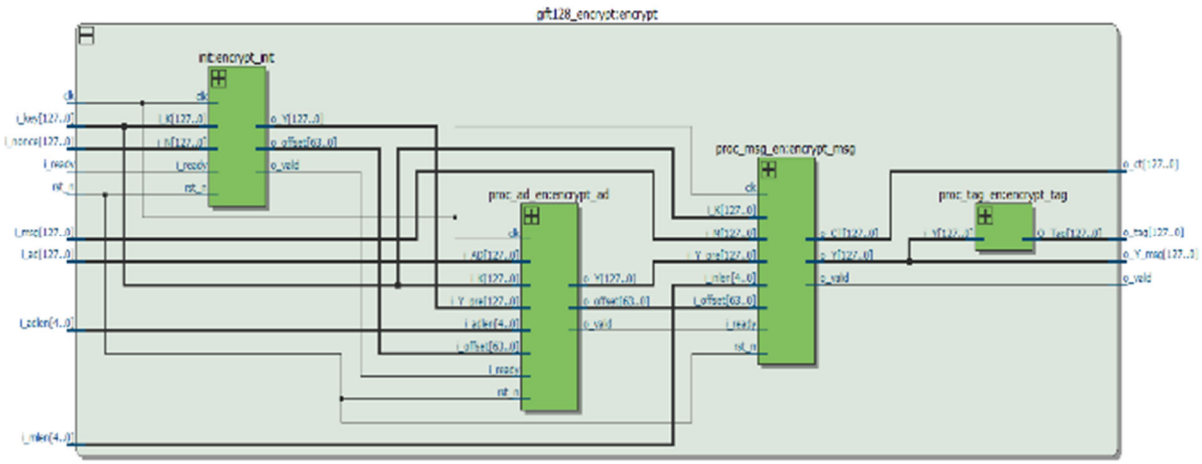
Hình bên dưới (Hình 6) trình bày nguyên lý tổ chức kiến trúc phần cứng của bộ mã hóa, nhằm thực hiện quy trình mã hóa và tạo thẻ Tag từ công cụ Quartus II, gồm đầy đủ các thành phần được mô tả trong thuật toán:

Khối **encrypt_init**: Khởi tạo (Initiate) giá trị ban đầu và giá trị offset để chuẩn bị đưa vào khối xử lý AD.

Khối **encrypt_ad**: Xử lý dữ liệu liên kết (Process_AD) dùng để kết hợp dữ liệu liên kết (AD) vào trong quá trình mã hóa.

Khối **encrypt_msg**: Xử lý bản mã đầu vào (Process_PT) dùng để mã hóa chuỗi dữ liệu ban đầu (Plaintext) thành chuỗi dữ liệu đã mã hóa (Ciphertext).

Khối **encrypt_tag**: Xử lý cuối cùng dùng để tạo ra thẻ Tag (Gen Tag) dùng để xác thực dữ liệu liên kết (AD) và chuỗi dữ liệu ban đầu (Plaintext). Nhờ đó mà nó đảm bảo được tính bảo mật và toàn vẹn của dữ liệu.



Hình 6. Quy trình mã hóa và tạo thẻ Tag

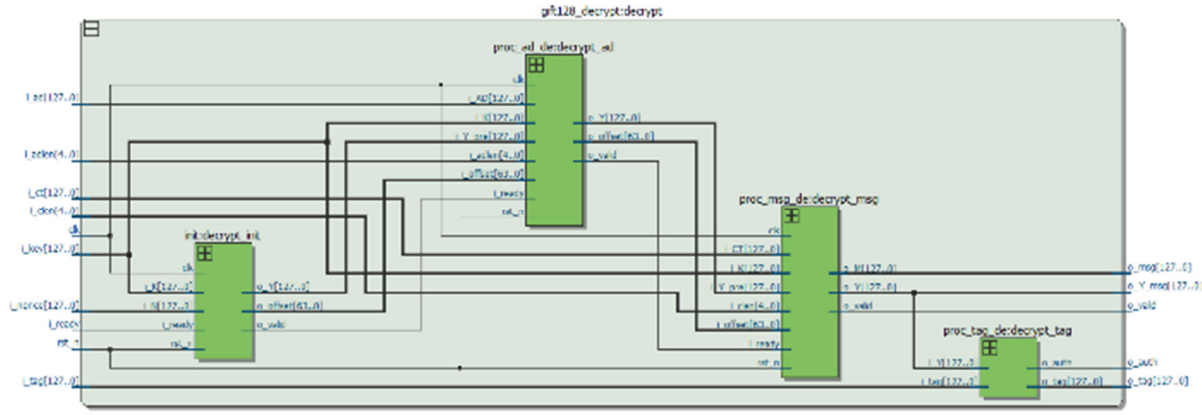
Tương tự, quy trình giải mã và xác thực dữ liệu từ công cụ Quartus II (Hình 7) cũng được xây dựng theo quy trình được chuẩn hóa, thực hiện tuần tự theo các bước:

Khối **decrypt_init**: Khởi tạo (Initiate) giá trị ban đầu và giá trị offset để chuẩn bị đưa vào khối xử lý AD.

Khối **decrypt_ad**: Xử lý dữ liệu liên kết (Process_AD) dùng để kết hợp dữ liệu liên kết (AD) vào trong quá trình giải mã.

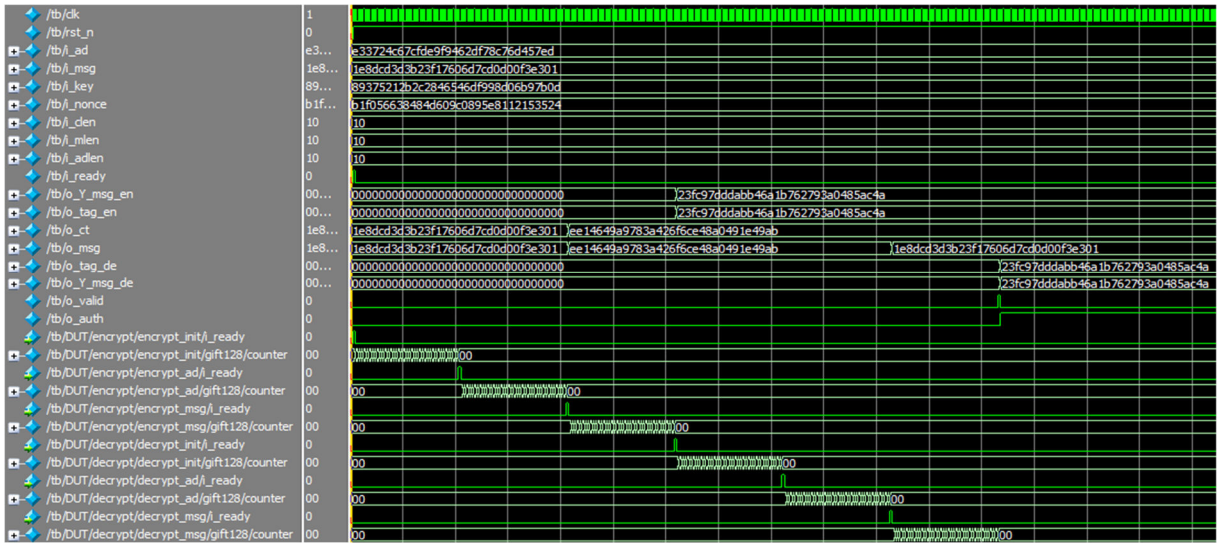
Khối **decrypt_msg**: Xử lý bản mã đã mã hóa dùng để giải mã chuỗi dữ liệu đã được mã hóa (Ciphertext) thành chuỗi dữ liệu ban đầu (Plaintext).

Khối **decrypt_tag**: Xử lý cuối cùng dùng để tạo ra thẻ Tag (Gen Tag) dùng để xác thực dữ liệu liên kết (AD) và chuỗi dữ liệu ban đầu (Plaintext). Nó được dùng để so sánh với thẻ Tag trong quy trình mã hóa, nếu hai thẻ Tag này giống nhau chứng tỏ dữ liệu liên kết (AD) và chuỗi dữ liệu ban đầu (Plaintext) không bị thay đổi trong quá trình truyền đi, ngược lại sẽ xác thực thất bại.

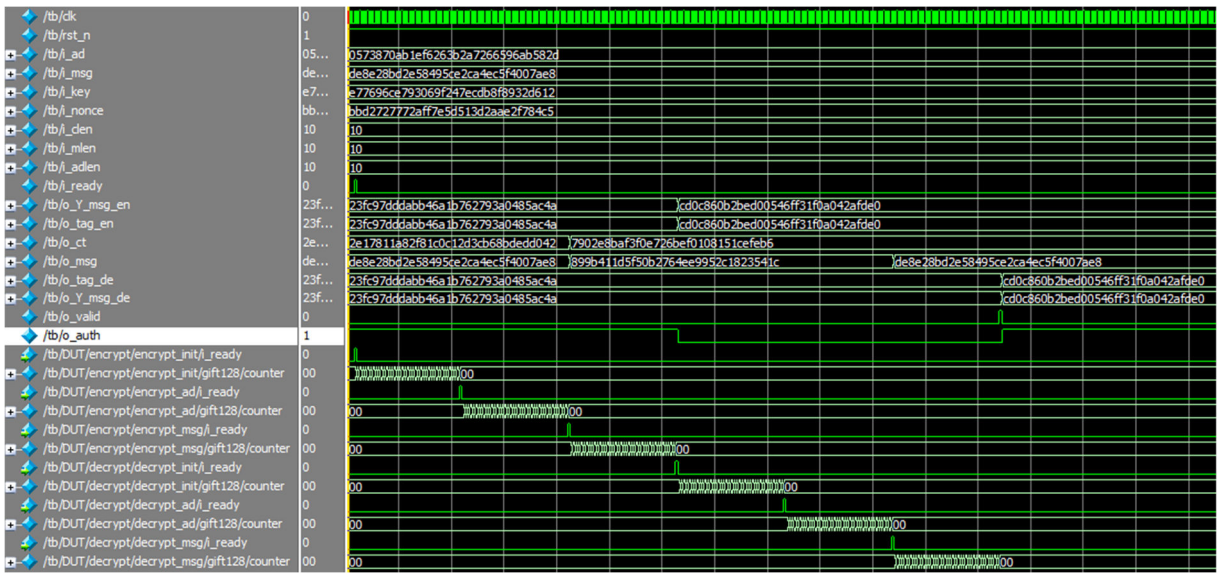


Hình 7. Quy trình giải mã và xác thực dữ liệu

Kết quả mô phỏng dưới đây được triển khai bằng Verilog HDL thực hiện toàn bộ quá trình mã hóa/giải mã và xác thực. Thực hiện hai quy trình kiểm thực với các chuỗi dữ liệu đầu vào như Plaintext, AD và Key được tạo ra bằng hàm ngẫu nhiên (Hình 8) và (Hình 9).



Hình 8. Kết quả kiểm thực lần 1



Hình 9. Kết quả kiểm thực lần 2

Bảng 2. Các giá trị vector vào\ra cho đánh giá thiết kế

	Dữ liệu kiểm thực lần 1	**Dữ liệu kiểm thực lần 2**
Plaintext	1E8DCD3C3B23F17606D7CD0D00F3E301	DE8E28BD2E58495CE2CA4EC5F4007AE8
Key	89375212B2C2846546DF998D06B97B0D	E77696CE793069F247ECDB8F8932D612
AD	E33724C67CFDE9F9462DF78C76D457ED	0573870AB1EF6263B2A7266596AB582C
Nonce	B1F056638484D609C0895E8112153524	BBD2727772AFF7E5D513D2AAE2F784C5
Cipherte xt	EE14649A9783A426F6CE48A0491E49AB	7902E8BAF3F0E726BEF0108151CEFEB6
Tag	23FC97DDDABB46A1B762793A0485AC4A	CD0C860B2BED00546FF31F0A042AFDE0
Plaintext (sau giải mã)	1E8DCD3C3B23F17606D7CD0D00F3E301	DE8E28BD2E58495CE2CA4EC5F4007AE8

Kết quả hiển thị cho thấy thuật toán mã hóa và giải mã hoạt động chính xác vì Plaintext đầu vào khớp hoàn toàn với Plaintext đầu ra sau khi qua quá trình mã hóa và giải mã, các Tag được tạo ra đúng cách và có thể dùng để kiểm tra tính xác thực của dữ liệu. Ngoài ra, toàn bộ quá trình mã hóa và giải mã mất 240 chu kỳ đồng hồ tương đương chỉ 24 us với tần số 10 MHz so với các thuật toán mã hóa phức tạp như RSA và ECC có thể lên đến 20 ms.

Từ kết quả thực hiện hai lần kiểm tra cho phép nhận xét rằng thuật toán GIFT-COFB 128-bit đã được triển khai thành công trên phần cứng và kiểm thực chính xác bằng mô phỏng. Hệ thống đáp ứng tốt yêu cầu về tính chính xác, bảo mật, và hiệu quả trong xử lý dữ liệu.

4. Kết luận

Đề tài đã trình bày cơ sở lý thuyết về thuật toán mã hóa sử dụng mã khối GIFT-128 kết hợp chế độ mã hóa xác thực COFB, cho phép nâng cao tính bảo mật và tính xác thực dữ liệu. Đã triển khai thành công trên phần cứng trên nền tảng ngôn ngữ mô tả phần cứng Verilog HDL và công nghệ FPGA, kết quả kiểm thực thuật toán GIFT-COFB 128-bit đáp ứng tốt về tính bảo mật và toàn vẹn của dữ liệu, cũng như hiệu suất xử lý tính toán, phù hợp cho các ứng dụng IoT có tài nguyên và khả năng xử lý tính toán phần cứng hạn chế.

Tài liệu tham khảo

[1] Turan, M.S., McKay, K., Chang, D., Bassham, L.E., Kang, J., Waller, N.D., Kelsey, J.M., & Hong, D. (2023). *Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process*. NIST Interagency/Internal Report (NIST IR) 8454. DOI: 10.6028/NIST.IR.8454.

[2] Vũ, L.V.T., & Minh, T.N. (2023). "Nghiên cứu và thực thi bộ mã hóa bảo mật theo giải thuật COMET với khối 128-bit". *Tạp chí Khoa học công nghệ - Trường Đại học Khoa học Huế*, Tập 23, Số 1.

[3] Liu, G., Yu, Q., Tang, L., Ma, S., Wei, C., Jia, K., Qin, L., Dong, X., & Shen, Y. (2023). "Ultra Low-Latency Block Cipher uLBC". *IACR Comms. in Cryptology*, 1(4), Article 25.

[4] Stallings, W. (2023). *Cryptography and Network Security: Principles and Practice* (8th Edition). Pearson.

[5] Elsadek, I., Ghonem, A., Abouzeid, S., Wallrabenstein, J., MacLean, E., Gardner, D., Aftabjahani, S., Cammarota, R., & Tawfik, E. (2025). "*Benchmarking NIST LWC Candidates Over GF22FDx Achieving 3.5 Tb/J and 4.4 Gbps for IoT Applications*". SSRN Preprint, May 30 2025. DOI: 10.2139/ssrn.5276162.

[6] Baksi, A., Breier, J., Chattopadhyay, A., Gerlich, T., Guilley, S., Gupta, N., Isobe, T., Jati, A., Jedlička, P., Kim, H., et al. (2023). "BAKSHEESH: Similar Yet Different From GIFT – Introducing a Lightweight Cipher". *IACR Cryptol. ePrint Arch.*, 2023:750.

[7] Buchanan, W.J., & Maglaras, L. (2023). "*Review of the NIST Lightweight Cryptography Finalists*". arXiv Preprint, arXiv:2303.14785 [cs.CR]. DOI: 10.48550/arXiv.2303.14785.

[8] Kaur, J., Cintas Canto, A., Mozaffari Kermani, M., & Azarderakhsh, R. (2023). "*A Comprehensive Survey on the Implementations, Attacks, and Countermeasures of the Current NIST Lightweight Cryptography Standard*". arXiv Preprint, arXiv:2304.06222 [cs.CR]. DOI: 10.48550/arXiv.2304.06222.

[9] Mohajerani, K., Beckwith, L., Abdulgadir, A., Ferrufino, E., Kaps, J.-P., & Gaj, K. (2023). "SCA Evaluation and Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process". *IACR Cryptol. ePrint Arch.*, 2023:484.

[10] Yadav, T., Kumar, M., Kumar, A., & Pal, S.K. (2025). "A Practical-Quantum Differential Attack on Block Ciphers". *Cryptography and Communications*, 17(2), 337–357. DOI: 10.1007/s12095-023-00650-6.

[11] Sun, X., Zhang, W., Rodríguez, R., & Liu, H. (2024). Known-Key Attack on GIFT-64 and GIFT-64 [g 0 c] Based on Correlation Matrices. In *Proc. 29th Australasian Conf. on Info. Security and Privacy (ACISP 2024)*, Springer LNCS (pp. 20–40).

[12] Inoue, A., Guo, C., & Minematsu, K. (2023). "Nonce-Misuse Resilience of Romulus-N and GIFT-COFB". *IET Information Security*, 17(3), 468–484. DOI: 10.1049/ise2.12110.