Chosen-Plaintext Attack on Energy-Efficient Hardware Implementation of GIFT-COFB

Yadi Zhong and Ujjwal Guin
Dept. of Electrical and Computer Engineering, Auburn University, AL, USA
{yadi, ujjwal.guin}@auburn.edu

***Abstract*—Electronic devices are connected now more than ever with the prevalence of the Internet of Things. The ever-increasing communication between these lightweight devices presses for the need to embed a cryptographic mechanism to ensure the confidentiality and authentication of data. Lightweight cryptography nicely supports the need for encryption mechanisms on IoT devices with limited memory, storage, and computing capability. This paper focuses on breaking the hardware implementation of GIFT-COFB, one of NIST's Lightweight Cryptography finalists. The 2-round partial unrolled design of GIFT-COFB is shown to be the most energy-efficient among all other r-round partial unrolling and fully unrolled settings [1]. In this paper, we propose a chosen-plaintext attack to retrieve the master key K effectively and demonstrate the feasibility of our proposed attack on the 2-round partial unrolled GIFT-COFB. Our efficient attack can derive the secret key by exploiting the nonlinearity of the Sboxes with a worst-case complexity of $O(2^4)$.**

***Index Terms*—GIFT-COFB, Sbox, partial unrolling**

I. Introduction

The recent advancement of the Internet of Things (IoT) results in more connected electronic devices than ever. Vast chunks of data are being transferred over the unsecured channel for increasingly ubiquitous computing. This may give rise to the potential breach of confidentiality, integrity, and authentication if, somehow, the devices transmit information without the proper protection mechanism. As IoT devices are resource-constrained and low-cost, have limited area, and less computation power, the previously standardized Advanced Encryption Standard (AES) block cipher is not suited for these devices. Thus, NIST instantiated the process to standardize lightweight cryptographic algorithm, stressing its importance on RFID tags, sensor nodes, industrial controllers, and smart cards [2]. Among all the candidates submitted to NIST, ten were selected as the finalists [3]. All finalists ensured tight security bounds and ensured efficient implementation in both hardware and software. One of the ten finalists is GIFT-COFB [4], which integrates combined-feedback mode (COFB) with GIFT-128 cipher [5] to offer authenticated encryption with associated data.

GIFT belongs to the Substitution-Permutation Networks (SPNs), which utilizes 4-bit Sbox and bit permutation PermBits as the underlying SPN. Compared to the lightweight block cipher PRESENT [6], GIFT provides better efficiency in both area and performance as well as mitigates the vulnerability to the linear hull attack [7] against PRESENT. Although there have been a few attacks against GIFT cipher, such as cache attack [8], side-channel attack [9], and fault-injection attack [10], GIFT-COFB has shown to be secure against large encryption queries [11]–[13]. Various forgery attacks described in [11]–[13] requires the attacker to perform either $O(2^{64})$ encryption or decryption blocks to break GIFT-COFB's authentication. As lightweight cryptography is implemented on IoT devices, it is equally important to examine the security bound for the same authenticated encryption in its hardware implementation.

In this paper, we propose a chosen-plaintext attack on the most-energy-efficient hardware implementation of GIFT-COFB [1]. As an attacker can have access to the IoT device, he/she can apply a nonce to the GIFT-COFB encryption oracle and observe the 2-round update from the 128-bit output tag. Multiple nonce-tag pairs can be captured through resetting the oracle and assigning new values to the input nonce. With several nonce-tag pairs, the adversary can recover both round keys of round 1 and 2, and use them to derive the 128-bit secret key, K, through the reverse of key schedule. The contributions of this paper are summarized as follows:

- We propose an efficient chosen-plaintext attack on the 2-round partial unrolled hardware implementations of GIFT-COFB. The overall complexity of this attack is on the order of $O(2^4)$. This attack can be extended to other partial unrolled variants of GIFT-COFB.
- We perform quantitative analysis on GIFT's SBox. The construction of the underlying 4-bit Sbox is crucial for breaking GIFT-COFB and yielding the optimal three nonce-tag pairs used for the recovery of key K.

The rest of the paper is organized as follows. We briefly introduce the background for hardware implementations of GIFT-COFB and describe the threat model in Section II. Our proposed attack is presented in Section III. The

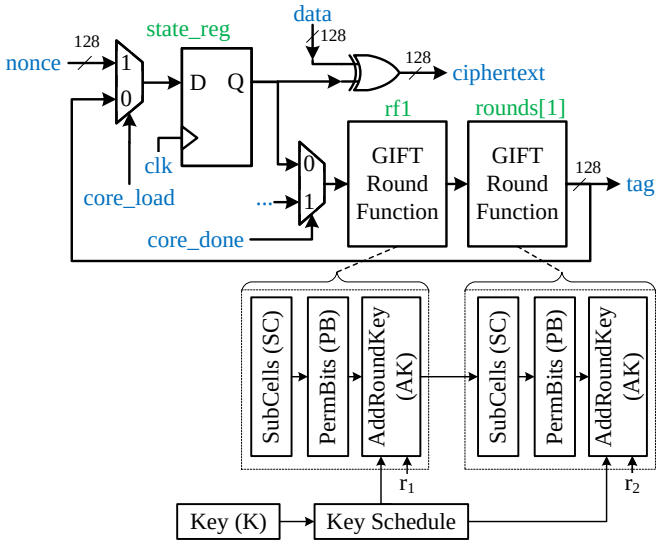

Figure 1: Hardware implementation of 2-round partial unrolled GIFT-COFB [1], [14].

result for the proposed attack is described in Section IV. Finally, we conclude the paper in Section V.

II. Background

As IoT devices are resource-constrained, having multicycle hardware implementations of lightweight ciphers on these embedded systems offer better area, power, and performance efficiency. It offsets the inherent limitations of these devices. Each clock cycle finishes one round of encryption, storing the intermediate result to the round registers. However, the major weakness in multicycle hardware implementation of crypto primitives is that the output for each encryption round is directly assigned to the ciphertext variable [15]. Consequently, by saving the result straight to the ciphertext, the adversary is granted access to the internal rounds, where the intermediate round result is updated at every clock cycle.

A. Threat Model

The threat model is given to defining the capabilities and intentions of an adversary, and is summarized as follows:

- An adversary has access to a fully functional IoT device where the secret key K is stored in the non-volatile memory (NVM). The possession of this device allows the adversary to perform the chosen-plaintext attack and observe the corresponding ciphertext.
- The adversary can also acquire the gate-level netlist of the corresponding cipher implemented in the IoT device. It can be extracted either through IC reverse engineering [16] or from GDSII files [17]. An untrusted foundry or a rouge employee of an SoC design house can pirate the GDSII file to an adversary.

B. Lightweight Hardware Implementation of GIFT-COFB

Under the requirement of authenticated encryption [2], GIFT-COFB supports the encryption of a fixed-length nonce, followed by variable-length associated data and variable-length plaintexts. GIFT-128 [5] block cipher, which consists of 40 encryption rounds is used in GIFT-COFB [1], [14]. Although the typical architecture for the multicycle implementation of block ciphers is to compute every round per clock cycle, it would be sub-optimal for GIFT-COFB. The computation of ciphertext could incur undesired latency for GIFT-COFB in authenticated encryption compared to other finalists whose underlying block ciphers are of a smaller round size [1]. It is possible to incorporate multiple rounds into one clock cycle at the cost of increased area utilization in replicating the round function. Caforio et al. [1] examined different partial r-round unrolling scenarios and the fully unrolled setting of GIFT-COFB and found that the minimum energy consumption of 0.251 nJ/128-bits is observed when two encryption rounds finish at each clock cycle ($r = 2$) along with clock gating and register borrowing.

We focus on this hardware implementation of GIFT-COFB. The datapath of interest is shown in Figure 1. Inputs, outputs, and wire names are presented in blue and the module instantiation names are in green. At the start of the authenticated encryption, GIFT-COFB loads a 128-bit nonce into the 128-bit register *state_reg* with selection bit *core_load* at logic 1 for the first clock cycle. The output from *state_reg* is XORed with the 128-bit input data (either the associated data or plaintext) to generate the ciphertext. After the nonce is passed to the *state_reg*, the control signal *core_load* is changed to 0. The default value for control signal *core_done* is logic 0, which allows the output from *state_reg* to pass through the multiplexer and it updates to logic 1 when the encryption reaches the 40^{th} round. Two GIFT round functions, *rf1* and *rounds[1]*, are serially connected. Each round function has two other inputs, the 6-bit round constants and the 128-bit round key, whose connections are not shown in Figure 1. The 128-bit *tag* receives the output from round function *rounds[1]*. These details can be found in *aead.vhd* and *controller.vhd* [14].

GIFT-COFB begins authenticated encryption by loading nonce and encrypting it with GIFT cipher and the master key K at the speed of 2 encryption rounds per clock cycle. Hence, the attacker could observe the second round result of GIFT's encryption of nonce or any consecutive 2-round (before the reach of 40^{th} round) through output variable *tag*. The remaining question for the adversary is to recover the secret key K under the 2-round encryption of GIFT. Without the loss of generality, we analyze the first two rounds of nonce encryption to explain our attack methodology.

III. Chosen-Plaintext Attack on GIFT-COFB

Lightweight crypto modules (e.g., GIFT-COFB) are designed to support authenticated encryption for low-cost IoT devices, which can be deployed in diverse locations. These devices can easily be accessed by an adversary, who can launch an attack to extract the secret key. This section presents a novel chosen-plaintext attack that breaks GIFT-COFB on the order of $O(2^4)$. Our attack can recover 4-bit key group in parallel even with the nonlinearity in Sboxes. Let us begin by analyzing the round function to launch the attack.

GIFT round function consists of three steps, SubCells (SC), PermBits (PB), and AddRoundKey (AK), as shown in the dashed boxes in Figure 1. SubCells comprises 32 Sboxes, where each 4-bit input cell is transformed by one 4-bit Sbox ($\mathtt{s}(\cdot)$), a non-linear and bijective mapping in $GF(2^4)$. PermBits performs bit-level permutation, and AddRoundKey is the modulo-2 addition of the 128-bit from PermBits with a 64-bit round key at bit location $[95...32]$. A 6-bit constant is XORed with bit locations 23, 19, 15, 11, 7, and 3, and constant 1 is XORed with bit location 127. We denote the 128-bit input nonce as N and the 128-bit output after 2 rounds as T, the 64-bit round key associated with the round functions as $\widehat{K^1}$ and $\widehat{K^2}$, and the corresponding round constants as $\widehat{r^1}$ and $\widehat{r^2}$. In uniformity with the 128-bit vector expressions below, we extend the 64-bit round keys, $\widehat{K^1}$, $\widehat{K^2}$, to 128-bit K^1 and K^2 by zero-padding into the remaining bit locations. Similarly, we expanded $\widehat{r^1}$ and $\widehat{r^2}$ to the zero-padded 128-bit r^1 and r^2. In the following, we present the detailed steps to obtain the secret key K.

- *Step 1 – Sample collection:* Two tags corresponding to two nonces are collected. The adversary first chooses a nonce, N, passes it to the oracle, and captures the corresponding output tag, T, after the first clock cycle of authenticated encryption. As two encryption rounds are performed in one clock cycle, we can compute the tag as:

$$T = \overbrace{\mathtt{PB}(\mathtt{SC}(\underbrace{\mathtt{PB}(\mathtt{SC}(N)) \oplus K^1 \oplus r^1}_{\text{1st round}})) \oplus K^2 \oplus r^2}^{\text{2nd round}} \quad (1)$$

  Since SubCells and PermBits are deterministic, the adversary can calculate $\mathtt{PB}(\mathtt{SC}(N))$ directly. We denote the XOR of r^1 and $\mathtt{PB}(\mathtt{SC}(N))$ as Q, where $Q = \mathtt{PB}(\mathtt{SC}(N)) \oplus r^1$. We can then rewrite Equation 1 as:

$$T = \mathtt{PB}(\mathtt{SC}(Q \oplus K^1)) \oplus K^2 \oplus r^2 \quad (2)$$

  Under a different input nonce N', $N' \neq N$, the adversary can obtains the 2-round output T' and derive the corresponding $Q' = \mathtt{PB}(\mathtt{SC}(N')) \oplus r^1$ as seen earlier:

$$T' = \mathtt{PB}(\mathtt{SC}(Q' \oplus K^1)) \oplus K^2 \oplus r^2 \quad (3)$$

  For the adversary, this can be achieved by resetting the GIFT-COFB crypto-system and then providing a difference nonce, N'.

- *Step 2 – Dependency Removal of round key K^2:* The adversary removes the dependency of the second round key K^2 by XORing Equation 2 and 3,

$$T \oplus T' = \mathtt{PB}(\mathtt{SC}(Q \oplus K^1)) \oplus \mathtt{PB}(\mathtt{SC}(Q' \oplus K^1)) \quad (4)$$

  The linearity of PermBits with its additive property allows us to rewrite Equation 4 as,

$$T \oplus T' = \mathtt{PB}(\mathtt{SC}(Q \oplus K^1) \oplus \mathtt{SC}(Q' \oplus K^1)). \quad (5)$$

  Since bit permutation in PermBits is reversible, it is straightforward for an adversary to derive back to the output from SubCells (SC),

$$\mathtt{PB}^{-1}(T \oplus T') = \mathtt{SC}(Q \oplus K^1) \oplus \mathtt{SC}(Q' \oplus K^1) \quad (6)$$

  where $\mathtt{PB}^{-1}$ is the inverse bitwise permutation of PB. Since $\mathtt{PB}^{-1}(T \oplus T')$ is a constant, we denote it as $L = \mathtt{PB}^{-1}(T \oplus T')$.

- *Step 3 – Decomposition of K^1 to 16 key cells:* Equation 6 can be further split into 32 4-bit cells. The middle sixteen cells can be represented as

$$L_j = \mathtt{s}(Q_j \oplus K^1_j) \oplus \mathtt{s}(Q'_j \oplus K^1_j) \quad (7)$$

  where cell index j satisfies $j \in \{23, 22, ..., 16\}$ since the 64-bit round key is only effective at bit indices $[95...32]$. However, it is not possible to uniquely determine each key cell K^1_j with Q_j and Q'_j, where $Q_j \neq Q'_j$, derived from two noncea N and N', where $N \neq N'$ and Equation 7 alone, where different values for a key cell K^1_j could lead to the same L_j (see details in Section IV).

- *Step 4 – Recovery of each key cell in K^1:* The adversary can now restart the GIFT-COFB with a third nonce, N'', and acquire its *tag* T''. Repeating Steps 1-3, the attacker obtains $Q'' = \mathtt{PB}(\mathtt{SC}(N'')) \oplus r^1$, $L' = \mathtt{PB}^{-1}(T \oplus T'')$, and

$$L'_j = \mathtt{s}(Q_j \oplus K^1_j) \oplus \mathtt{s}(Q''_j \oplus K^1_j). \quad (8)$$

  With carefully chosen nonces $\{N, N', N''\}$, each key cell K^1_j can be uniquely recovered with Equation 7 and 8 by exhaustive search on all 16 combinations of 4-bit key, as shown in Section IV. The attacker can compute all sixteen key cells in K^1 in parallel, since each 4-bit key has its own Equation 7 and 8 independent of other key bits.

- *Step 5 – Recovery of K^2 and the secret key K:* After deriving the entire round key K^1, the second round key K^2 can be simply obtained through

$$K^2 = T \oplus \mathtt{PB}(\mathtt{SC}(\mathtt{PB}(\mathtt{SC}(N)) \oplus K^1)) \oplus r^2, \quad (9)$$

where T, $\texttt{PB}(\texttt{SC}(\texttt{PB}(\texttt{SC}(N))$, and K^1 are all known to the attacker. Then, the master key K is recovered through the reversing of key schedule [5] with round keys K^1 and K^2, $K = \{K^2_{[95..64]}||K^1_{[95..64]}||K^2_{[63..32]}||K^1_{[63..32]}\}$, where $||$ denotes the concatenation of bit vectors. The complexity of this attack lies in the recovery of K^1, not K^2. Thus, the attack fully deciphers the secret key K.

IV. Result

In this section, we quantitatively analyze the number of different nonces required for the attacker to derive the secret key K based on the construction of GIFT's Sbox. The computation complexity arises primarily from the computation of round key K^1, since K^2 is computed through Equation 9 with $\boldsymbol{O}(1)$. First, we show that having two nonce-tag pairs is insufficient to derive the value for any key cells. When we obtain two different nonce-tag pairs, we can perform Steps 1-3 and proceed to Equation 7 of Section III, where each key cell can be solved independently. To cover all the possibilities of Q_j and Q'_j, we exhaustively search all 16 combinations of K^1_j under every possible pair of $\{Q_j, Q'_j\}$ with $Q_j \neq Q'_j$. From our simulation [18], under any fixed $\{Q_j.Q'_j\}$, where $Q_j \neq Q'_j$, at least two solutions exist for a key cell K^1_j giving the same value of L_j.

On the other hand, if the attacker applies three nonce N, N', and N'' and obtains the corresponding tag output, he/she subsequently gets Equation 7-8 for each key cell from Steps 1-4. Out of all possible, $\binom{16}{3} = 560$, pairs of $\{Q_j, Q'_j, Q''_j\}$, where $Q_j \neq Q'_j \neq Q''_j$, there are 416 pairs that give different $\{L_j, L'_j\}$ for all 16 possible key cells, allowing the adversary to uniquely determine each 4-bit key [18]. This enables the recovery of one key cell through the exhaustive search of $2^4 = 16$ possible combinations to find the correct solution satisfying Equation 7 and 8. All sixteen key cells in K^1 can be solved separately and in parallel, resulting in the time complexity of $O(2^4)$ on searching the correct key value. The construction of nonce can be easily derived under the desired values for Q's with $N = \texttt{SC}^{-1}(\texttt{PB}^{-1}(Q \oplus r^1))$.

V. Conclusion

This paper presents a novel chosen-plaintext attack on the energy-efficient hardware implementation of GIFT-COFB, one of the finalists for NIST's Lightweight Cryptography. The proposed attack exploits the 2-round instantiation of GIFT block cipher inside the partial unrolled structure of GIFT-COFB. The entire secret key can be solved with a minimum of three *nonce-tag* pairs. The key cells can be recovered in parallel by solving the corresponding constraint equations, resulting in a $O(2^4)$ worst-case complexity.

References

[1] A. Caforio, F. Balli, and S. Banik, "Energy Analysis of Lightweight AEAD Circuits," in *International Conference on Cryptology and Network Security*. Springer, 2020, pp. 23–42.

[2] L. Bassham, Ç. Çalık, K. McKay, and M. S. Turan, "Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process," *US National Institute of Standards and Technology*, 2018.

[3] NIST CSRC, Finalists - Lightweight Cryptography, https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[4] S. Banik, A. Chakraborti, T. Iwata, K. Minematsu, M. Nandi, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT-COFB," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 738, 2020.

[5] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, "GIFT: A Small Present," in *International Conference on Cryptographic Hardware and Embedded Systems*. Springer, 2017, pp. 321–345.

[6] A. Bogdanov, L. R. Knudsen, G. Leander, C. Paar, A. Poschmann, M. J. Robshaw, Y. Seurin, and C. Vikkelsoe, "Present: An ultra-lightweight block cipher," in *International workshop on cryptographic hardware and embedded systems*. Springer, 2007, pp. 450–466.

[7] J. Nakahara, P. Sepehrdad, B. Zhang, and M. Wang, "Linear (Hull) and Algebraic Cryptanalysis of the Block Cipher PRESENT," in *International Conference on Cryptology and Network Security*. Springer, 2009, pp. 58–75.

[8] C. Reinbrecht, A. Aljuffri, S. Hamdioui, M. Taouil, and J. Sepúlveda, "GRINCH: A Cache Attack against GIFT Lightweight Cipher," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 549–554.

[9] S. Patranabis, N. Datta, D. Jap, J. Breier, S. Bhasin, and D. Mukhopadhyay, "SCADFA: Combined SCA+DFA Attacks on Block Ciphers with Practical Validations," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1498–1510, 2019.

[10] H. Luo, W. Chen, X. Ming, and Y. Wu, "General differential fault attack on present and gift cipher with nibble," *IEEE Access*, vol. 9, pp. 37 697–37 706, 2021.

[11] A. Inoue and K. Minematsu, "Gift-cofb is tightly birthday secure with encryption queries," *Cryptology ePrint Archive*, 2021.

[12] M. Khairallah, "Observations on the tightness of the security bounds of gift-cofb and hyena." *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 1463, 2020.

[13] ——, "Security of cofb against chosen ciphertext attacks," Cryptology ePrint Archive, Report 2021/648, Tech. Rep., 2021.

[14] Energy Analysis of Lightweight AEAD Circuit, https://github.com/qantik/Energy-Analysis-of-Lightweight-AEAD-Circuit.

[15] A. Aghaie, A. Moradi, S. Rasoolzadeh, A. R. Shahmirzadi, F. Schellenberg, and T. Schneider, "Impeccable circuits," *IEEE Transactions on Computers*, vol. 69, no. 3, pp. 361–376, 2019.

[16] S. E. Quadir, J. Chen, D. Forte, N. Asadizanjani, S. Shahbazmohamadi, L. Wang, J. Chandy, and M. Tehranipoor, "A Survey on Chip to System Reverse Engineering," *ACM journal on emerging technologies in computing systems (JETC)*, vol. 13, no. 1, pp. 1–34, 2016.

[17] R. Torrance and D. James, "The State-of-the-Art in IC Reverse Engineering," in *International Workshop on Cryptographic Hardware and Embedded Systems*, 2009, pp. 363–381.

[18] Exhaustive Search on Key Cells of First Round Key of GIFT-COFB, https://github.com/yadi14/exhaustive-search-on-gift-key.