GIFT-COFB: Lightweight Serial Architectures

Andrea Caforio[1], Daniel Collins[1], **Subhadeep Banik**[2], Francesco Regazzoni[2,3]

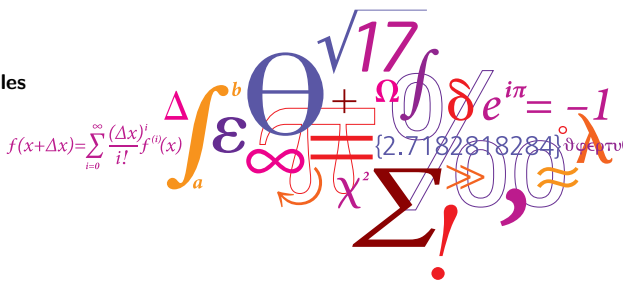
[1]LASEC, EPFL Switzerland

[2]Univesita della Svizzera Italiana, Lugano, Switzerland

[3]University of Amsterdam, Netherlands

Cryptarchi Workshop 2022, Porquerolles

31st May 2022



Outline

- Introduction
- Circuit Idea
- Faster Circuit
- Threshold implementation

Introduction

GIFT-COFB

- GIFT-COFB: Lightweight AEAD Scheme
- Submission to NIST Lightweight Cryptography Competition.
  - → Currently a 3rd Round Candidate.
  - → Along with 9 other candidates.
- Extremely Lightweight architecture.
- Requires one storage element with block cipher.

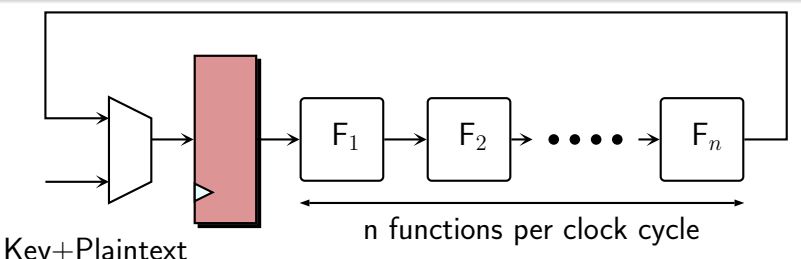
Implementation of Block Ciphers: Round-unrolled

Round unrolled architectures

- n round functions per clock cycle.
- Requires $\approx R/n$ clock cycles to complete execution.

  $\rightarrow$ Higher $n \rightarrow$ Higher throughput.

  $\rightarrow$ Higher $n \rightarrow$ Higher area/power consumption.



Implementation of Block Ciphers: Serial

Serial Architectures

- Reduce datapath width + reduce resources
- Ex: AES circuit of Moradi et al (Eurocrypt 2011)

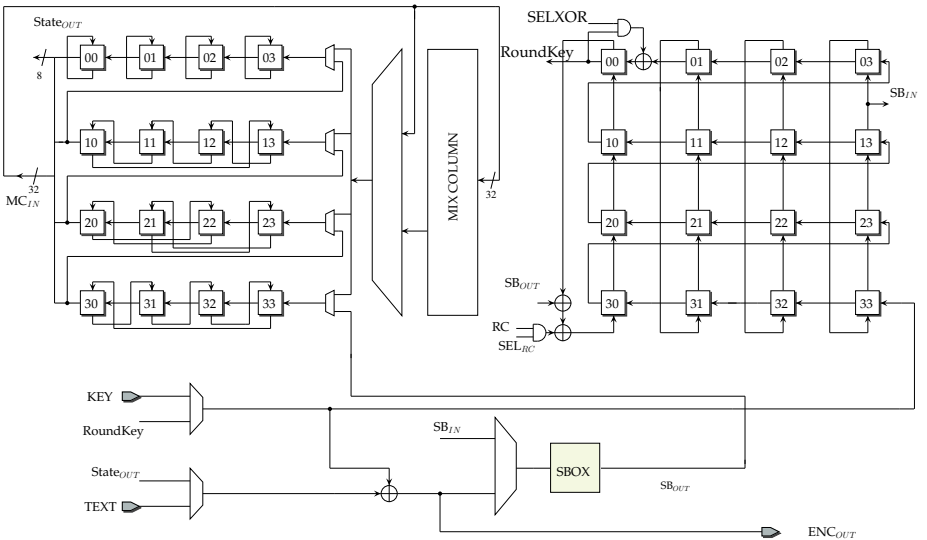
  → One S-box+One Mixcolumn circuit+8-bit datapath.

  → All multiplexers and xor gates need to be of 8-bits.

  → 21 cycles per round → 226 cycles per encryption .

  → Only 2400 GE silicon area.

Implementation of Block Ciphers: Serial



Implementation of Block Ciphers: Bit-Serial

Bit-Serial

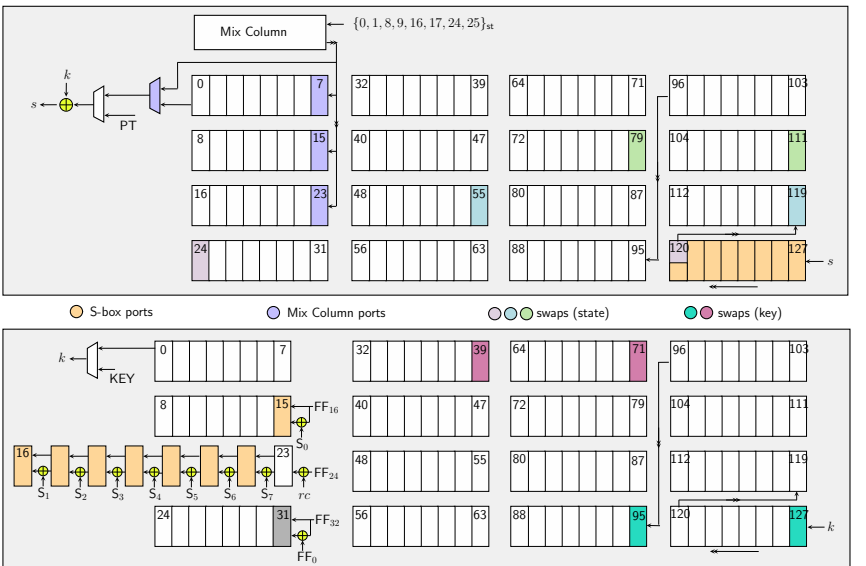
- Datapath width reduced to a single bit !!.
- All component gates need to be of 1-bit.
- Reduce area footprint at the cost of throughput.

  → Ex: AES circuit of Balli et al (IACR TCHES 2021).

  → 128 cycles per round, 1408 cycles per encryption.

  → 1267 GE silicon Area.

Implementation of Block Ciphers: Bit-Serial



Some results on Bit Serial Implementations

Serial Architectures

- Only one pair of Scan-FF (FF + multiplexer) can generate any permutation

  → Result first proven in Banik et al (IACR TOSC 2020).

  → Any Swap + 1 bit rotate generates entire Permutation group.

- However many take many cycles to compute

  → PRESENT block cipher: One swap → 1472 cycles per round.

  → 3 swaps → 64 cycles per round

GIFT 128

GIFT-128 [Banik et al CHES 2017]

- Lightweight block cipher appeared in CHES 2017.
- Bit permutation as the linear layer
- Initially designed keeping in mind software efficiency
- Efficient in Software and hardware platforms

GIFT 128: Bitserial architecture

GIFT 128

Permutation structure

$$S_3 = x_{127}x_{126}\cdots x_{97}x_{96}, \qquad S_2 = x_{95}x_{94}\cdots x_{65}x_{64},$$
$$S_1 = x_{63}x_{62}\cdots x_{33}x_{32}, \qquad S_0 = x_{31}x_{30}\cdots x_1x_0.$$

The bit permutation Π now reduces to four independent sub-permutations $\Pi_3, \Pi_2, \Pi_1, \Pi_0$ that act on each lane

$$\Pi(x_{127}\cdots x_0) = \Pi_3(x_{127}\cdots x_{96})\Pi_2(x_{95}\cdots x_{64})\Pi_1(x_{63}\cdots x_{32})\Pi_0(x_{31}\cdots x_0).$$

GIFT 128

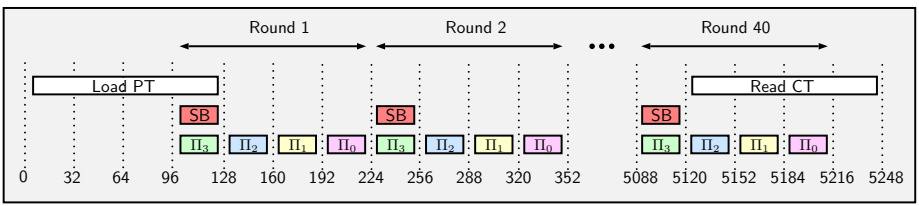
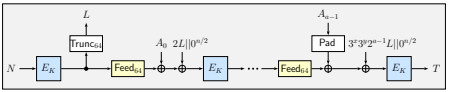
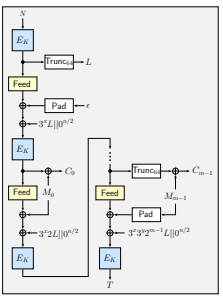


Figure: Timeline diagram of the *swap-and-rotate* GIFT-128 implementation; the numbers in the x-axis denote clock cycles.

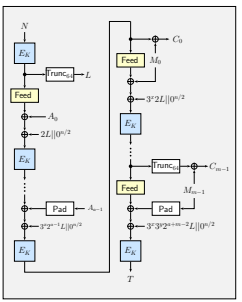
GIFT-COFB



(a) $A \neq \epsilon,\ M = \epsilon$



(b) $A = \epsilon,\ M \neq \epsilon$

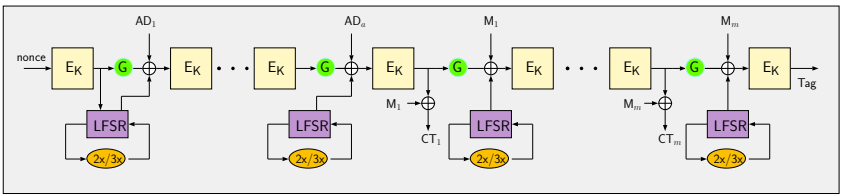


(c) $A \neq \epsilon,\ M \neq \epsilon$

COFB Flow

- Encryption interspersed by 3 operations: Feed, Update L, Add to state.

GIFT-COFB



Main Idea

- Encrypt
- Update L
- Perform Feed on state + Absorb message/AD
- Add together

GIFT-COFB

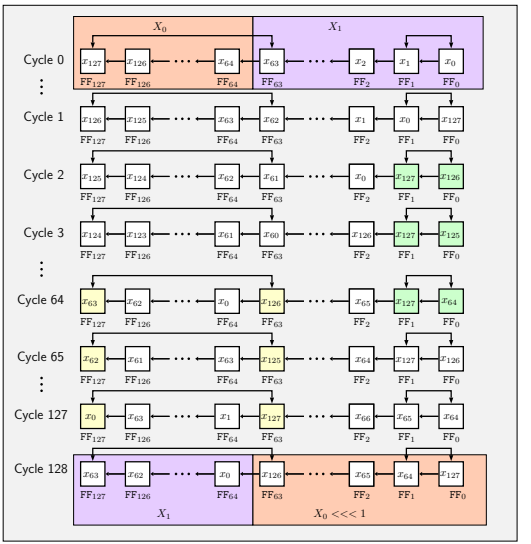
Feed

$$\mathsf{Feed}(X) = (X_1, X_0 \lll 1).$$

$L \rightarrow 2^x 3^y L$

$$x = \begin{cases} 1 & \text{if } |A| \bmod n = 0 \text{ and } A \neq \epsilon, \\ 2 & \text{otherwise;} \end{cases} \quad y = \begin{cases} 1 & \text{if } |M| \bmod n = 0 \text{ and } M \neq \epsilon, \\ 2 & \text{otherwise.} \end{cases}$$

Feed can be done in 128 cycles



Feed + Absorb can be done in 128 cycles

Feed+Absorb

• Note that if $X = x_{127}x_{126}\cdots x_0$ and $M = m_{127}m_{126}\cdots m_0$, then the i-th bit u_i of Feed$(X) \oplus M$ is given as:

$$u_i = \begin{cases} m_i \oplus x_{i-64} & \text{if } 64 \leq i < 128, \\ m_i \oplus x_{i+63} & \text{if } 0 < i \leq 63, \\ m_i \oplus x_{127} & \text{if } i = 0 \end{cases}$$

• For any i, the state bit x_{i-64} (for $64 \leq i < 128$), x_{i+63} (for $0 < i \leq 63$) and x_{127} (for $i = 0$) is always present at FF_{63} at clock cycle i.

• Thus, to implement the above, we need one additional XOR gate before the 63rd flip-flop in the state register.

Finite field multiplication

2x and 3x over 64 cycles

- We have

$$2 \times l_{63}l_{62}\cdots l_0 \ = (L \ll 1) \oplus (l_{63} * 0^{59}11011)$$

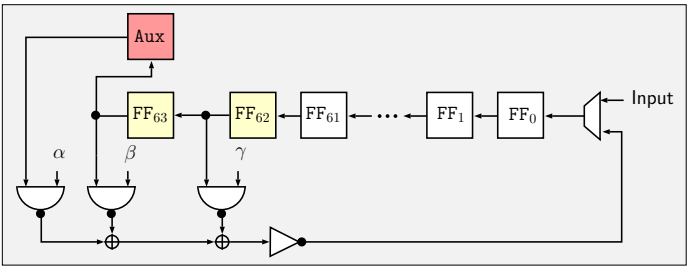
- And 3x is given as

$$3 \times l_{63}l_{62}\cdots l_0 = (l_{62} \oplus l_{63})(l_{61} \oplus l_{62})\cdots(l_4 \oplus l_5)(l_3 \oplus l_{63} \oplus l_4)(l_2 \oplus l_{63} \oplus l_1)$$
$$(l_1 \oplus l_2)(l_0 \oplus l_{63} \oplus l_1)(l_{63} \oplus l_0).$$

- If $V = v_{63}v_{62}\cdots v_0 = 2 \times l_{63}l_{62}\cdots l_0$ and $W = w_{63}w_{62}\cdots w_0 = 3 \times l_{63}l_{62}\cdots l_0$, where v_i, w_i are given as

$$v_i = \begin{cases} l_{i-1} \oplus l_{63} & \text{if } i \in \{1,3,4\}, \\ l_{63} & \text{if } i = 0, \\ l_{i-1} & \text{otherwise} \end{cases} \qquad w_i = \begin{cases} l_{i-1} \oplus l_{63} \oplus l_i & \text{if } i \in \{1,3,4\}, \\ l_{63} \oplus l_i & \text{if } i = 0, \\ l_{i-1} \oplus l_i & \text{otherwise.} \end{cases}$$

Finite field multiplication



2x and 3x over 64 cycles

- we have

$$v_i = \begin{cases} l_{i-1} \oplus l_{63} & \text{if } i \in \{1,3,4\}, \\ l_{63} & \text{if } i = 0, \\ l_{i-1} & \text{otherwise} \end{cases} \qquad w_i = \begin{cases} l_{i-1} \oplus l_{63} \oplus l_i & \text{if } i \in \{1,3,4\}, \\ l_{63} \oplus l_i & \text{if } i = 0, \\ l_{i-1} \oplus l_i & \text{otherwise.} \end{cases}$$

- In cycle 0, bit l_{63} is first stored in an auxiliary register `Aux`.
- Write $u = (\alpha \cdot \texttt{Aux}) \oplus (\beta \cdot \texttt{FF}_{63}) \oplus (\gamma \cdot \texttt{FF}_{62})$

Timeline

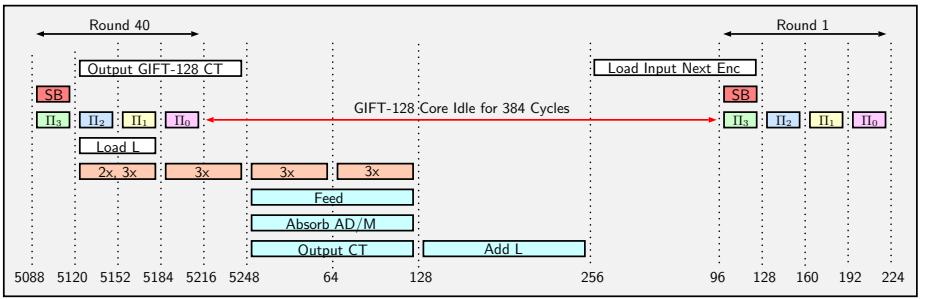


Figure: Timeline diagram and cycle-by-cycle description of GIFT-COFB-SER-S for two successive encryptions. Note the interval of 3×128 idle cycles between encryptions.

Circuit Diagram

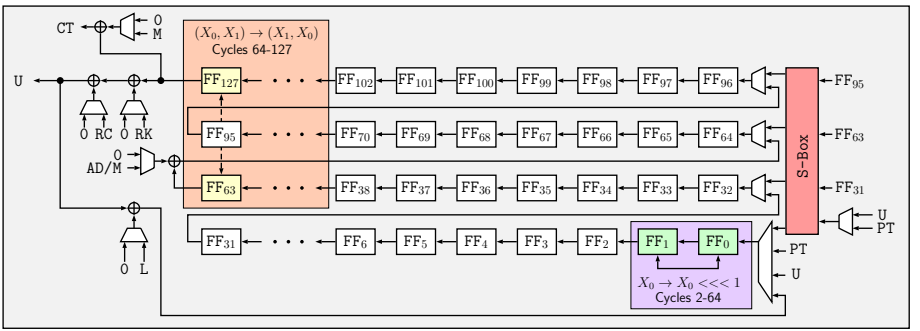
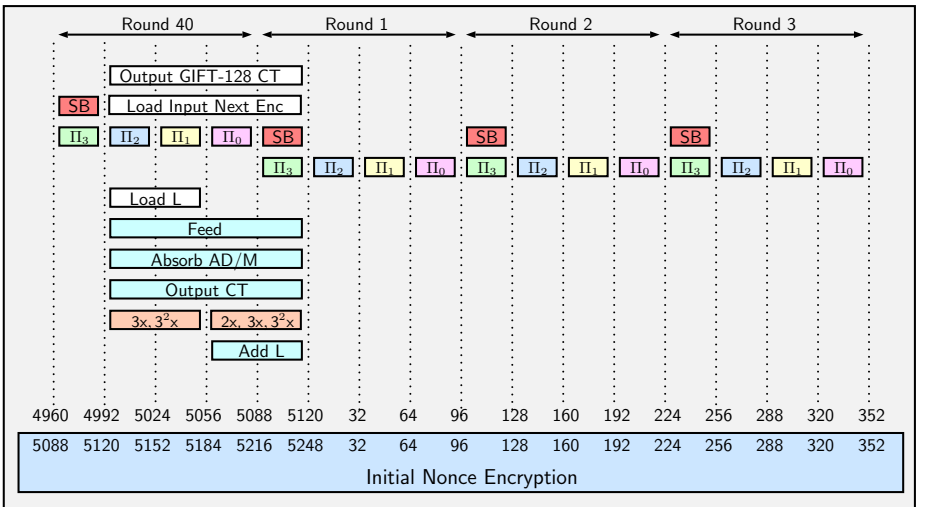


Figure: GIFT-COFB-SER-S state pipeline.

Faster Circuit



Faster Circuit

Feed is Linear

- $\text{Feed}^{-1}(L||0^{64}) = 0^{64}||L$ and
- $\text{Feed}(X \oplus \text{Feed}^{-1}(D) \oplus (0^{64}||L))) = \text{Feed}(X) \oplus D \oplus L||0^{64}$.

- Reorder the sequence of databits to be added.
- First 64 cycles no contribution from L to addition $\rightarrow$ Use it to update L.

Perform all $2^x 3^y$ in 64 cycles

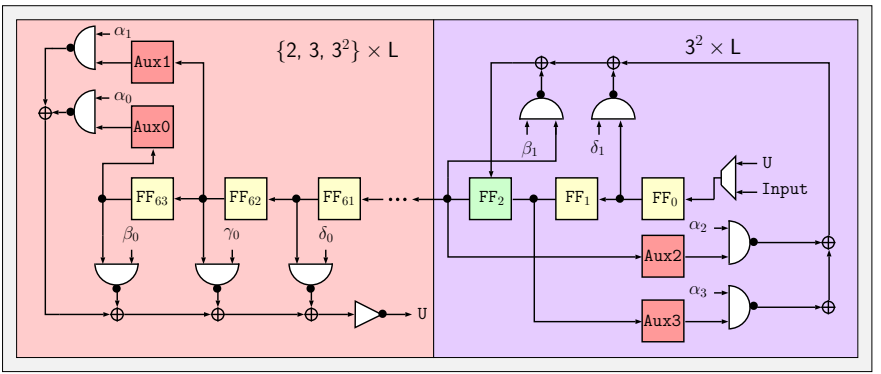


Figure: L state pipeline that performs the multiplication by the factors 2, 3, 3^2, 3^3 and 3^4.

First Order Threshold Implementation

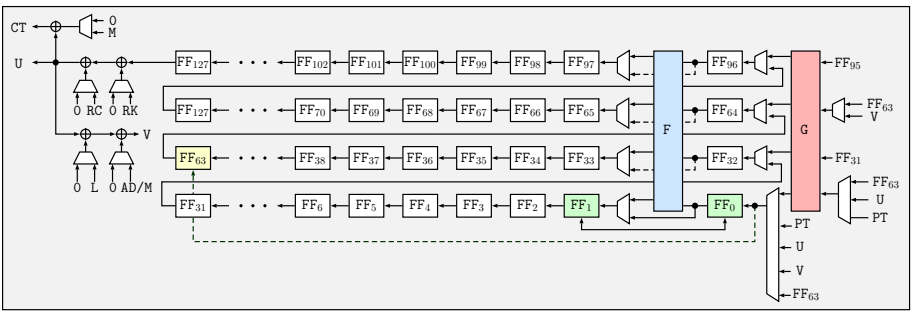
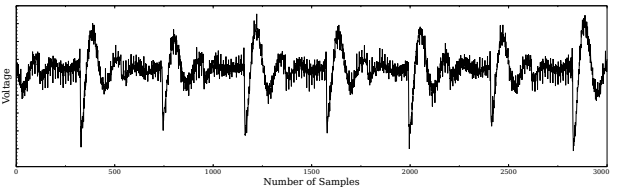
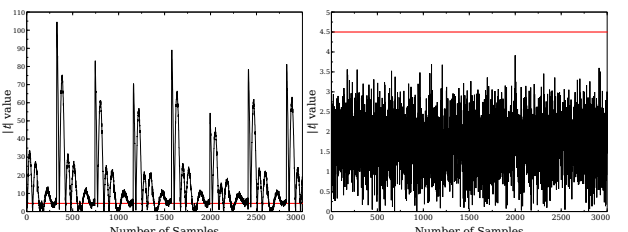


Figure: One of the three state pipeline shares of the GIFT-COFB-SER-TI circuit.

Results of T-Tests



(a) A sample trace taken over 7 cycles.



(b) 20K traces and masks off.(c) 10M traces and masks on.

Figure: Sample trace (top) and t-test results. The red lines $\Rightarrow |t| = 4.5$.

Results

Table: Synthesis results overview for lightweight block cipher based NIST LWC competitors using the STM 90 nm cell library at a clock frequency of 10 MHz. Latency and energy correspond to the encryption of 128 bits of AD and 1024 message bits. Highlighted schemes are NIST LWC finalists.

	Datapath	Area	Latency	Power	Energy	Reference
	Bits	GE	Cycles	μW	nJ	
SUNDAE-GIFT	1	1201	92544	55.48	513.4	BCB Tches 2021
SAEAES	1	1350	24448	84.47	206.5	BCB Tches 2021
Romulus	1	1778	55431	82.28	456.1	BCB Tches 2021
SKINNY-AEAD	1	3589	72960	143.7	1048	BCB Tches 2021
GIFT-COFB	128	3927	400	156.3	6.254	COFB document
GIFT-COFB-SER-S	1	1443	54784	50.11	275.8	This work
GIFT-COFB-SER-F	1	1485	51328	62.15	319.8	This work
GIFT-COFB-SER-TI	1	3384	51328	158.1	813.5	This work

THANK YOU