electronics

Article

An Area-Optimized and Power-Efficient CBC-PRESENT and HMAC-PHOTON

Chi Trung Ngo [1], Jason K. Eshraghian [2] and Jong-Phil Hong [1,*]

[1] School of Electrical Engineering, Chungbuk National University, Cheongju 28644, Korea; trung@cbnu.ac.kr
[2] Department of Electrical and Computer Engineering, University of California Santa Cruz, Santa Cruz, CA 95064, USA; jeshragh@ucsc.edu
* Correspondence: jphong@cbnu.ac.kr

Abstract: This paper introduces an area-optimized and power-efficient implementation of the Cipher Block Chaining (CBC) mode for an ultra-lightweight block cipher, PRESENT, and the Keyed-Hash Message Authentication Code (HMAC)-expanded PHOTON by using a feedback path for a single block in the scheme. The proposed scheme is designed, taped out, and integrated as a System-on-a-Chip (SoC) in a 65-nm CMOS process. An experimental analysis and comparison between a conventional implementation of CBC-PRESENT/HMAC-PHOTON with the proposed feedback basis is performed. The proposed CBC-PRESENT/HMAC-PHOTON has 128-bit plaintext/text and a 128-bit secret key, which have a gate count of 5683/20,698 and low power consumption of 1.03/2.62 mW with a throughput of 182.9/14.9 Mbps at the maximum clock frequency of 100 MHz, respectively. The overall improvement in area and power dissipation is 13/50.34% and 14.87/75.28% when compared to a conventional design.

Keywords: lightweight cryptography; HMAC-PHOTON; CBC-PRESENT; IoT; authentication

Citation: Ngo, C.T.; Eshraghian, J.K.; Hong, J.-P. An Area-Optimized and Power-Efficient CBC-PRESENT and HMAC-PHOTON. *Electronics* **2022**, *11*, 2380. https://doi.org/10.3390/electronics11152380

Academic Editor: Cheng-Chi Lee

Received: 22 June 2022
Accepted: 25 July 2022
Published: 29 July 2022

1. Introduction

The Internet of Things (IoT) is an emerging paradigm in which billions of physical devices can be connected together in a heterogeneous network [1]. The exchange of data between these devices, such as radio-frequency identification (RFID) tags and IoT sensors and smart cards, raises concerns about data security and privacy [2–4]. This demands the exploration of cryptographic methods that enable secure communication to ultimately ensure the protection of confidential data, integrity, and authentication for IoT networks. The National Institute of Standards and Technology (NIST) specified five block cipher modes to provide data confidentiality: Electronic Code Book (ECB), Cipher Block Chaining (CBC), Cipher Feedback (CFB), Output Feedback (OFB), and Counter (CTR) [5]. Other than ECB, which is the smallest cipher mode, the modes require an Initial Vector (IV) to make ciphered messages unique. The modes can be used in conjunction with an approved symmetric key algorithm by using a Federal Information Processing Standard (FIPS), such as the Advanced Encryption Standard (AES) and Triple DES (TDES). However, NIST also specifies the use of Keyed-Hash Message Authentication Code (HMAC) to pursue message authentication and integrity [6]. The HMAC algorithm requires an approved hash function, such as the SHA-2 family [7], to generate Message Authentication Codes (MACs).

However, most conventional cryptography methods and algorithms are implemented in security systems intended for desktop/server environments, which demand a large implementation area, a high-power consumption, and a large memory capacity [8]. Fox example, the proposed TDES structure in [9] requires approximately 10,641 equivalent transistor gates, which exceeds the gate count of 1000–10,000 for low-cost RFID tags recommended by NIST [10]. In addition, standardized hash functions, such as SHA-2, are too large to fit within resource-constrained devices, occupying a 10,868 gate equivalent (GE) [11]. As a result, these algorithms are unsuitable for resource-constrained IoT devices, such as embedded systems, RFID devices, and low-power sensor networks. Thus, several lightweight cryptography (LWC) methods have been introduced as security solutions for sensitive data in constrained IoT devices [12]. When compared with conventional cryptography methods, LWC can reduce a silicon area occupation and power consumption by using fewer computing resources and lower power supply voltages that are better tailored for edge devices [13]. Among various LWC algorithms, PRESENT is one of the first standardized lightweight symmetric key algorithms designed for security services for RFID air interface communication [14]. Moreover, PHOTON is among the first approved lightweight hash functions designed for constrained devices. CBC-PRESENT and HMAC-PHOTON are potential LWC candidates for IoT sensor devices due to their compactness. In the former, CBC is an operation mode of the lightweight algorithm PRESENT. In the latter, PHOTON is a LWC hash function and HMAC is a message authentication protocol using cryptographic hash functions. However, in the conventional operational flows of both methods, the designs utilize multiple hash functions and block ciphers independently. This in turn increases their implementation areas and power consumption due to the large number of hash functions and block ciphers utilized.

In this paper, an optimized operational flow of CBC-PRESENT and HMAC-PHOTON is proposed by applying feedback such that the implementation of the proposed structure requires only a single hash function and a block cipher within a unified scheme. This paper extends our work in [15,16], where the hardware architectures of both algorithms are briefly introduced. The proposed structure is shown to reduce the number of either block ciphers or hash functions, hence reducing the implementation overhead in terms of area and power consumption. Thus, it can be combined with security primitives, such as physically unclonable functions (PUFs) [17–20], to provide the security services for edge devices while still meeting their requirements. This approach can be applied to HMAC based on various hash functions (SHA, SPONGENT, etc.) and the CBC modes of various block ciphers (AES, CLEFIA, etc.) regardless of size of plaintext and key. In this paper, this optimization is performed for PRESENT-CBC and HMAC-PHOTON with 128-bit plaintext and a 128-bit key. Both algorithms are implemented in a 65-nm CMOS process with the experimental results of our chip provided in full. The results of the CBC-PRESENT encryption, decryption, and HMAC-PHOTON are verified by using a MATLAB implementation and test vectors in their original papers [21,22]. The rest of the paper is organized as follows. In Sections 3 and 4, analyses of the original algorithms and the conventional operation of CBC-PRESENT and HMAC-PHOTON are presented to deliver the proposed optimization architecture for each structure. Section 5 analyzes the security strengths of CBC-PRESENT and HMAC-PHOTON. Finally, Section 6 summarizes the findings of this paper.

2. Literature Review

IoT devices are considered resource-constrained devices [4]. Resources refer to the hardware perspective, including area and power consumption. Conventional cryptography algorithms, each of which requires a high power and large area, are unsuitable for IoT devices. Consequently, lightweight cryptography algorithms are necessary to balance security requirements with hardware requirements. In the early work on lightweight cryptography in the 80s and 90s, numerous compact cipher implementations were proposed, namely Noekeon [23], Des [24], Camellia [25], etc. The next decades saw extensive research on optimizing various constraints with a special effort on area occupation. The international organization of standards published two documents to standardize PRESENT [21], CLEFIA [26], Led [27], etc. as a lightweight block cipher [22] and to standardize PHOTON [22], SPONGENT [28], Lesamnta-LW [29], etc. as lightweight hash functions [30]. Since that point forward, numerous algorithms have been proposed to focus on optimizing speed and enhancing security, but no official ISO standards have been released. Among the standardized LWC, PRESENT and PHOTON hardware requirements are competitive with today's leading compact block ciphers and hash functions [21,22].

Moreover, cryptography methods are moving toward parallel structures to increase system speeds and integrate multiple security services into single systems. In 1981, the NIST specified four modes of operation to satisfy data confidentiality: ECB, CBC, OFB, and CFB [31]. The institute recommended CTR as a fifth mode of operation in 2001. CTR mode adopts a parallel structure for encryption and decryption methodology, but it costs additional effort to keep a requirement on a counter function. In 2010, the NIST addressed the latest mode of operation named XTS-AES [32]. This mode utilizes heavyweight AES to guarantee confidentiality for storage devices. This mode requires double encryption with two independent keys for each block cipher. Thus, the complexity of the XTS structure is unsuitable for low-resource devices. To guarantee data integrity, the NIST approved HMAC [6] in 2002. Later, hybrid-mode CCM and GCM, which combined confidentiality and data integrity, were introduced [33,34]. In fact, combining security services increases the complexity of the mode of operation, thus increasing the area occupation and power consumption. Deploying the parallelizable structure, the implemented area increases linearly with the input block. In addition, among the recommended mode of operations in [5], CBC shows a competitive implementation result [35].

Using lightweight cryptography in a conventional operation mode with parallel structures increases resource consumption. To make the area occupation suitable for IoT devices, we applied a feedback path and resource reuse techniques with lightweight algorithms and compactness cryptography methods.

3. Proposed CBC-PRESENT Algorithm and Hardware Architecture

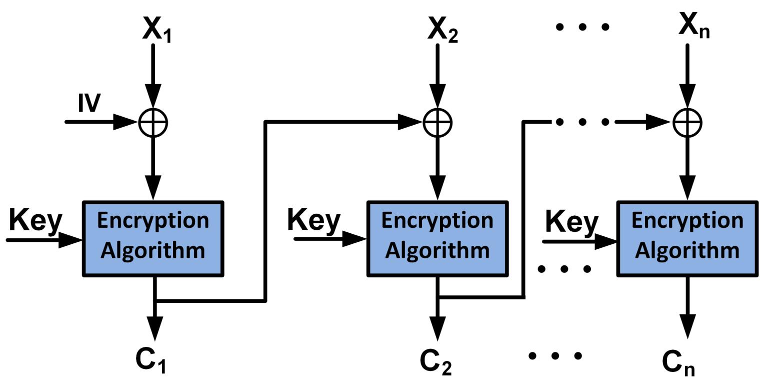
3.1. Proposed Area Optimization of CBC Architecture

The block cipher mode of operation is specified in (1) and (2). Figure 1 illustrates the conventional CBC mode structure as recommended by NIST [5]. In this figure, the symbol $\oplus$ represents the XOR operation. The size of the conventional structure increases linearly with respect to the length of the input plaintext ($X_{1:n}$)/ciphertext ($C_{1:n}$).In the encryption process, the raw plaintext is split to the sub-block plaintexts X_i, as illustrated in Figure 1a. Before an encryption by a cryptography algorithm, the plaintext sub-block X_i is XORed with a previous ciphertext. An initial plaintext sub-block is XORed with an Initialization Vector (IV). In contrast, decryption reverses the encryption process, as illustrated in Figure 1b. A conventional implementation of the block cipher adopts a parallel datapath architecture [5]. Therefore, it costs M encryption/decryption blocks to generate the output/input ciphertext/plaintext.

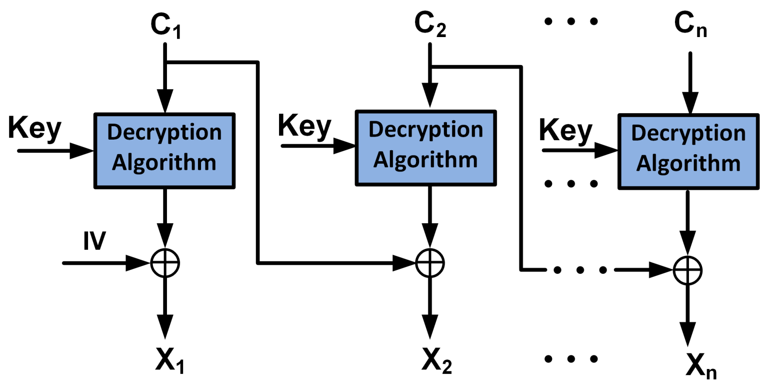
$$C_i = Enc_K(X_i \oplus C_{i-1}), \text{ with } C_0 = IV. \tag{1}$$

$$X_i = Dec_K(C_i \oplus C_{i-1}), \text{ with } C_0 = IV. \tag{2}$$

To overcome the increasing area of the conventional structure, we propose a serial datapath architecture to reduce the number of cryptography algorithm blocks in the system by incorporating feedback for the CBC mode. The encryption dataflow is shown in Figure 2a. For the decryption process, inputs are replaced with C_i and outputs switch between operating modes (encryption and decryption). The proposed structure uses the CBC sequential block to switch between the operating modes and perform the XORed bits between the ciphertext and plaintext as represented by (1) and (2). By taking advantage of the feedback technique and the serial datapath, only one cryptography block is required to encrypt $M \times X_i$ blocks or to decrypt $M \times C_i$ blocks. If M is significantly large, the area consumption of the CBC sequential block can be considered negligible. Then, our structure reduces the area occupation by a factor of M when compared to the conventional structure. Additionally, the CBC sequential block requires three states to control the operating modes of the cryptography algorithm, as specified in Figure 2b. A “RESET” state resets the algorithm after completing one encryption/decryption cycle in a “WORK” state. A controller loops between the two states until it encrypts or decrypts an entire input bit sequence and generates an output at the “FINISH” state.



(**a**)



(**b**)

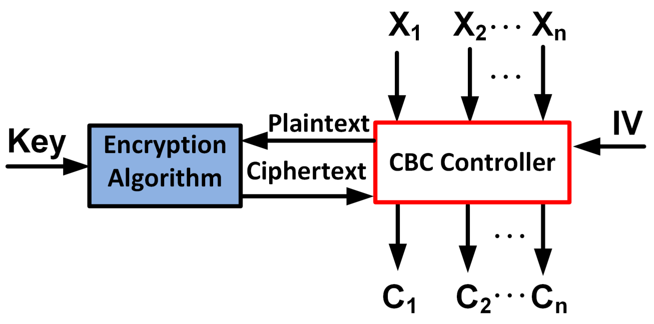
Figure 1. (**a**) CBC mode encryption, (**b**) CBC mode decryption.

The latencies for the encryption and decryption process are denoted as t_{en} and t_{de}, respectively. By incorporating feedback, the latency of the proposed structure is $M \times (t_{\text{en}} + 1)$ for encryption and $M \times (t_{\text{de}} + 1)$ for decryption. The additive "+1" term appears because the controller transitions between a "WORK" state and "RESET" state. In the encryption architecture, ciphertext C_i depends upon the concurrent plaintext sub-block X_i and the previous ciphertext C_{i-1}. Thus, the conventional structure costs $M \times t_{\text{en}}$ to generate a final ciphertext from the original plaintext. For encrypting $M \times X_i$ blocks, one defines (3), which shows the relationship between the latency and the number of inputs. Δ represents the difference in latency between the proposed structure and the conventional architecture as a percentage:

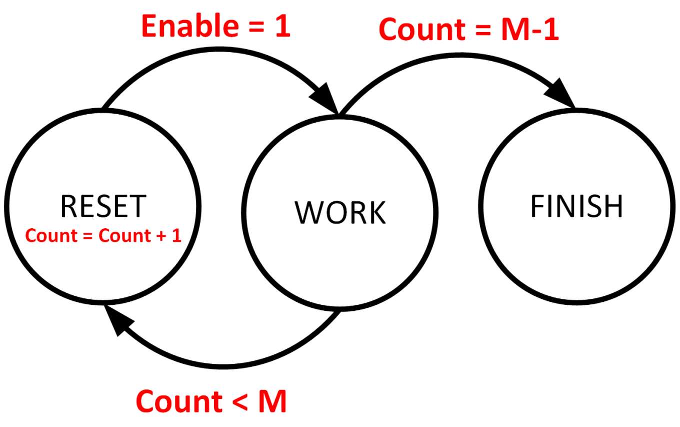
$$\Delta = \frac{M \times (t_{\text{en}} + 1) - M \times t_{\text{en}}}{M \times t_{\text{en}}} \times 100\% = \frac{1}{t_{\text{en}}} \times 100\%, \tag{3}$$

$$\Delta = \frac{M \times (t_{\text{de}} + 1) - t_{\text{de}}}{t_{\text{de}}} \times 100\% = (M - 1 + \frac{M}{t_{\text{de}}}) \times 100\%. \tag{4}$$

Equation (3) shows that Δ depends only on t_{en} and not on the number of inputs. If the encryption latency for the cryptographic algorithm is in the order of 50–100 clock cycles, Δ yields 1–2%, which can be considered a negligible increase. In fact, the existing implementations of the cryptographic algorithms requires more than 50 clock cycles to generate an output, such as PRESENT (563 cycles in [10]) and AES (160 cycles in [36]). A slight increase in latency can be considered acceptable in light of the area reduction by a factor of M. However, the decryption process does not depend on the previous ciphertext; therefore, deciphering all the ciphertext takes only t_{de}. The latency increases linearly with M as specified in (4), which highlights a more critical trade-off between the area occupation and latency during decryption, as it can no longer be parallelized when using our feedback-based approach. In addition, the feedback structure can be applied to other cipher blocks, such as ECB, CFB, OFB, and CTR, to reduce the area occupation at the cost of the latency.



(**a**)



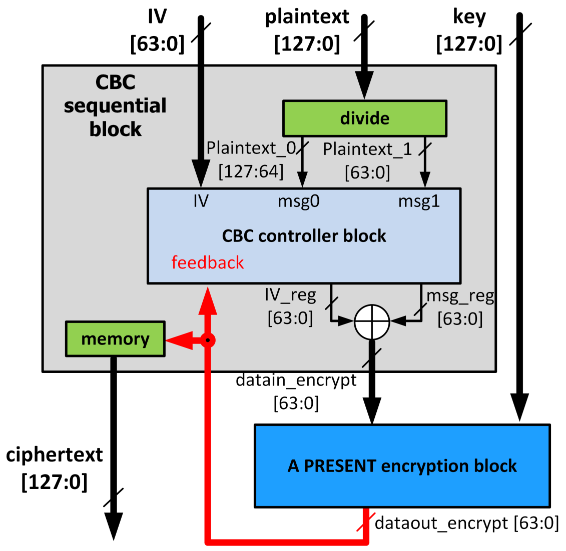
(**b**)

Figure 2. (**a**) Proposed hardware architecture of CBC encryption and (**b**) Finite State Machine (FSM) of CBC sequential block.

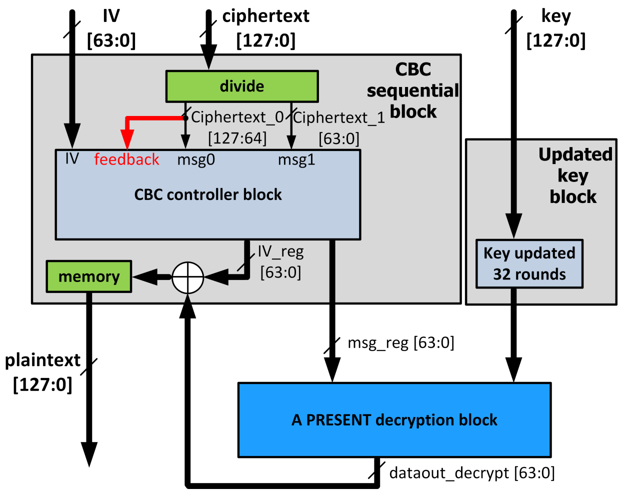
3.2. Proposed Hardware Implementation of CBC-PRESENT

We implement the lightweight algorithm PRESENT-128 for a CBC expansion encryption in our SoC. PRESENT-128 generates a 64-bit ciphertext from a 64-bit plaintext and a 128-bit key. CBC-PRESENT uses 128-bit plaintext to generate 128-bit ciphertext. Therefore, the value of the "count" (Figure 2b) is two in this case.

Figure 3a demonstrates the proposed hardware architecture of PRESENT-128 with 128-bit plaintext and CBC expansion encryption. In the CBC sequential block, the plaintext was separated into two 64-bit plaintext sub-blocks (Plaintext_0 for a first encryption and Plaintext_1 for a second encryption) to match with the input requirement of the PRESENT block cipher. IV_reg stores the IV value for a first encryption, then a dataout_encrypt of the first encryption step is stored in IV_reg for a second encryption. The selection is performed by the CBC controller that is illustrated in Figure 4. In the WORK state, the input plaintext of the PRESENT encryption block is the result of an XORed between IV_reg and msg_reg. In contrast, the decryption process of CBC-PRESENT is shown in Figure 3b. The keys for the encryption process are calculated in an update key block. Similar to an encryption operation, the ciphertext is also divided into two 64-bit ciphertext sub-blocks (Ciphertext_0 for the first decryption and Ciphertext_1 for the second decryption). After the first decryption, dataout_decrypt is XORed with IV to generate plaintext_0. Plaintext_1 is obtained by XORing the dataout_decrypt of the second decryption with ciphertext_1.



(**a**)



(**b**)

Figure 3. (**a**) CBC-PRESENT encryption and (**b**) CBC-PRESENT decryption.

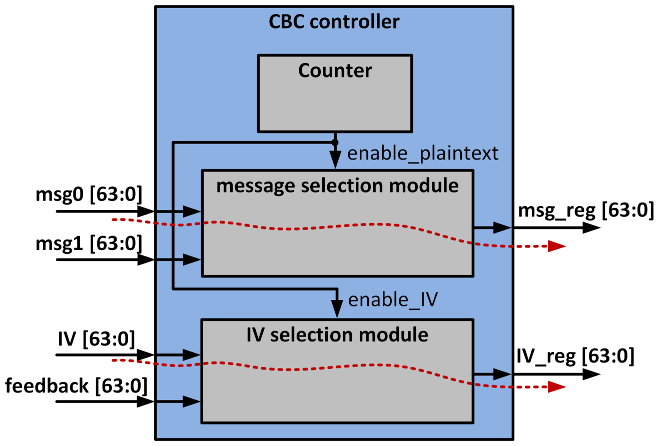
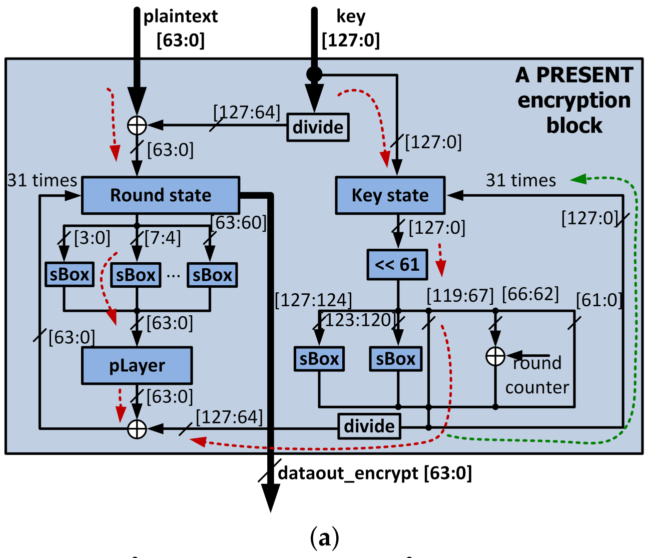
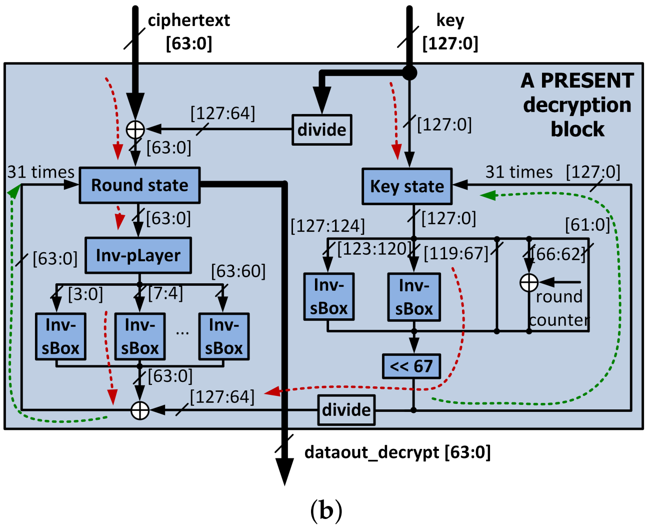


Figure 4. Hardware architecture of the CBC controller block.

Figure 5 shows the architecture of the PRESENT encryption and decryption block cipher. In our work, we implement PRESENT by following the specifications for it in its original paper [21]. The order of the data flow is illustrated in Figure 5b where the red arrows depict the sequential operation of the key schedule and the round schedule that includes sBox/Inv-sBox, pLayer/Inv-pLayer, and a shift register. The green arrows illustrate the feedback in a PRESENT block schedule. A PRESENT block requires 31 rounds of operation to generate a final output. A round state uses a counter to count the number of rounds and registers to store data from feedback. For each encryption round, the stored data from the round state must go through an sBox block and a pLayer block, which is then XORed with the key obtained from the key schedule. The sBox block in PRESENT is a 4-bit to 4-bit substitution. PLayer is a bit permutation following the table from [21]. In contrast, the decryption uses the Inv-sBox, Inv-pLayer which is the inversion mapping of sBox and pLayer, respectively, to obtain the plaintext. The key state includes a counter to count the number of rounds and registers for a storing key from feedback. The notations "$<< 61$" and "$<< 67$" are a 61-bit left shift registers for encryption and a 67-bit left shift registers for decryption, respectively.



(a)



(b)

Figure 5. Hardware architecture of PRESENT (**a**) encryption and (**b**) decryption.

4. Proposed HMAC-PHOTON Algorithm and Hardware Architecture

The NIST standard defines HMAC as a MAC that uses a cryptography hash function in conjunction with a secret key [6]. HMAC requires two distinct parameters, a message input text, and secret key K. Figure 6 shows the HMAC operation to generate a MAC value. In this figure, K_0 is the key K after undergoing pre-processing to obtain a sufficient bit length output. H is the approved hash function, "ipad" refers to an inner pad that is a repetition of the byte $\times$ 36 to achieve the same length as K_0, opad is an outer pad that repeats the byte × 5c to achieve the same length as K_0, and the symbol || is an operator that concatenates its arguments. In Figure 6, the $HMAC_K$ (text)—the MAC value generated from the HMAC operation—is represented as (5).

$$HMAC_K(text) = H[(K_0 \oplus opad)||(H[K_0 \oplus ipad||text])]. \tag{5}$$

At the initial step, determine K_0; if the length of an initial secret key K is smaller than the length of input B of the hash function, zeros are appended to the end of K to create the B-byte string K_0. If the length of K is bigger than the length of B, K_0 is obtained by hashing K, then appending zeros to create the B-byte string K_0. Otherwise, K_0 equals K. Then, K_0 is XORed with the ipad and concatenated with the text to initialize the inner hash function. The outer hash is computed by concatenating the output of the inner hash and the XORed result from K_0 and the opad. Values of the opad and ipad were chosen for ease of implementation and to provide a high Hamming distance for the pads [37]. The hash function is initialized multiple times to guarantee the security of HMAC, which is explained in its original presentation by Ref. [37].

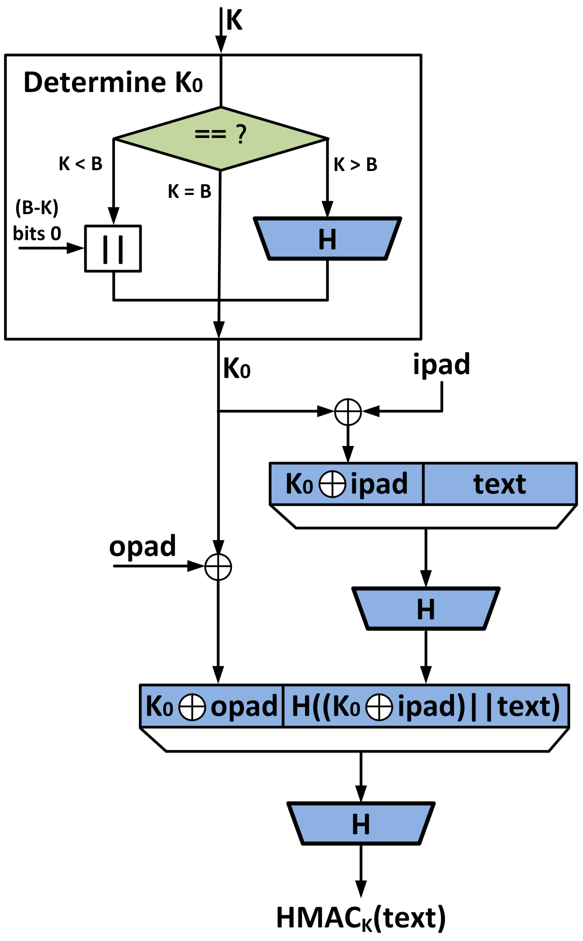


Figure 6. HMAC operation.

4.1. Proposed Area Optimization of HMAC Hardware Architecture

When the secret key K > B, the HMAC operation requires hashing three times to generate a MAC value. In this case, the conventional HMAC costs three hash function blocks, as illustrated in Figure 7. Thus, the conventional hardware architecture of HMAC suffers from a high area occupation because of a duplication of the hash function. Before executing hash_function 2 and hash_function 3, the system must wait for hash_function 1 and hash_function 2, respectively, to complete the processes. Therefore, during operation, only one hash function block is active and the other two remaining blocks are redundant.

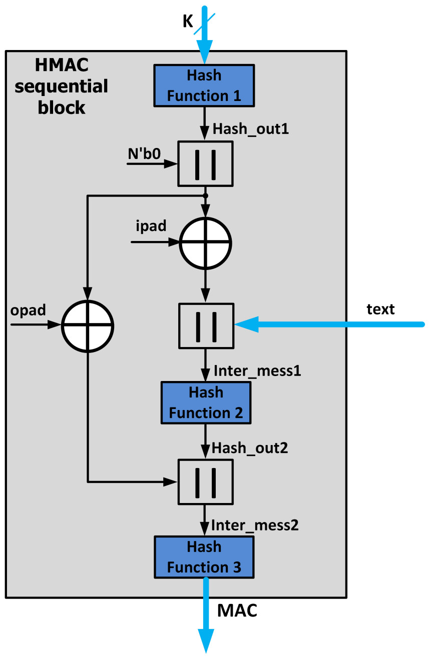


Figure 7. The conventional hardware architecture of HMAC when K > B.

Figure 8 presents the proposed hardware architecture of HMAC. As seen in this figure, three hash functions are replaced with an HMAC controller that contains only one hash function block. By reusing the hash function block, the proposed HMAC can reduce the area occupation and power consumption. In Figure 8, hash_out1 is generated from K then is XORed with the ipad and is concatenated with a message to obtain inter_mess1. hash_out2 is obtained by hashing inter_mess1. The input for the final hashing is generated by concatenating hash_out2 and the XORed result between hash_out1 and opad.

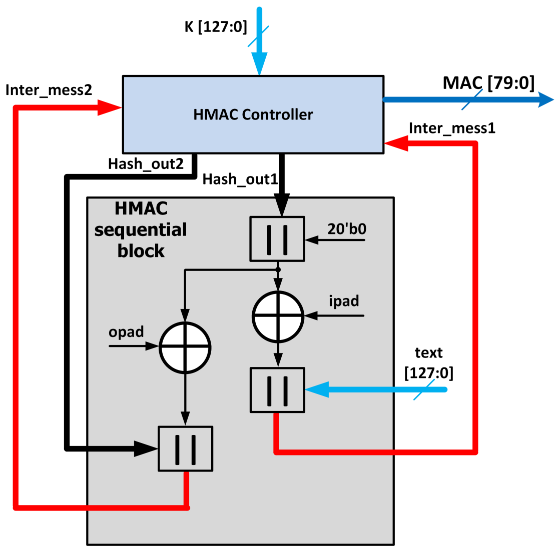


Figure 8. The proposed hardware architecture of HMAC.

The optimization of HMAC is analogous to the previous section. We deploy only one hash function block combined with the HMAC controller to reduce the area consumption of the HMAC algorithm. The operation principle of the HMAC controller block is similar to the CBC controller block. In our implementation, the number for the total hashing time depends on the initial secret key K setup. If the input bit K is smaller or equal to the input of the hash function (100 bit for PHOTON80), a MAC value is generated by performing hashing two times. Otherwise, it takes three repetitions of hashing to obtain a

MAC value. The change in the latency follows Equation (3), where t_{en} is replaced with the time to complete a hash function. Commonly used hash functions, such as SHA [7], take approximately 50–100 clock cycles to generate a MAC value. Thus, the latency increase is only 1–2%, which can be treated as negligible.

4.2. Hardware Implementation of HMAC-PHOTON

For compatibility with IoT devices, we present a lightweight ASIC implementation of HMAC based on the PHOTON family of hash functions. The bit size of the secret key (K = 128) here is larger than the block size (B = 100) of the PHOTON hash function, so K_0 is generated by hashing K then appending zeros. The hash out of HMAC-PHOTON is 80-bit (L = 80); therefore, $K_0 = H(K_0)$, 20′b0. The message is selected as 128 bit in length in this case. After the initial setup, the sequence of steps from Figure 6 is followed. Figure 9 illustrates the proposed hardware architecture of the HMAC controller block. This block is designed by using a counter associated with three selection modules. In our implementation, the first output of the hash function Hash_out1 is an HMAC key that is selected by following the "Determine K_0" step. After that, the hash function runs another two rounds with the inputs inter_mess1 and inter_mess2 to generate a final MAC value.

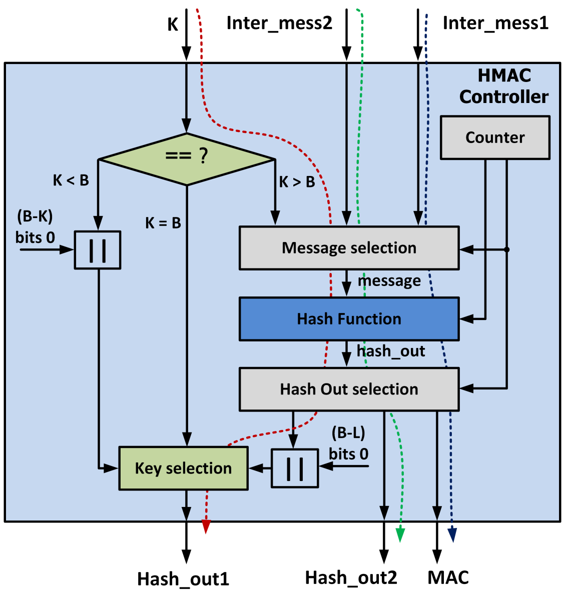


Figure 9. Proposed hardware architecture of HMAC controller.

The hardware architecture of PHOTON, which follows [22], is illustrated in Figure 10. The red arrows demonstrate the absorbing process, the blue arrows represent the squeezing process, and the pink arrow displays the final hash output generation. The PHOTON selection and message-padding block select the PHOTON message corresponding with each step and implement the padding process. The output of this block (message_padded) is imported into the absorbing process, and the message_absorbing signal enters the hash function permutation. The size of the message input decides the running time of the absorbing process. After the absorbing process occurs, a squeezing process is activated. The absorbing and squeezing process controller includes counter and memory. In our design, the squeezing process runs five times, and the output of hash_permutation in each instance is stored in the hash out merging block to generate the final hash_out. The structure of the hash function permutation is designed based on a permutation controller block and a PHOTON round block, as shown in Figure 10. A PHOTON round includes four operations: addconstant, subcells, shiftrows and mixcolumn. For further information regarding the four operations, the reader is referred to the original proposal of them in [22]. The permutation controller block is constructed from a counter and memory to control the operation of the permutation process. In the permutation process, a PHOTON round is called 12 times, and the last PHOTON round generates the final hash_permutation.

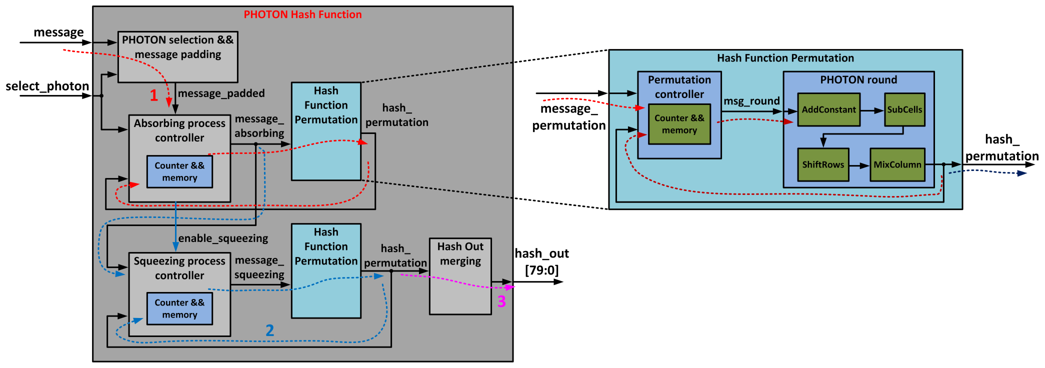


Figure 10. Hardware architecture of PHOTON.

5. Measurement Results

A prototype of the proposed CBC-PRESENT and HMAC-PHOTON was fabricated in a 65 nm CMOS process to verify the performance of the proposed architectures.

Figure 11 shows a die micrograph of the proposed system. For the measurement, a Xilinx FPGA board with the implemented SoC were used as shown in Figure 12. A test-bench was generated using MATLAB and transmitted to the FPGA board via a universal asynchronous receiver-transmitter (UART) cable. The FPGA stored the test-bench in SRAM, then used a scan-chain structure to import the value of each input. After the input setup phase, the output data was stored in SRAM and transmitted back to MATLAB in a binary format. The captured data from MATLAB was converted to hexadecimal and compared with the algorithm test-vector. In addition, a Logic Analyzer (LA) was used to capture the data at the I/O pin of the SoC. The data was visualized in the LA's display screen, and its value was expected to be identical with the data captured from the FPGA. By comparing the data between the LA and FPGA, we could guarantee the correctness of the measurement. In addition, a random input vector was imported into the SoC and the MATLAB test bench. The test results from MATLAB were used to verify the correctness of all the test vectors, which were not included in the original paper.

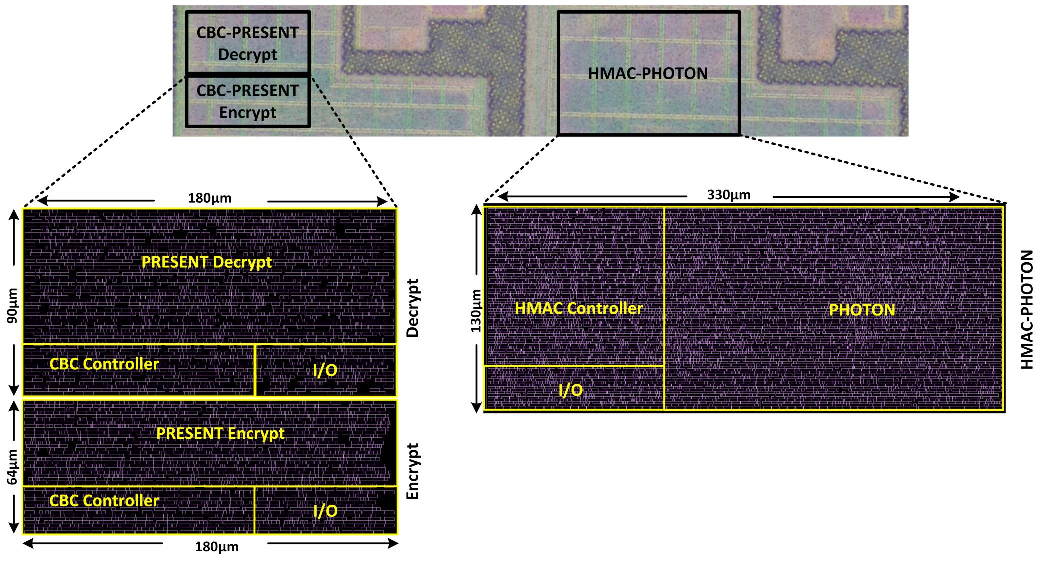


Figure 11. Die micrograph of the CBC-PRESENT & HMAC-PHOTON.

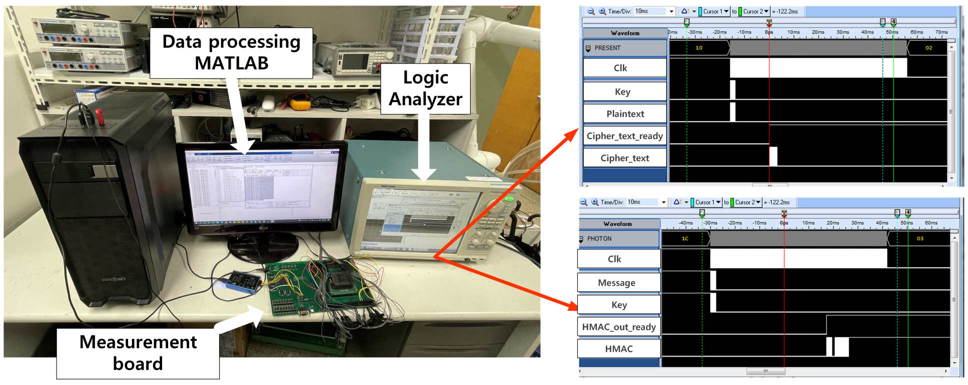


Figure 12. SoC measurement environment. MATLAB generated test vectors that were streamed to the measurement board, where the output results were displayed on the interface of the logic analyzer and were transmitted back to MATLAB in binary format.

The performance results were expressed in terms of area (GE), power consumption, latency, and throughput. The area in GE was obtained by dividing the total area by the area of a two-input NAND gate. Latency was defined as the measure of time between when the inputs were imported into the chip and the time at which the chip generated the outputs. Throughput was the rate at which outputs were produced by following (6). All designs were implemented at a 100 MHz clock frequency.

$$\text{Throughput} = \frac{\text{Number of bits of data input}}{\text{latency}} \times \text{frequency}. \quad (6)$$

For the performance comparison, each algorithm was designed using Verilog and synthesized using Synopsys tools. We compared the controller of each algorithm with the conventional structure to highlight the efficiency of replacing PRESENT128 encryption/decryption with the CBC controller block and replacing PHOTON80 with the HMAC controller. The synthesized results of the proposed CBC-PRESENT showed that the area consumption of a CBC controller costs only 1226 GE, while a PRESENT encryption and a PRESENT decryption require 3132 GE and 5999 GE, respectively. An area occupation of the PRESENT encryption and decryption is around 2.5 times and 5 times larger than the area consumption of the CBC controller. However, an HMAC controller costs 4038 GE. Meanwhile, a PHOTON requires 13,321 GE, which is three times greater than the area occupation of the HMAC controller. Therefore, using the proposed controller block to replace the cryptography algorithm saves a significant amount of area when compared to the conventional architecture.

Table 1 shows a performance and security comparison of our proposed design, a conventional CBC-PRESENT, and AES. Table 2 displays a performance and security comparison of the proposed HMAC-PHOTON 80 architecture with the conventional architecture and HMAC-SHA-256. We set an identical message and key size (128-bit plaintext and 128-bit key) for a fair evaluation between all the implementations. We also included AES synthesis results and an HMAC-SHA256 synthesis in the comparison to highlight the efficiency of our work in terms of area occupation and power consumption. Figures 13 and 14 show the comparison results of the area occupation and power consumption in the proposed, conventional CBC-PRESENT and AES, respectively. The proposed architecture of the CBC-PRESENT obtains less than the conventional designs by 13% in the occupied area and by 14.87% in dissipated power. Compared to our AES design, PRESENT-CBC reduces area occupation by 79.27% and 41.14% in terms of power dissipation, as shown in Figures 13 and 14. In addition, our encryption design has better area implementation than the design in [35]. However, as can be seen in Figure 15, the proposed HMAC-PHOTON decreased by 50.34% in area and 75.28% in power usage compared to HMAC-SHA256. In addition, the results of the proposed HMAC-PHOTON reduced the area and power dissipation of the conventional HMAC-PHOTON by 41.57% and 26.61% with regards to the area and power dissipation of conventional designs, respectively. The most advanced contribution of the proposed architectures is the significant reduction of power and area consumption by using feedback to optimize the serial procedure.

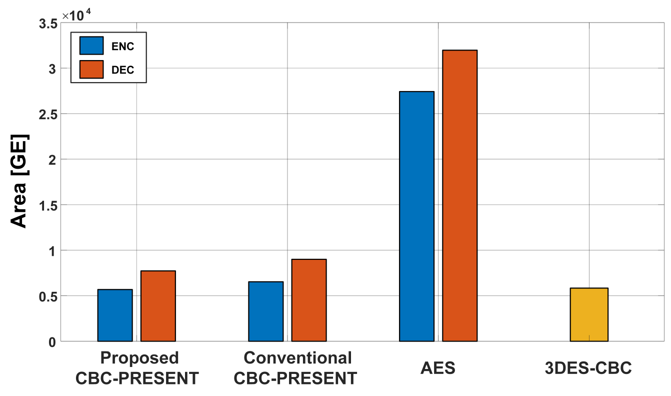


Figure 13. Die area occupation of the proposed CBC-PRESENT when compared to a conventional design and AES.

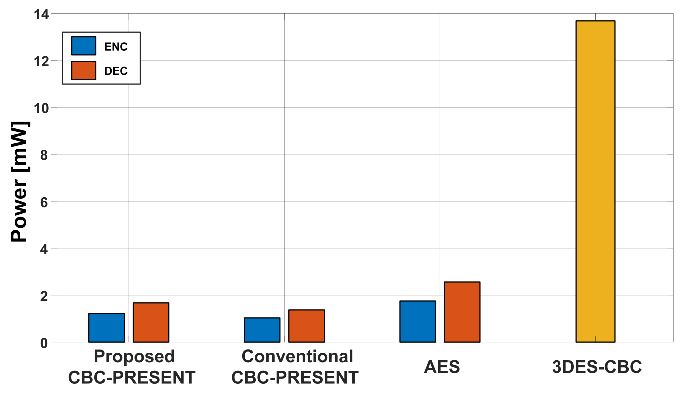


Figure 14. Power consumption of the proposed CBC-PRESENT when compared to a conventional design and AES.

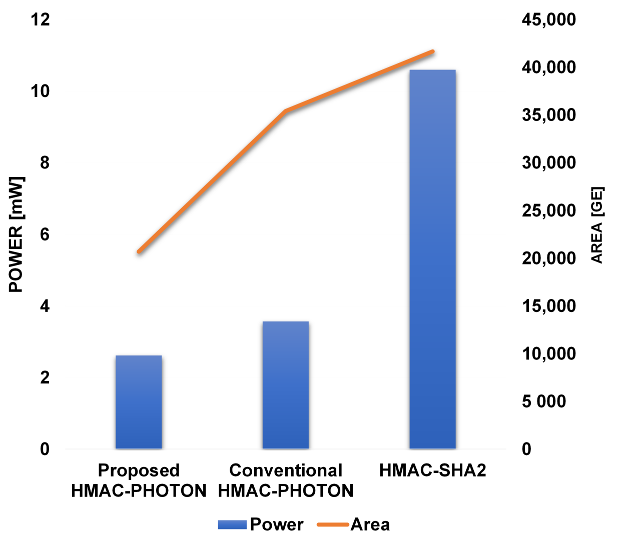


Figure 15. Area and power consumption of cryptography algorithm base HMAC.

Table 1. Performance and security comparison of CBC-PRESENT.

	Proposed CBC-PRESENT		**Conventional CBC-PRESENT**		**AES**		**3DES-CBC [38]**
	ENC	DEC	ENC	DEC	ENC	DEC	
CMOS Technology [nm]	65	65	65	65	65	65	65
Plaintext [bit]	128	128	128	128	128	128	64
Key size [bit]	128	128	128	128	128	128	144
Block size [bit]	64	64	64	64	128	64	
Cycles	70	105	68	70	15	22	
Frequency [MHz]	100	100	100	100	100	100	2100
Throughput [Mbps]	188.2	182.9	182.9	121.9	853.3	581.8	2840
Area [GE]	5.68K	7.72K	6.53K	9K	27K	36K	5.84K
Power [mW]	1.03	1.37	1.21	1.67	1.75	2.56	13.68

Table 2. Performance and security comparison of HMAC-PHOTON.

	Proposed HMAC-PHOTON	**Conventional HMAC-PHOTON**	**HMAC-SHA256**
CMOS Technology [nm]	65	65	65
Message [bit]	128	128	128
Key size [bit]	128	128	128
Hash size [bit]	80	80	256
Cycles	860	856	320
Frequency [MHz]	100	100	100
Throughput [Mbps]	14.9	14.9	40
Area [GE]	17,359	35,423	41,681
Power [mW]	2.62	3.57	10.6

6. Conclusions

This paper presents an area-optimized and power-efficient method for CBC-PRESENT and HMAC-PHOTON with 128-bit plaintext and a 128-bit key by applying a feedback path in hardware structures to promote resource reuse. The proposed structures are implemented in a 65-nm CMOS process. Compared with the conventional CBC-PRESENT structure, the proposed CBC-PRESENT structure with a feedback path reduces the implementation area by 13% and power consumption by 14.87%. Similarly, compared with the conventional HMAC-PHOTON, the proposed HMAC-PHOTON with a feedback path also reduces the implementation area by 41.57% and the power consumption by 26.61%.

The proposed cryptography structures should be applied to information security SoC, such as constrained IoT devices demanding small areas and a low power consumption. Future research should consider adding additional circuit elements to protect the design from differential power analysis (DPA).

Author Contributions: Conceptualization, J.-P.H.; methodology, J.-P.H.; software, C.T.N.; validation, C.T.N.; formal analysis, C.T.N.; investigation, C.T.N.; resources, C.T.N. and J.-P.H.; writing—original draft preparation, C.T.N.; writing—review, J.K.E.; supervision, J.-P.H.; project administration, J.-P.H.; funding acquisition, J.-P.H. All authors have read and agreed to the published version of the manuscript.

Funding: This research was supported by Mid-Career Researcher Program through the National Research Foundation of Korea (NRF) funded by the MSIT (Ministry of Science and ICT) (NRF-2021R1A2C2005258), and by the MIST, Korea, under the Grand Information Technology Research Center support program (IITP-2022-2020-0-01462) supervised by the IITP (Institute for Information & communications Technology Planning & Evaluation).

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Greengard, S. *The Internet of Things (MIT Press Essential Knowledge Series)*; The MIT Press: Cambridge, MA, USA, 2015.
2. Frustaci, M.; Pace, P.; Aloi, G.; Fortino, G. Evaluating Critical Security Issues of the IoT World: Present and Future Challenges. *IEEE Internet Things J.* **2018**, *5*, 2483–2495. [CrossRef]
3. Zhou, W.; Jia, Y.; Peng, A.; Zhang, Y.; Liu, P. The Effect of IoT New Features on Security and Privacy: New Threats, Existing Solutions, and Challenges Yet to Be Solved. *IEEE Internet Things J.* **2019**, *6*, 1606–1616. [CrossRef]
4. Meneghello, F.; Calore, M.; Zucchetto, D.; Polese, M.; Zanella, A. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices. *IEEE Internet Things J.* **2019**, *6*, 8182–8201. [CrossRef]
5. Dworkin, M. *Sp 800-38A: Recommendation for Block Cipher Modes of Operation: Methods and Techniques*; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2001.
6. National Institute for Science Technology (NIST). *The Keyed-Hash Message Authentication Code (HMAC) (FIPS PUB 198);* National Institute of Standards and Technology: Gaithersburg, MD, USA, 2002.
7. Sklavos, K. Implementation of the SHA-2 Hash Family Standard Using FPGAs. *J. Supercomput.* **2005**, *31*, 227–248. [CrossRef]
8. Thakor, V.A.; Razzaque, M.A.; Khandaker, M.R.A. Lightweight Cryptography Algorithms for Resource-Constrained IoT Devices: A Review, Comparison and Research Opportunities. *IEEE Access* **2021**, *9*, 28177–28193. [CrossRef]
9. Patel, D.; Muresan, R. Triple-DES ASIC Module for a Power-Smart System-on-Chip Architecture. In Proceedings of the 2006 Canadian Conference on Electrical and Computer Engineering, Ottawa, ON, Canada, 7–10 May 2006.
10. Rolfes, C.; Poschmann, A.; Leander, G.; Paar, C. Ultra-Lightweight Implementations for Smart Devices—Security for 1000 Gate Equivalents. In *Proceedings of the Smart Card Research and Advanced Applications*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 89–103.
11. Feldhofer, M.; Rechberger, C. A Case Against Currently Used Hash Functions in RFID Protocols. In *OTM Confederated International Conferences "On the Move to Meaningful Internet Systems"*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 372–381.
12. McKay, K.; Bassham, L.; Turan, M.S.; Mouha, N. *Report on Lightweight Cryptography*; Technical Report; NIST Interagency/Internal Report (NISTIR); National Institute of Standards and Technology: Gaithersburg, MD, USA, 2017.
13. Singh, S.; Sharma, P.K.; Moon, S.Y.; Park, J.H. Advanced Lightweight Encryption Algorithms for IoT Devices: Survey Challenges and Solutions. *J. Ambient Intell. Hum. Comput.* **2017**, *4*, 1–18. [CrossRef]
14. *ISO/IEC 29192-2:2019*; Information Security—Lightweight Cryptography—Part 2: Block Ciphers. International Organization for Standardization: London, UK, 2019.
15. Le, D.N.; Baek, S.; Choi, K.U.; Hong, J.P. An Area Optimization and Power Efficient Method for HMAC-PHOTON Lightweight Cryptography. In Proceedings of the 31st Hot Chips Symposium 2019, Cupertino, CA, USA, 18–20 August 2019.
16. Le, D.N. An Area Efficient and Low Power Entity Authentication SoC Based on Physical Uncloneable Function for IoT Device. Master's Thesis, School of Electrical Engineering, Chungbuk National University, Cheongju, Korea 2020.
17. Nam, J.W.; Ahn, J.H.; Hong, J.P. Compact SRAM-Based PUF Chip Employing Body Voltage Control Technique. *IEEE Access* **2022**, *10*, 22311–22319. [CrossRef]
18. Choi, K.U.; Baek, S.; Heo, J.; Hong, J.P. A 100% Stable Sense-Amplifier-Based Physically Unclonable Function with Individually Embedded Non-Volatile Memory. *IEEE Access* **2020**, *8*, 21857–21865. [CrossRef]
19. Baek, S.; Yu, G.H.; Kim, J.; Ngo, C.T.; Eshraghian, J.K.; Hong, J.P. A Reconfigurable SRAM Based CMOS PUF with Challenge to Response Pairs. *IEEE Access* **2021**, *9*, 79947–79960. [CrossRef]
20. Nam, J.W.; Kim, J.; Hong, J.P. Stochastic Cell- and Bit-Discard Technique to Improve Randomness of a TRNG. *Electronics* **2022**, *11*, 1735. [CrossRef]
21. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A.; Robshaw, M.J.B.; Seurin, Y.; Vikkelsoe, C. PRESENT: An Ultra-Lightweight Block Cipher. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2007, Vienna, Austria, 10–13 September 2007; Paillier, P., Verbauwhede, I., Eds.; Springer: Berlin/Heidelberg, Germany, 2007; pp. 450–466.
22. Guo, J.; Peyrin, T.; Poschmann, A. The PHOTON Family of Lightweight Hash Functions. In Proceedings of the Advances in Cryptology—CRYPTO 2011, Santa Barbara, CA, USA, 14–18 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 222–239.
23. Daemen, J.; Peeters, M.; Assche, G.; Rijmen, V. The noekeon block cipher. In Proceedings of the First Open NESSIE Workshop 2000, Leuven, Belgium, 13–14 November 2000; pp. 1–30.
24. Leander, G.; Paar, C.; Poschmann, A.; Schramm, K. New Lightweight DES Variants. In *International Workshop on Fast Software Encryption, Luxembourg, 26–28 March 2007*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 196–210.
25. Aoki, K. Camellia: A 128-bit block cipher suitable for multiple platforms—Design and analysis. In *Proceedings of the 7th Annual International Workshop, SAC 2000, Waterloo, ON, Canada, 14–15 August 2000*; Springer: Berlin/Heidelberg, Germany, 2000; pp. 39–56.
26. Shirai, T.; Shibutani, K.; Akishita, T.; Moriai, S.; Iwata, T. The 128-bit blockcipher CLEFIA. In Proceedings of the 14th International Workshop, Luxembourg, 26–28 March 2007; pp. 181–195.
27. Knudsen, L.; Leander, G.; Poschmann, A.; Robshaw, M.J. PRINT-cipher: A block cipher for IC-printing. In Proceedings of the 12th International Workshop, Santa Barbara, CA, USA, 17–20 August 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 16–32.

28. Bogdanov, A.; Knežević, M.; Leander, G.; Toz, D.; Varıcı, K.; Verbauwhede, I. spongent: A Lightweight Hash Function. In Proceedings of the Cryptographic Hardware and Embedded Systems—CHES 2011, Nara, Japan, 28 September–1 October 2011; Preneel, B., Takagi, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 312–325.
29. Hirose, S.; Ideguchi, K.; Kuwakado, H.; Owada, T.; Preneel, B.; Yoshida, H. A Lightweight 256-Bit Hash Function for Hardware and Low-End Devices: Lesamnta-LW. In Proceedings of the Information Security and Cryptology—ICISC 2010, Seoul, Korea, 1–3 December 2010; Rhee, K.H., Nyang, D., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 151–168.
30. *ISO/IEC 29192-5:2016*; Information Technology—Security Techniques—Lightweight Cryptography—Part 5: Hash-Functions. International Organization for Standardization: London, UK, 2016.
31. *FIPS 81*; DES Modes of Operation. US Department of Commerce: Washington, DC, USA, 1980; Federal Information Processing Standard (FIPS), Publication 81, National Bureau of Standards.
32. Dworkin, M. *Recommendation for Block Cipher Modes of Operation: The XTS-AES Mode for Confidentiality on Storage Devices*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2010.
33. Dworkin, M. *NIST Special Publication 800-38C: Recommendation for Block Cipher Modes of Operation: The CCM Mode for Authentication and Confidentiality*; Technical Report; National Institute of Standards and Technology, U.S. Department of Commerce: Gaithersburg, MD, USA, 2004.
34. National Institute for Science Technology (NIST). *Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*; Technical Report; National Institute of Standards and Technology: Gaithersburg, MD, USA, 2007.
35. Prathiba, A.; Bhaaskaran, V.S.K. FPGA Implementation and Analysis of the Block Cipher Mode Architectures for the PRESENT Light Weight Encryption Algorithm. *Indian J. Sci. Technol.* **2016**, *9*, 1–8. [CrossRef]
36. Hamalainen, P.; Alho, T.; Hannikainen, M.; Hamalainen, T. Design and Implementation of Low-Area and Low-Power AES Encryption Hardware Core. In Proceedings of the 9th EUROMICRO Conference on Digital System Design (DSD'06), Dubrovnik, Croatia, 30 August–1 September 2006; pp. 577–583.
37. Bellare, M.; Canetti, R.; Krawczyk, H. Keying Hash Functions for Message Authentication. In Proceedings of the Advances in Cryptology—CRYPTO '96, Santa Barbara, CA, USA, 18–22 August 1996; Koblitz, N., Ed.; Springer: Berlin/Heidelberg, Germany, 1996; pp. 1–15.
38. He, Y.; Li, S. A 3DES implementation especially for CBC feedback loop mode. In Proceedings of the 2017 IEEE International Symposium on Circuits and Systems (ISCAS), Baltimore, MD, USA, 28–31 May 2017; pp. 1–4.