Computers and Electrical Engineering 116 (2024) 109168

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Efficient implementations of CRAFT cipher for Internet of Things

Jiahao Xiang, Lang Li *

College of Computer Science and Technology, Hengyang Normal University, Hengyang, 421002, China
Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang Normal University, Hengyang, 421002, China

ARTICLE INFO

Keywords:
Internet of Things
Lightweight block cipher
Field Programmable Gate Arrays(FPGA)
Low-area
High-throughput

ABSTRACT

The rapid growth of the Internet of Things (IoT) highlights the importance of lightweight cryptography in maintaining security. However, enhancing performance while ensuring the same level of security remains a significant challenge. This paper presents two innovative architectures for the CRAFT lightweight block cipher, aiming to enhance performance without compromising security. The novel Serial and Unrolled architectures are introduced to achieve low area usage and high throughput, respectively. Specifically, the Serial architecture reduces the datapath from 64-bit to 4-bit, significantly decreasing the area. The Unrolled architecture, on the other hand, minimizes latency from 32 to 16. Additionally, Boolean satisfiability (SAT) solvers are employed to identify a lower-cost area implementation of the S-Box. The proposed designs underwent evaluation on three distinct FPGA platforms: Artix-7, Kintex-7, and Spartan-7. The results show that the low area design reduces area usage by 10.16% compared to the previous design. Additionally, the S-Box implementation achieves a significant area reduction of 28.9%. On the other hand, the unrolled design enhances the maximum throughput by 40.53% compared to the previous design. Therefore, the proposed designs could offer enhanced performance while maintaining security for IoT devices.

1. Introduction

The Internet of Things (IoT) is rapidly integrating into various aspects of everyday life. With this advancement, a growing number of security issues are emerging. These security concerns are extensively discussed in [1]. To ensure data protection, the cryptography techniques outlined in [2] are recommended.

However, the resource constraints of many IoT devices pose challenges for the implementation of robust security measures. These devices often have limited memory, processing power, and energy. Therefore, the security measures need to be lightweight to ensure they do not overburden the resources. Lightweight cryptography, a subset of cryptography, provides solutions specifically designed for these resource-limited devices, as discussed in [3].

The field of lightweight cryptography has received considerable attention in recent years. Examples of this include PRESENT [4], LED [5], Midori [6], QTL [7], GIFT [8], CRAFT [9], Shadow [10], DULBC [11], IVLBC [12], BipBip [13], and LELBC [14]. More ciphers can be found in [15]. Concurrently, there has been significant research into side channel attacks on lightweight ciphers. This research can provide valuable insights into the physical security of IoT devices.

Differential fault analysis, which is a type of side channel attack, was first introduced by [16]. This concept was later elaborated in more detail by [17]. In response to these fault attacks, several countermeasures have been suggested. For example, [18] introduced a scheme for error detection. In the realm of Post-Quantum Cryptography, [19] proposed specific error detection methods. It is

* Correspondence to: Hengyang Normal University, Hengyang, 421002, China.
E-mail address: lilang@hynu.edu.cn (L. Li).

https://doi.org/10.1016/j.compeleceng.2024.109168
Received 9 January 2024; Received in revised form 8 March 2024; Accepted 8 March 2024
Available online 15 March 2024

Table 1
Main notations.

Notation	Description
TK_i	tweakeys used in the ith round
RC_i	round constant for the ith round
R_i	Function for the ith round
SB	Sub-Box
MC	Mix-Columns
PN	PermuteNibbles
PK	Permutation used in key schedule
$\oplus$	XOR operation
$\|\|$	Concatenation operation
$\sim$	Inverse operation
$\wedge$	And operation
$\vee$	Or operation

important to note that these methods do increase the hardware consumption of the cipher system. The CRAFT cipher was designed with resistance to fault attacks in mind. However, it currently lacks efficient implementations. To make it suitable for use in more constrained environments, development of more efficient implementations is necessary.

Efficient implementation allows lightweight ciphers to be used in various settings. A hardware implementation can enhance the performance of these ciphers in resource-limited environments. Several researchers have proposed optimized architectures for various ciphers. Lara-Nino et al. [20] introduced a 16-bit datapath architecture for the PRESENT cipher, which resulted in reduced area and power consumption. Similarly, Pandey et al. [21] suggested an optimized key schedule for the same cipher, leading to a smaller area. Shahbazi et al. [22] put forth an 8-bit serial architecture for AES, which also reduces area and power consumption. Li et al. [23] presented unrolled architectures and a low-cost architecture for PRINCE, optimizing both throughput and area separately. Further enhancements have been made by Bharathi et al. [24], who improved the performance of the PRESENT cipher by expanding the key length. Lastly, Yang et al. [25] shared components in the cipher process for LILLIPUT, resulting in a smaller area.

This work presents the first implementation of CRAFT on FPGA platforms. Two architectures for CRAFT, Serial and Unrolled, are proposed. The Serial architecture reduces the datapath from 64-bit to 4-bit, meaning it only uses one S-Box, which significantly reduces the area usage. The Unrolled architecture reduces the latency of the encryption process, thereby improving the throughput rate. The optimal implementation of the S-Box, aimed at further area reduction, is determined using a SAT solver in conjunction with the GEC encoding scheme. The experiments are conducted on three different FPGA platforms: Artix-7, Kintex-7, and Spartan-7. The source code for the proposed designs is available online.[1] The main contributions of this article are as follows.

- Two architectures for CRAFT, Serial and Unrolled, are proposed. These are optimized for area and throughput, respectively. The Serial architecture reduces the area usage by 10.16% compared to the work of Beierle et al. [9]. The Unrolled architecture doubles the throughput rate compared to the same work.
- The optimal implementation of the S-Box, which results in further area reduction, is identified using a SAT solver. The proposed S-Box implementation achieves a 28.9% area reduction compared to the work of Bao et al. [26].
- The architectures are implemented across three different FPGA platforms: Artix-7, Kintex-7, and Spartan-7. This variety allows engineers to select the platform that best suits their application needs.

The remainder of this article unfolds as follows: Section 2 delves into the specifics of CRAFT. The duo of proposed architectures for CRAFT are explored in Section 3. Section 4 details the metrics and environment used for the experimental evaluation and comprehensive performance analysis. A thorough discussion of all the architectures is provided in Section 5. Lastly, Section 6 encapsulates the work done and points towards potential avenues for future research.

2. Specification of CRAFT

CRAFT is a lightweight tweakable block cipher that operates on a 64-bit plaintext size, a 128-bit key size, and a 64-bit tweak size. It outputs a 64-bit ciphertext. Although quantum computing has enhanced capabilities to attack ciphers, as highlighted by Darzi et al. [27] and Canto et al. [28], it is noteworthy that the probabilistic algorithm based on quantum computing, proposed by Grover et al. [29], could reduce the key space from 128-bit to 64-bit. However, this reduction is not sufficient to brute force attack the CRAFT cipher with current computational capabilities. The confusion and diffusion properties of CRAFT ensure that the distribution of probabilities between the plaintext and ciphertext is independent. For more details on CRAFT, refer to Fig. 1 which depicts its architecture. The encryption process of CRAFT is outlined in Algorithm 1. The decryption process is similar to the encryption process, with the only difference being that the round keys are applied in reverse order. The main notations used throughout this paper are outlined in Table 1.

[1] https://github.com/xjh2000/craft_implementation

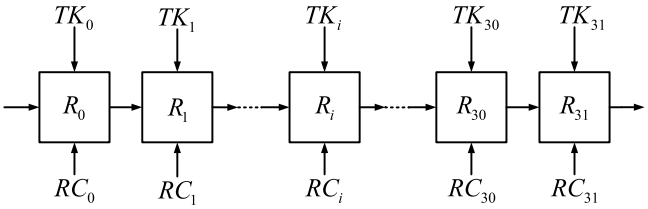

Fig. 1. Architecture of CRAFT.

Algorithm 1 CRAFT Encryption Process

Input: Plaintext P, Key $K_0||K_1$, Tweak T
Output: Ciphertext C

$TK_0 \leftarrow K_0 \oplus T$
$TK_1 \leftarrow K_1 \oplus T$
$TK_2 \leftarrow K_0 \oplus PK(T)$
$TK_3 \leftarrow K_1 \oplus PK(T)$
$C \leftarrow P$
for $i \leftarrow 0$ to 31 **do**
    $C \leftarrow MC(C)$
    $C_{4,5} \leftarrow C_{4,5} \oplus RC_i$
    $C \leftarrow C \oplus TK_{i \bmod 4}$
    if $i \neq 31$ **then**
        $C \leftarrow PN(C)$
        $C \leftarrow SB(C)$
    end if
end for

Table 2
S-Box of CRAFT.

Input	Output	Input	Output
0	c	8	8
1	a	9	9
2	d	a	1
3	3	b	5
4	e	c	0
5	b	d	2
6	f	e	4
7	7	f	6

The round function is composed of three distinct operations: Mix-Columns, PermuteNibbles, and Sub-Box. The Mix-Columns operation is a linear transformation that multiplies the input column by a constant matrix, M, to generate the output column. Notably, M is an involutory matrix, as shown in Eq. (1).

$$M = \begin{bmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \tag{1}$$

The PermuteNibbles operation, an involutory permutation, operates on 4-bit nibbles. This operation triggers additional S-Boxes, thereby bolstering the cipher's security. The illustration of the PermuteNibbles operation is provided in Eq. (2). The permutation PK is utilized in the key schedule, as depicted in Eq. (3).

$$PN = [15, 12, 13, 14, 10, 9, 8, 11, 6, 5, 4, 7, 1, 2, 3, 0] \tag{2}$$

$$PK = [12, 10, 15, 5, 14, 8, 9, 2, 11, 3, 7, 4, 6, 0, 1, 13] \tag{3}$$

The Sub-Box operation, a nonlinear transformation, introduces confusion into the cipher. This operation is performed using a 4-bit S-Box. The values are represented in hexadecimal notation, as shown in Table 2.

Two Linear Shift Feedback Registers (LSFRs), a and b, are used to concatenate the round constants. The round constants is defined as $RC = (a_3, a_2, a_1, a_0, 0, b_2, b_1, b_0)$. The initial round constant, RC_0, is set to 0×11.

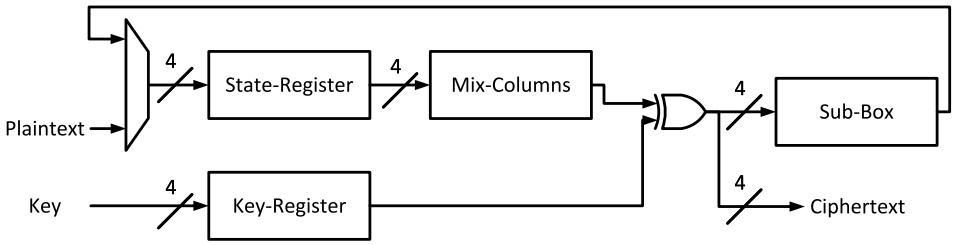

Fig. 2. Serial architecture of CRAFT.

The iterative architecture, often referred to as the round-based architecture, was first introduced in the original CRAFT paper [9]. This architecture works by executing one round function per cycle. It employs a single round function to encrypt a block. This function includes operations such as Mix-Columns, PermuteNibbles, Sub-Box, AddConstant, and AddTweakey. To encrypt a single block, this round function is run over 32 cycles. In the meantime, the Key Schedule runs concurrently with the round function.

3. Implementations

For the first time, the components of CRAFT have been optimized to achieve efficient area and throughput, resulting in two proposed implementation architectures: Serial and Unrolled.

3.1. Serial architecture (SA)

The purpose of the serial architecture is to reduce the datapath, which represents the number of bits dealt with in one cycle. For instance, the CRAFT cipher has a 64-bit block size, meaning it can process 64-bit data in one cycle when using the iterative architecture. In contrast, the serial architecture reduces the datapath from 64-bit to 4-bit, meaning it can only process 4-bit data in one cycle. Despite this limitation, serial architectures can significantly reduce area usage by reusing components, making them a viable alternative to iterative architectures. For example, the quantity of S-Boxes is diminished from 16 to 1. The clock gating technique is also employed to enable each component and minimize the energy consumption of encryption.

The architecture, depicted in Fig. 2, comprises a single Sub-Box, a 4-bit Mix-columns, and two register banks. The Key-Register is used to store keys. The State-Register, on the other hand, is used to store plaintext. They also hold intermediate results temporarily. The design incorporates a feedback path to store intermediate results in the State-Register bank. Additionally, the PermuteNibbles function is integrated into the State-Register bank.

3.1.1. S-Box optimization

The S-Box is a crucial component of the encryption algorithm, adding to its complexity. However, it also demands a significant amount of area. There are several ways to implement the S-Box. One prevalent approach is to use a lookup table (LUT), a technique described by Lara-Nino et al. [20]. This approach, while effective, requires many flip-flops, which can lead to a substantial increase in area consumption. An alternative method is to use the logical equivalent expression of the S-Box. This method, suggested by Bao et al. [26] and Feng et al. [30], can help reduce area consumption.

Boolean satisfiability (SAT) solvers can be used to find S-Boxes that meet specific implementation requirements, such as meet to certain software or hardware implementation requirements. To elaborate, the S-Box implementation can be encoded as Boolean constraints. This is done by representing the S-Box as a truth table and using Boolean variables to denote the input and output bits of the S-Box. The constraints are then formulated based on the desired properties of the S-Box. Once the S-Box properties are encoded as Boolean constraints, these constraints can be satisfied by a SAT solver. This assignment corresponds to an S-Box that fulfills the desired properties.

A measure of the number of logical gates required to implement the Boolean formula that represents a SAT instance is the Gate Equivalent Complexity (GEC). To calculate the GEC, the Boolean formula is converted into a circuit of logical gates, such as AND, OR, and NOT gates. The total count of these gates in the circuit gives the GEC of the instance. In this design, the GEC encoding scheme from Feng et al. [30] is optimized and used to implement the S-Box. The encoding scheme is detailed in Eqs. (4):

$$
\begin{aligned}
&\forall i \in \{0, 1, \ldots, K-1\} : \\
&T_i = F_{if}(GT_i[0], \sim (X_{4i} \cdot X_{4i+1}) \cdot \sim X_{4i+2} \cdot X_{4i+3}) \\
&\quad + F_{if}(GT_i[1], X_{4i+2} \cdot (X_{4i} + X_{4i+1})) \\
&\quad + F_{if}(GT_i[2], X_{4i} \cdot X_{4i+1} \cdot X_{4i+2}) \\
&\quad + F_{if}(GT_i[3], X_{4i+2}) + F_{if}(GT_i[4], X_{4i}) \\
&\quad + F_{if}(GT_i[5], X_{4i} \cdot X_{4i+1}) \\
&\quad + F_{if}(GT_i[6], X_{4i} + X_{4i+1}) + F_{if}(GT_i[7], max)
\end{aligned}
\tag{4}
$$

Table 3
Encoding of different types of logical gates.

Logical expression	GT_i	Gate type
$X_0 \oplus X_1$	2	XOR
$\sim (X_0 \oplus X_1)$	3	XNOR
$X_0 \wedge X_1$	4	AND
$\sim (X_0 \wedge X_1)$	5	NAND
$X_0 \vee X_1$	6	OR
$\sim (X_0 \vee X_1)$	7	NOR
$\sim X_0$	9	NOT
$\sim X_1$	11	NOT
$\sim X_2$	17	NOT
$X_0 \oplus X_1 \oplus X_2$	18	XOR3
$\sim (X_0 \oplus X_1 \oplus X_2)$	19	XNOR3
$X_0 \wedge X_1 \wedge X_2$	32	AND3
$\sim (X_0 \wedge X_1 \wedge X_2)$	33	NAND3
$X_0 \vee X_1 \vee X_2$	118	OR3
$\sim (X_0 \vee X_1 \vee X_2)$	119	NOR3
$\sim ((X_0 \wedge X_1) \vee (\sim (X_2 \vee X_3)))$	176	MAOI1
$\sim (\sim (X_0 \wedge X_1) \wedge ((X_2 \vee X_3)))$	177	MOAI1

where K is numbers of the logical gates, $X_{4i} - X_{4i+3}$ is the input of the ith logical gate, T_i is the output of the ith logical gate, and F_{if} is a function that returns the value of the second argument if the first argument is true and returns the value of zero otherwise. The value of max is all one's in the binary expression, which is represented logically as an inverse. GT_i denotes the type of the ith logical gate, represented as an 8-bit binary number. The least significant bit of GT_i is indexed at seven. Table 3 enumerates the various types of logical gates employed in this encoding scheme.

Eqs. (5) display the optimized scheme of the S-Box, where $X_3 - X_0$ represents the input and $Y_3 - Y_0$ represents the output. The proposed S-Box scheme is implemented using four MOAI1 gates, three MAOI1 gates, and one AND3 gate. This configuration of the S-Box module results in a 28.9% reduction in area compared to the method proposed by Bao et al. [26], based on gate equivalent (GE) estimation using the UMC 180 nm library.

$$\begin{aligned}
T_0 &= \mathrm{MAOI1}(X_0, X_1, X_0, X_1)\\
T_1 &= \mathrm{AND3}(X_3, X_2, X_3)\\
T_2 &= \mathrm{MAOI1}(X_1, X_2, X_0, X_3)\\
T_3 &= \mathrm{MOAI1}(X_1, X_0, X_2, X_2)\\
T_4 &= \mathrm{MOAI1}(X_3, T_0, T_3, T_3)\\
T_5 &= \mathrm{MOAI1}(T_3, T_0, X_0, T_1)\\
T_6 &= \mathrm{MAOI1}(X_0, T_0, X_3, T_0)\\
T_7 &= \mathrm{MOAI1}(X_0, T_1, T_2, T_2)\\
Y_3 &= T_4 \quad Y_2 = T_6 \quad Y_1 = T_7 \quad Y_0 = T_5
\end{aligned} \tag{5}$$

3.1.2. Mix-Columns optimization

The Mix-Columns component is a linear transformation of the input column. The output column is generated by multiplying the input column with a constant matrix M. M is a involutory matrix, which means $M^2 = E$, where E is the identity matrix. It is easy to decrypt the ciphertext by multiplying the ciphertext with M again. Eq. (6) illustrates the Mix-columns component. Here, $I_{3,j}$, $I_{2,j}$, $I_{1,j}$, and $I_{0,j}$ represent the input column, while $I'_{3,j}$, $I'_{2,j}$, $I'_{1,j}$, and $I'_{0,j}$ denote the output column. The column index is given by j, where j ranges from 0 to 3.

$$\begin{bmatrix} I'_{3,j} \\ I'_{2,j} \\ I'_{1,j} \\ I'_{0,j} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 \end{bmatrix} \begin{bmatrix} I_{3,j} \\ I_{2,j} \\ I_{1,j} \\ I_{0,j} \end{bmatrix} \tag{6}$$

In order to reduce the area of this component, the serial architecture of Mix-Columns is utilized, as shown in Fig. 3. The serial architecture of Mix-Columns requires four 4-bit registers, two multiplexers, and three XOR gates. The operation of Mix-Columns involves three distinct stages: freeze, shift, and add. During the freeze stage, the register values are kept unchanged by setting both CM_0 and CM_1 to 0. In the shift stage, a shift in the register values from RM_0 to RM_4 is induced by setting both CM_0 and CM_1 to 1. Finally, in the add stage, an addition operation on the column values is executed according to Eq. (6). This is achieved by setting CM_0 and CM_1 to 0 and 1, respectively.

Fig. 4 presents the timing diagram for the serial architecture of the Mix-Columns operation. It requires five clock cycles to compute the next columns from the previous ones, and an additional four clock cycles to transfer data from the internal register of Mix-Columns to the State-Register. Therefore, a complete state round requires a total of 36 clock cycles.

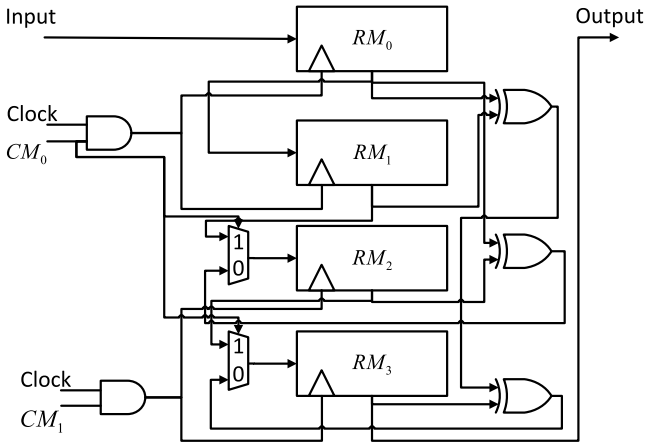

Fig. 3. Serial architecture of Mix-Columns with clock gating.

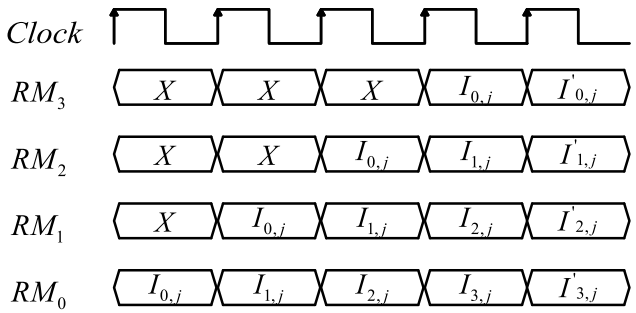

Fig. 4. Timing diagram for the Serial Architecture of Mix-Columns.

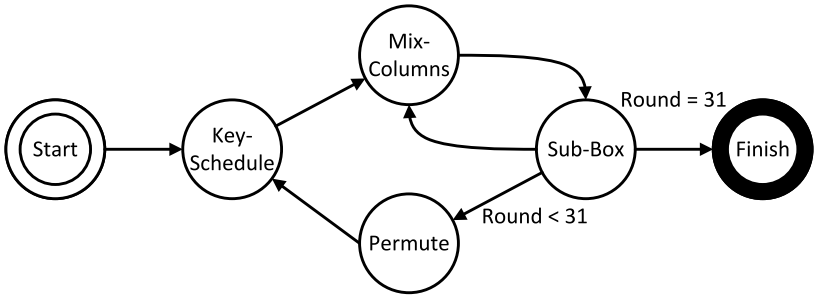

Fig. 5. Finite-state machine for Serial Architecture.

3.1.3. Control units

As depicted in Fig. 5, the finite-state machine (FSM) initiates the encryption process by loading the initial key into the Key-Register and the plaintext into the State-Register. During the Key-Schedule phase, the key is expanded while the gate clocks of the Mix-Columns and State-Register are turned off. Next, the Mix-Columns phase begins, where one column of the State-Register is stored in the Mix-Columns registers. The Mix-Columns operation on one column takes five clock cycles to execute in this phase. Once this phase is finished, the gate clocks for the State-Register and Key-Register are turned off. Following this, the Sub-Box phase commences. During this phase, the data from the Mix-Columns registers is transferred again to the State-Register and XORed with the keys. This process requires an additional four clock cycles. This cycle between the Mix-Columns and Sub-Box phases is repeated four times for the four columns of the State-Register. Subsequently, the Permute operation is carried out within the State-Register, requiring a single clock cycle. The encryption process concludes when the Round counter reaches 31, at which point the ciphertext is stored in the State-Register.

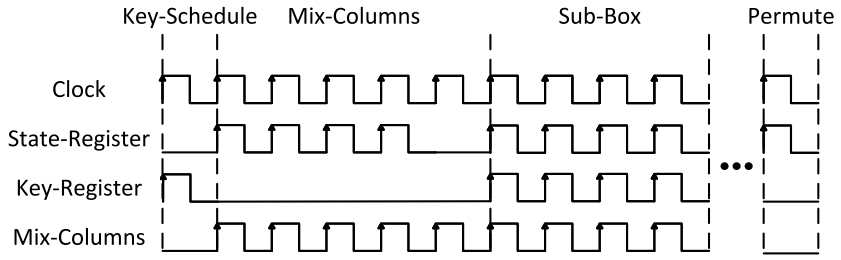

Fig. 6. Timing diagram for Serial Architecture.

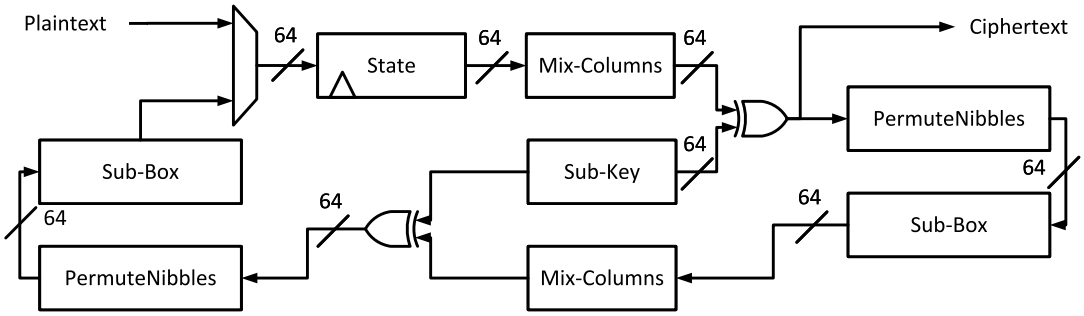

Fig. 7. Unrolled architecture of CRAFT.

In the work of Shahbazi et al. [22], it is discussed that the dynamic power consumption of the encryption process can be mitigated through the use of clock gating. The clock gating technique is independently applied to the State-Register, Key-Register, and Mix-Columns. For instance, in the Permute phase, the gate clocks of the Key-Register and Mix-Columns are disabled as these components are not in use. This helps save a significant amount of power. Fig. 6 shows the timing diagram of a design that uses the clock gating technique.

3.2. Unrolled architecture (UA)

The unrolled architecture allows for the execution of more than one round function in a single cycle. In contrast, the iterative architecture of the CRAFT cipher only runs one round function per cycle. This approach can significantly reduce the latency of the encryption process. It does this by reducing the number of cycles needed to encrypt one block size plaintext, thereby improving the throughput rate.

The unrolled architecture shown in Fig. 7 includes two Sub-Boxes, two Mix-Columns, a Key, a State-Register, two PermuteNibbles, and one feedback path. It is designed to perform a 30-round encryption process in just 15 cycles. Only the Mix-Columns and Add-Key operations are carried out in the final cycle, finishing the encryption process in a total of 16 cycles.

The unrolled architecture, which is based on the iterative architecture from the work of Beierle et al. [9], completes the encryption process in only 16 cycles, compared to the 32 cycles needed by the iterative architecture. Here, a cycle includes two round functions of CRAFT. While this approach might use more area, it provides higher throughput at the same frequency.

4. Experimental results

In ASIC implementations, the Gate Equivalent (GE) is often used to evaluate the area consumption of a design. One GE corresponds to the area of a two-input NAND gate. The area is computed in terms of GEs. This is done by dividing the total area (measured in μm^2) by the area of a two-input NAND gate (also measured in μm^2). However, the number of GEs can vary depending on the specific technology used, as Turan et al. discuss in their work [31]. For instance, the number of GEs for the same design will differ between UMC 180 nm technology and TSMC 180 nm technology. Therefore, GE is not suitable for comparing the area consumption of different designs on different technologies. For a fair comparison, the area consumption of the proposed designs is assessed using FPGA implementations. This is a technique similarly employed in the study by Mohajerani et al. [3].

Table 4
Description of different architectures.

Architecture	Description	Reference
SA	Serial Architecture	This work
UA	Unrolled Architecture	This work
IA	Iterative Architecture	[9]

4.1. Platform

The designs proposed in this study were implemented on a Xilinx FPGA board, utilizing the Vivado v2023.2 software for deployment. Benchmarking was performed across three distinct FPGA platforms to ensure a diverse testing environment: Artix-7(xc7a100tcsg324-1), Kintex-7(xc7k70tfbg484-1), and Spartan-7(xc7s100fgga484-1). Artix-7 offers high performance in resource-limited situations. Spartan-7 is designed for high-restriction environments. Kintex-7 is well-suited for use in applications such as 3G and 4G wireless technologies.

4.2. Area

The area consumed by the proposed designs is quantified using the Area metric, which encompasses components such as Flip-Flops, LUTs, and Slices. For a balanced comparison, the embedded memory blocks of the FPGA were not utilized. This was achieved by disabling the relevant settings in the VHDL, as recommended in the design guidelines. Also, all designs were synthesized and implemented using the same settings, specifically, the default settings of Vivado Synthesis and Implementation.

4.3. Throughput

The efficiency of the proposed designs is assessed using the Throughput metric. This metric uses three parameters: the maximum throughput rate, the throughput rate at 100 MHz, and the throughput rate per slice. The maximum throughput rate is the highest rate that our designs can achieve, calculated using Eq. (7). The throughput rate at 100 MHz shows the rate achievable when the clock frequency is set to 100 MHz, calculated using Eq. (8). The throughput rate per slice is a measure of efficiency, calculated by dividing the throughput rate by the number of Slices, as defined in Eq. (9). In these computations, the Plaintext Size is set to 64-bit. Latency denotes the count of clock cycles needed to encrypt a single block, and Slices represent the quantity of Slices consumed by the design.

$$MaxThroughput(Thr) = \frac{MaxFrequency \times BlockSize}{Latency} \tag{7}$$

$$Throughput_{@100\ \mathrm{MHz}}(Thr^{*}) = \frac{100\ \mathrm{MHz} \times BlockSize}{Latency} \tag{8}$$

$$ThroughputPerSlice = \frac{Thr}{Slices} \tag{9}$$

4.4. Power and energy

The Power metric, which includes both dynamic and static power consumption, is used to evaluate the power consumption of the proposed designs, as defined in Eq. (10). On the other hand, the Energy metric measures the energy consumption of the designs. It is calculated by multiplying the power consumption by the time needed to encrypt a single block. This time is determined by dividing the latency by the frequency, as explained in Eq. (11).

$$\begin{aligned}\text{Total Power}(TP) = {} & \text{Dynamic Power}(DP) \\ & + \text{Static Power}(SP)\end{aligned} \tag{10}$$

$$Energy(E) = \frac{TP \times Latency}{Frequency} \tag{11}$$

4.5. Results

This section presents the results of the proposed designs, focusing on three key aspects: area consumption, throughput performance, and power and energy metrics. These results are demonstrated across three different FPGA platforms, namely Artix-7, Kintex-7, and Spartan-7. The various architectures used in this study are described in Table 4. For a comparison of area consumption across these designs, refer to Table 5. Throughput results are provided in Table 6, while Table 7 contains information on power and energy consumption.

Table 5
Area used in three architectures.

Platform	Design	$State(bit)$	$Key(bit)$	FF	LUT	$Slices$
Artix-7	SA	64	128	**144**	**177**	**59**
	UA	64	128	157	378	111
	IA	64	128	159	201	65
Kintex-7	SA	64	128	**144**	**178**	**58**
	UA	64	128	157	377	115
	IA	64	128	159	205	66
Spartan-7	SA	64	128	**144**	**177**	**57**
	UA	64	128	157	381	118
	IA	64	128	158	200	64

Table 6
Throughput results in three architectures.

Platform	Design	$Latency$	$MaxF$(MHz)	Thr(Mbps)	Thr^*(Mbps)[a]	$Thr/Slices$(Kbps/Slices)
Artix-7	SA	1215	**557.41**	29.36	5.27	497.65
	UA	**16**	142.38	**569.52**	**400.00**	5130.81
	IA	32	202.63	405.26	200.00	**6234.77**
Kintex-7	SA	1215	**853.97**	44.98	5.27	775.57
	UA	**16**	175.25	**701.00**	**400.00**	6095.65
	IA	32	301.38	602.76	200.00	**9132.73**
Spartan-7	SA	1215	**525.76**	27.69	5.27	485.87
	UA	**16**	138.86	**555.44**	**400.00**	4707.12
	IA	32	186.98	373.96	200.00	**5843.13**

[a] Throughput rate at 100 MHz.

Table 7
Power and energy consumption in three architectures.

Platform	Design	DP(mW)	SP(mW)	TP(mW)	E(uJ)	E/bit(nJ/bit)
Artix-7	SA	**2.00**	139.00	**141.00**	1.71	26.77
	UA	7.00	139.00	146.00	**0.02**	**0.37**
	IA	3.00	139.00	142.00	0.05	0.71
Kintex-7	SA	**2.00**	145.00	**147.00**	1.79	27.91
	UA	8.00	145.00	153.00	**0.02**	**0.38**
	IA	3.00	145.00	148.00	0.05	0.74
Spartan-7	SA	**2.00**	140.00	**142.00**	1.73	26.96
	UA	7.00	140.00	147.00	**0.02**	**0.37**
	IA	3.00	140.00	143.00	0.05	0.72

DP: Dynamic Power & SP: Static Power & TP: Total Power & E: Energy.

The area consumption of the proposed designs is evaluated based on three factors: Flip-Flops (FF), Look-Up Tables (LUT), and Slices. These designs are compared with the iterative architecture of CRAFT, as detailed in the work of Beierle et al. [9]. The results indicate that the proposed serial architecture consume 10.16% less area than the iterative architecture of CRAFT.

Regarding the FF, the key schedule of the CRAFT cipher is implemented using multiplexers. This eliminates the need for FF to store the sub-key, resulting in a lower FF count compared to other ciphers. This is a significant factor contributing to the CRAFT cipher's requirement of less than 1000 GE, which is the lowest known requirement on the IBM 130 nm ASIC library, as demonstrated in the study by Beierle et al. [9]. A comparison of FF counts is provided in Fig. 8.

As illustrated in Fig. 9, when it comes to LUT, the proposed serial architecture require fewer LUTs than the iterative architecture of CRAFT. This is attributed to the fact that the proposed designs utilize a single S-Box, in contrast to the 16 S-Boxes used by the iterative architecture of CRAFT. Furthermore, the proposed designs also require fewer LUTs than the unrolled architecture of CRAFT, which uses 32 S-Boxes, compared to just one in the proposed designs.

Thanks to the reduction in Flip-Flop (FF) and Look-Up Table (LUT) usage, the serial architecture has fewer slices compared to the iterative architecture of CRAFT. In terms of Slices efficiency, Spartan-7 outperforms both Artix-7 and Kintex-7 platforms. These results are illustrated in Fig. 10. However, the lower Max Frequency of Spartan-7 will be considered in the Throughput comparison.

Fig. 11 illustrates that the proposed serial architecture have a higher Max Frequency than the iterative architecture of CRAFT. This improvement is due to two key factors. First, the S-Box is optimized with the GEC encoding scheme, reducing its delay. Second, the serial architecture of the design further reduces the overall delay of the encryption process. However, among all platforms, Spartan-7 has the lowest Max Frequency, primarily because it has the fewest LUTs, as shown in Fig. 9. The unrolled architecture reduces the latency to 16. This effectively doubles the throughput rate at 100 MHz when compared to the iterative architecture of CRAFT. Additionally, the unrolled architecture improves the maximum throughput by 40.53% on the Artix-7 platform. This data is presented in Table 6.

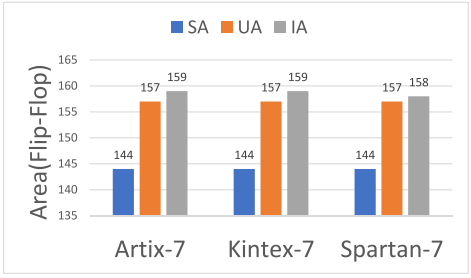

Fig. 8. Comparison of flip-flop in three architectures.

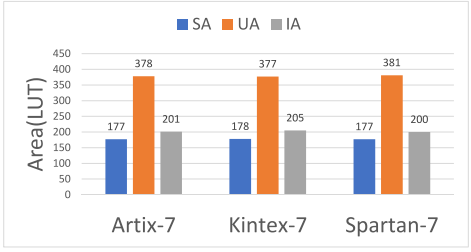

Fig. 9. Comparison of look-up tables in three architectures.

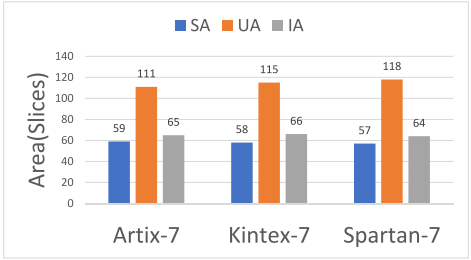

Fig. 10. Comparison of slices in three architectures.

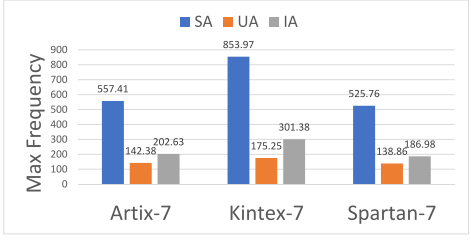

Fig. 11. Comparison of max frequency in three architectures.

The serial architecture has the highest energy per bit due to its higher latency, which results in the smallest area. Conversely, the unrolled architecture has the lowest energy per bit because it has the lowest latency. The energy per bit of the iterative architecture of CRAFT falls between that of the serial architecture and the unrolled architecture. Compared to the iterative architecture of CRAFT, the unrolled architecture reduces energy per bit by 47.89%. Fig. 12 illustrates the energy per bit for the three architectures.

5. Discussion

This section discusses the performance of three different architectures of CRAFT: the serial architecture, the unrolled architecture, and the iterative architecture. These architectures are compared and analyzed to determine which one is best suited for different environments.

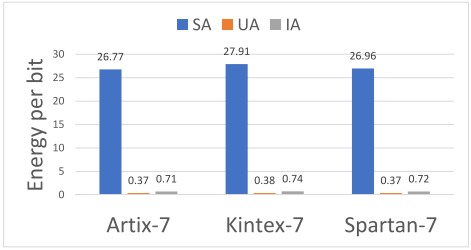

Fig. 12. Comparison of energy per bit in three architectures.

The serial architecture of CRAFT is designed to minimize area consumption. It has the lowest area consumption among the three architectures, which results in it having the lowest dynamic power. Additionally, it boasts the highest frequency among the three architectures. However, it has a high latency, leading to the lowest maximum throughput among the three architectures. The serial architecture is suitable for resource-limited environments where high throughput is not a requirement.

The unrolled architecture of CRAFT aims to maximize throughput. It boasts the lowest latency among the three architectures, which contributes to its highest maximum throughput, despite having the lowest maximum frequency. This architecture is also energy-efficient, offering the lowest energy per bit among the three architectures. However, it does have the highest area consumption. The unrolled architecture is best suited for environments where high throughput and low energy are priorities, and low area is not a requirement.

The iterative architecture of CRAFT is designed to strike a balance between area consumption and throughput. While it does not have the lowest area consumption, the highest frequency, the lowest latency, the highest maximum throughput, or the lowest energy per bit among the three architectures, it does have the highest throughput per slice. The iterative architecture is suitable for environments where moderate throughput is required at the lowest possible area cost.

6. Conclusion

Given the diverse performance requirements arising from the use of IoT devices in various contexts, achieving optimal security without compromising performance presents a significant challenge. Implementing effective solutions is one way to attain this balance between security and performance.

This research presents two unique designs for the CRAFT Lightweight cipher – Serial and Unrolled – both aimed at boosting performance. The Serial architecture reduces the area consumption by 10.16% compared to the iterative architecture of CRAFT. The Unrolled architecture reduces the latency to 16. This effectively doubles the throughput rate at 100 MHz compared to the iterative architecture of CRAFT, resulting in a 40.53% improvement in maximum throughput. Furthermore, the Unrolled architecture also reduces the energy per bit by 47.89% when compared to the iterative architecture of CRAFT. To the best of our knowledge, the Serial architecture establishes a new record for area efficiency on an FPGA configured with a 64-bit block size and 128-bit key size. These proposed architectures are therefore highly suitable for environments with IoT devices.

Future work could extend the proposed architectures to other lightweight ciphers and examine their performance. Additionally, these lightweight ciphers could be implemented in a way that makes them resistant to side channel attacks.

CRediT authorship contribution statement

Jiahao Xiang: Conceptualization, Methodology, Software, Data curation, Writing – original draft, Visualization, Investigation. **Lang Li:** Software, Validation, Supervision, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

The source code related to this paper has been made publicly available online. The URL can be found within the paper itself.

Acknowledgments

This research is supported by the Hunan Provincial Natural Science Foundation of China (2022JJ30103), “the 14th Five Year Plan” Key Disciplines and Application-oriented Special Disciplines of Hunan Province, China (Xiangjiaotong [2022] 351), the Science and Technology Innovation Program of Hunan Province, China (2016TP1020).

References

[1] 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(5):8182–201. http://dx.doi.org/10.1109/JIOT.2019.2935189.

[2] Swessi D, Idoudi H. A survey on Internet-of-Things security: Threats and emerging countermeasures. Wirel Pers Commun 2022;124(2):1557–92. http://dx.doi.org/10.1007/S11277-021-09420-0.

[3] Mohajerani K, Haeussler R, Nagpal R, Farahmand F, Abdulgadir A, Kaps J, et al. FPGA benchmarking of round 2 candidates in the NIST lightweight cryptography standardization process: Methodology, metrics, tools, and results. IACR Cryptol ePrint Arch 2020;1207.

[4] Bogdanov A, Knudsen LR, Leander G, Paar C, Poschmann A, Robshaw MJB, et al. PRESENT: An ultra-lightweight block Cipher. In: Paillier P, Verbauwhede I, editors. Cryptographic hardware and embedded systems - CHES 2007, 9th international workshop, vienna, Austria, September 10-13, 2007, proceedings. Lecture notes in computer science, vol. 4727, Springer; 2007, p. 450–66. http://dx.doi.org/10.1007/978-3-540-74735-2_{3}{1}.

[5] Guo J, Peyrin T, Poschmann A, Robshaw MJB. The LED block Cipher. In: Preneel B, Takagi T, editors. Cryptographic hardware and embedded systems - CHES 2011 - 13th international workshop, nara, Japan, September 28 - October 1, 2011. proceedings. Lecture notes in computer science, vol. 6917, Springer; 2011, p. 326–41. http://dx.doi.org/10.1007/978-3-642-23951-9_{2}{2}.

[6] Banik S, Bogdanov A, Isobe T, Shibutani K, Hiwatari H, Akishita T, et al. Midori: A block Cipher for low energy. In: Iwata T, Cheon JH, editors. Advances in cryptology - ASIACRyPT 2015 - 21st international conference on the theory and application of cryptology and information security, auckland, New zealand, November 29 - December 3, 2015, proceedings, part II. Lecture notes in computer science, vol. 9453, Springer; 2015, p. 411–36. http://dx.doi.org/10.1007/978-3-662-48800-3_{1}{7}.

[7] Li L, Liu B, Wang H. QTL: A new ultra-lightweight block Cipher. Microprocess Microsyst 2016;45:45–55. http://dx.doi.org/10.1016/j.micpro.2016.03.011.

[8] Banik S, Pandey SK, Peyrin T, Sasaki Y, Sim SM, Todo Y. GIFT: a small present - towards reaching the limit of lightweight encryption. In: Fischer W, Homma N, editors. Cryptographic hardware and embedded systems - CHES 2017 - 19th international conference, taipei, Taiwan, September 25-28, 2017, proceedings. Lecture notes in computer science, vol. 10529, Springer; 2017, p. 321–45. http://dx.doi.org/10.1007/978-3-319-66787-4_{1}{6}.

[9] Beierle C, Leander G, Moradi A, Rasoolzadeh S. CRAFT: Lightweight tweakable block Cipher with efficient protection against DFA attacks. IACR Trans Symmetric Cryptol 2019;2019(1):5–45. http://dx.doi.org/10.13154/TOSC.V2019.I1.5-45.

[10] Guo Y, Li L, Liu B. Shadow: A lightweight block Cipher for IoT nodes. IEEE Internet Things J 2021;8(16):13014–23. http://dx.doi.org/10.1109/JIOT.2021.3064203.

[11] Yang J, Li L, Guo Y, Huang X. DULBC: A dynamic ultra-lightweight block Cipher with high-throughput. Integr 2022;87:221–30. http://dx.doi.org/10.1016/J.VLSI.2022.07.011.

[12] Huang X, Li L, Yang J. IVLBC: An involutive lightweight block Cipher for Internet of Things. IEEE Syst J 2023;17(2):3192–203. http://dx.doi.org/10.1109/JSYST.2022.3227951.

[13] Belkheyar Y, Daemen J, Dobraunig C, Ghosh S, Rasoolzadeh S. BipBip: A low-latency tweakable block Cipher with small dimensions. IACR Trans Cryptogr Hardw Embed Syst 2023;2023(1):326–68. http://dx.doi.org/10.46586/TCHES.V2023.I1.326-368.

[14] Song Q, Li L, Huang X. LELBC: A low energy lightweight block Cipher for smart agriculture. Internet Things 2024;25:101022. http://dx.doi.org/10.1016/j.iot.2023.101022.

[15] Zakaria AA, Azni AH, Ridzuan F, Zakaria NH, Daud M. Systematic literature review: Trend analysis on the design of lightweight block Cipher. J King Saud Univ Comput Inf Sci 2023;35(5):101550. http://dx.doi.org/10.1016/J.JKSUCI.2023.04.003.

[16] Biham E, Shamir A. Differential fault analysis of secret key cryptosystems. In: Lecture notes in computer science, Springer Berlin Heidelberg; 1997, p. 513–25. http://dx.doi.org/10.1007/bfb0052259.

[17] Kermani MM, Azarderakhsh R, Mirakhorli M. Multidisciplinary approaches and challenges in integrating emerging medical devices security research and education. In: 2016 ASEE Annual Conference & Exposition Proceedings. ASEE Conferences; 2016, http://dx.doi.org/10.18260/p.25761.

[18] Kaur J, Canto AC, Kermani MM, Azarderakhsh R. Hardware constructions for error detection in WG-29 stream Cipher benchmarked on FPGA. IEEE Trans Comput-Aided Des Integr Circuits Syst 2024;1. http://dx.doi.org/10.1109/tcad.2023.3338108.

[19] Canto AC, Sarker A, Kaur J, Kermani MM, Azarderakhsh R. Error detection schemes assessed on FPGA for multipliers in lattice-based key encapsulation mechanisms in post-quantum cryptography. IEEE Trans Emerg Top Comput 2023;11(3):791–7. http://dx.doi.org/10.1109/tetc.2022.3217006.

[20] Lara-Nino CA, Diaz-Perez A, Morales-Sandoval M. Lightweight hardware architectures for the present Cipher in FPGA. IEEE Trans Circuits Syst I Regul Pap 2017;64-I(9):2544–55. http://dx.doi.org/10.1109/TCSI.2017.2686783.

[21] Pandey JG, Goel T, Karmakar A. Hardware architectures for PRESENT block Cipher and their FPGA implementations. IET Circuits Devices Syst 2019;13(7):958–69. http://dx.doi.org/10.1049/IET-CDS.2018.5273.

[22] Shahbazi K, Ko S. Area-efficient nano-AES implementation for Internet-of-Things devices. IEEE Trans Very Large Scale Integr Syst 2021;29(1):136–48. http://dx.doi.org/10.1109/TVLSI.2020.3033928.

[23] Li L, Feng J, Liu B, Guo Y, Li Q. Implementation of PRINCE with resource-efficient structures based on FPGAs. Front Inf Technol Electron Eng 2021;22(11):1505–16. http://dx.doi.org/10.1631/FITEE.2000688.

[24] Bharathi R, Parvatham N. Light-weight present block Cipher model for IoT security on FPGA. Intell Autom Soft Comput 2022;33(1):35–49. http://dx.doi.org/10.32604/iasc.2022.020681.

[25] Yang J, Li L, Huang X. Low area and high throughput hardware implementations for the LILLIPUT Cipher. Int J Circuit Theory Appl 2023. http://dx.doi.org/10.1002/cta.3892.

[26] Bao Z, Guo J, Ling S, Sasaki Y. PEIGEN–a platform for evaluation, implementation, and generation of S-boxes. IACR Trans Symmetr Cryptol 2019;330–94. http://dx.doi.org/10.46586/tosc.v2019.i1.330-394.

[27] Darzi S, Ahmadi K, Aghapour S, Yavuz AA, Kermani MM. Envisioning the future of cyber security in post-quantum era: A survey on PQ standardization, applications, challenges and opportunities. 2023, http://dx.doi.org/10.48550/ARXIV.2310.12037, CoRR abs/2310.12037, arXiv:2310.12037.

[28] Canto AC, Kaur J, Kermani MM, Azarderakhsh R. Algorithmic security is insufficient: A comprehensive survey on implementation attacks haunting post-quantum security. 2023, http://dx.doi.org/10.48550/ARXIV.2305.13544, CoRR abs/2305.13544, arXiv:2305.13544.

[29] Grover LK. A fast quantum mechanical algorithm for database search. In: Proceedings of the twenty-eighth annual ACM symposium on theory of computing - STOC '96. ACM Press; 1996, http://dx.doi.org/10.1145/237814.237866.

[30] Feng J, Wei Y, Zhang F, Pasalic E, Zhou Y. Novel optimized implementations of lightweight cryptographic S-boxes via SAT solvers. IEEE Trans Circuits Syst I Regul Pap 2023;1–14. http://dx.doi.org/10.1109/tcsi.2023.3325559.

[31] Turan MS, McKay K, Chang D, Bassham LE, Kang J, Waller ND, et al. Status report on the final round of the NIST lightweight cryptography standardization process. US: National Institute of Standards and Technology; 2023, http://dx.doi.org/10.6028/nist.ir.8454.