Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Implementation of PRINCE with resource-efficient structures based on FPGAs*

Lang LI[†1,2,3], Jingya FENG[†‡1,2], Botao LIU[1,3], Ying GUO[1,3], Qiuping LI[1,3]

[1]*Hunan Provincial Key Laboratory of Intelligent Information Processing and Application, Hengyang Normal University, Hengyang 421002, China*

[2]*College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China*

[3]*College of Computer Science and Technology, Hengyang Normal University, Hengyang 421002, China*

[†]E-mail: lilang911@126.com; fengjyk@126.com

Received Dec. 9, 2020; Revision accepted Mar. 8, 2021; Crosschecked Oct. 12, 2021

Abstract: In this era of pervasive computing, low-resource devices have been deployed in various fields. PRINCE is a lightweight block cipher designed for low latency, and is suitable for pervasive computing applications. In this paper, we propose new circuit structures for PRINCE components by sharing and simplifying logic circuits, to achieve the goal of using a smaller number of logic gates to obtain the same result. Based on the new circuit structures of components and the best sharing among components, we propose three new hardware architectures for PRINCE. The architectures are simulated and synthesized on different programmable gate array devices. The results on Virtex-6 show that compared with existing architectures, the resource consumption of the unrolled, low-cost, and two-cycle architectures is reduced by 73, 119, and 380 slices, respectively. The low-cost architecture costs only 137 slices. The unrolled architecture costs 409 slices and has a throughput of 5.34 Gb/s. To our knowledge, for the hardware implementation of PRINCE, the new low-cost architecture sets new area records, and the new unrolled architecture sets new throughput records. Therefore, the newly proposed architectures are more resource-efficient and suitable for lightweight, latency-critical applications.

Key words: Lightweight block cipher; Field-programmable gate array (FPGA); Low-cost; PRINCE; Embedded security
https://doi.org/10.1631/FITEE.2000688 **CLC number:** TP309

1 Introduction

Low-resource devices have been deployed in a wide range of fields, such as medical treatment, transportation, military affairs, industrial automation, critical infrastructure, and wearable and portable applications. In these fields, because data about private activities are obtained, including personal health status, purchase records, social relationships, and other sensitive information, security of the data has become very important (Xu et al., 2014; Alioto, 2017; Xiong et al., 2019). Due to the limited memory and computing power of these devices, a new branch of cryptography called lightweight cryptography has been developed. Combined with lightweight cryptography, these data can be effectively protected (Dhanda et al., 2020). In the last few years, lightweight cryptography with particularly low implementation cost has attracted widespread attention. More and more lightweight block ciphers have been proposed, such as

‡ Corresponding author

* Project supported by the Scientific Research Fund of Hunan Provincial Education Department, China (Nos. 19A072 and 20C0268), the Science and Technology Innovation Program of Hunan Province, China (No. 2016TP1020), the Application-Oriented Special Disciplines, Double First-Class University Project of Hunan Province, China (No. Xiangjiaotong [2018] 469), the Science Foundation Project of Hengyang Normal University, China (No. 18D23), and the Postgraduate Scientific Research Innovation Project of Hunan Province, China (No. CX20190980)

ORCID: Lang LI, https://orcid.org/0000-0002-4832-4499; Jingya FENG, https://orcid.org/0000-0002-8109-1201

PRINCE (Borghoff et al., 2012), Loong (Liu et al., 2019), BORON (Bansod et al., 2017), SIMECK (Yang et al., 2015), SIMON (Beaulieu et al., 2015), SPECK (Beaulieu et al., 2015), RECTANGLE (Zhang et al., 2015), LBlock (Wu and Zhang, 2011), and PRESENT (Bogdanov et al., 2007).

In recent years, lightweight block ciphers have become increasingly popular in resource-constrained environments, and the hardware implementation of ciphers has attracted widespread attention from researchers (Kitsos et al., 2012; Philip and Vaithiyanathan, 2017; Dahiphale et al, 2020). Hardware architectures aimed at different indicators have been proposed. Some attempted to minimize the circuit implementation (Ayesha et al., 2020; Rashidi, 2020b; Shrivastava et al., 2020), some attempted to maximize the throughput or increase the speed (Feizi et al., 2015; Rashidi, 2019), and others attempted to obtain a trade-off among different indicators (Balderas-Contreras et al., 2008; Abed et al., 2019; Dahiphale et al., 2019). PRINCE, proposed by Borghoff et al. (2012), is the first lightweight block cipher to be optimized with respect to latency, and is suitable for pervasive computing applications. Abbas et al. (2014) first implemented the hardware design of PRINCE on field programmable gate arrays (FPGAs). The results showed that on Virtex-4, the throughput was 2032 Mb/s and the efficiency was 2.126 Mb/s, while on Virtex-6, the throughput was 4184 Mb/s and the efficiency was 8.681 Mb/s. Subsequently, a hardware IP-core of PRINCE cipher with high speed and low resource was implemented on Virtex-4 FPGA in Abbas et al. (2015). Hardware implementations of PRINCE, PRESENT, RECTANGLE, KATAN, SIMON, and SPECK based on an unrolled structure have been compared on Virtex-6 (Maene and Verbauwhede, 2015). The results showed that PRINCE was the fastest among the six lightweight block ciphers and was highly competitive. Rashidi (2020b) proposed two structures of PRINCE: low-cost and two-cycle. The implementation results showed that compared with other existing structures, PRINCE structures had higher throughput and shorter critical path delay.

For resource efficiency, we first analyze the components of PRINCE and the existing hardware implementation structures. Then, we propose new circuit structures of the PRINCE components using a smaller number of logic gates. Based on the new circuit structures and the best sharing among components, we propose three new hardware architectures of PRINCE. Finally, the structures are synthesized on different FPGAs. Performance is measured by important indicators such as the maximum frequency, throughput, and area. Compared with previous structures, the new hardware structures are resource-efficient and suitable for lightweight, latency-critical applications. The detailed contributions of this paper are as follows:

1. Using a smaller number of logic gates, a new structure of multiplication by $\boldsymbol{M}'$ is proposed. Compared with previous methods, the number of XOR gates required by this new structure is reduced by 32.

2. The implementations of RC_i-add and K_1-add have been simplified. Compared with other implementations, these implementations need to store only five round constants ($\mathrm{RC}_{1\text{-to-}5}$) and obtain the value of $K_1 \oplus \alpha$ to achieve the same result.

3. Based on the new structure of the multiplication by $\boldsymbol{M}'$ and the new method of RC_i-add and K_1-add, new low-cost, unrolled, and two-cycle structures are proposed. In the low-cost structure, the multiplication by $\boldsymbol{M}'$ and the RC_i-add and K_1-add implementations are shared among all the rounds. It is implemented to minimize the area consumption. Experimental results show that compared with existing structures, the new structures are resource-efficient.

2 Description of the PRINCE block cipher

PRINCE is a lightweight block cipher based on the substitution-permutation network. In this cipher, a 64-bit input plaintext P is transformed into a 64-bit output ciphertext C using a 128-bit master key K. Fig. 1 shows the main encryption of PRINCE. It has 12 rounds called $\mathrm{PRINCE}_{\mathrm{core}}$. Each round of $\mathrm{PRINCE}_{\mathrm{core}}$ consists of a key addition, a non-linear layer, a linear layer, and a round constant addition. Other notations used in this paper are shown Table 1.

2.1 Key schedule

The process of the key schedule is as follows:

1. Divide the 128-bit primary key K of PRINCE into two 64-bit subkeys K_0 and K_1, where $K=K_0||K_1$.

2. Expand the 128-bit primary key to 192-bit

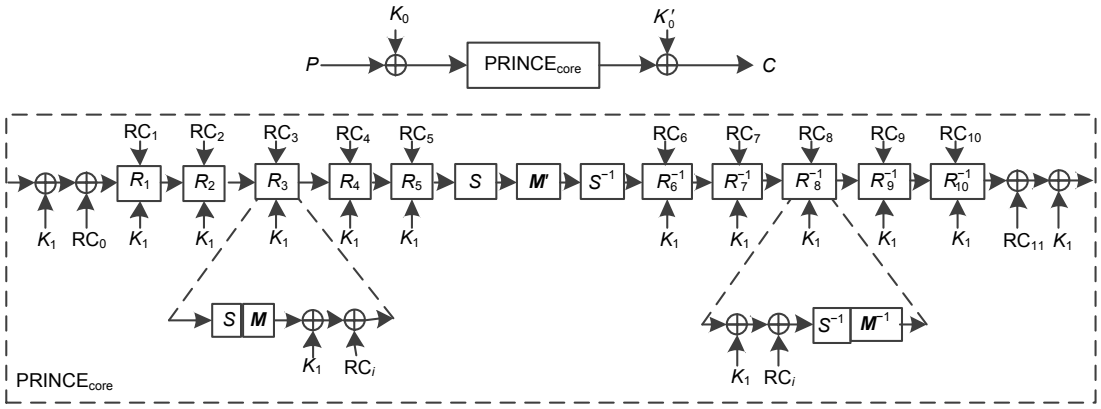

Fig. 1 Encryption structure of PRINCE

Table 1 Notations used in this paper

Notation	Description
K_0, K_0' (K_1)	64-bit whitening keys (round key)
RC$_i$	64-bit round constant in the i^{th} round
SR	Nibble-based shiftrow
S/S^{-1}	4×4 S-box/inverse S-box
$\boldsymbol{M}/\boldsymbol{M}'/\boldsymbol{M}^{-1}$	MixColumns
R_i, R_i^{-1}	Round functions
RC$_i$-add	Round constant addition in the i^{th} round
K_1-add	Round key addition
⊕, XOR	Bitwise exclusive-OR operations
>>>	Cyclic right shift
>>	Right shift

data through a simple mapping: $(K_0||K_1)\rightarrow(K_0||K_0'||K_1)$, where $K_0'=(K_0>>>1)\oplus(K_0>>63)$.

Two 64-bit subkeys K_0 and K_0' are used as the whitening key for the first and last rounds, respectively. The 64-bit subkey K_1 is used for the round key addition of PRINCE$_{core}$.

2.2 S-box layer

The S-box layer is the only non-linear transformation component of PRINCE, and it uses 4-bit S-boxes, i.e., 4-bit input and 4-bit output. The S-boxes used for obtaining the replacement result are given in Table 2 in hexadecimal notations.

2.3 Linear layer

The linear layer of PRINCE is the multiplication by 64×64 matrices: $\boldsymbol{M}$, $\boldsymbol{M}'$, and $\boldsymbol{M}^{-1}$. $\boldsymbol{M}'$ is an involutive matrix, defined as 16 nibbles, m_0, m_1, …, m_{15}. The nibble-based shiftrow (SR) operation is (m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7, m_8, m_9, m_{10}, m_{11}, m_{12}, m_{13}, m_{14}, m_{15})→(m_0, m_5, m_{10}, m_{15}, m_4, m_9, m_{14}, m_3, m_8, m_{13}, m_2, m_7, m_{12}, m_1, m_6, m_{11}). In the $\boldsymbol{M}$-layer, $\boldsymbol{M}'$ is combined with an application of SR to ensure that it achieves full diffusion after two rounds. In the $\boldsymbol{M}^{-1}$-layer, $\boldsymbol{M}^{-1}=\boldsymbol{M}'\times\mathrm{SR}^{-1}(I)$, where I is the input of the $\boldsymbol{M}^{-1}$-layer. The construction of $\boldsymbol{M}'$ requires the following four 4×4 matrices:

Table 2 S-boxes of PRINCE

x	$S(x)$	x	$S(x)$	x	$S(x)$	x	$S(x)$
0x0	0xB	0x4	0xA	0x8	0x6	0xC	0xE
0x1	0xF	0x5	0xC	0x9	0x7	0xD	0x5
0x2	0x3	0x6	0x9	0xA	0x8	0xE	0xD
0x3	0x2	0x7	0x1	0xB	0x0	0xF	0x4

$$\boldsymbol{M}_0=\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\quad \boldsymbol{M}_1=\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},$$

$$\boldsymbol{M}_2=\begin{pmatrix}0&0&0&0\\0&1&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix},\quad \boldsymbol{M}_3=\begin{pmatrix}1&0&0&0\\0&0&0&0\\0&0&1&0\\0&0&0&1\end{pmatrix}.$$

The 64×64 diagonal matrix $\boldsymbol{M}'$ is constructed with four 16×16 matrices $\left(\hat{\boldsymbol{M}}^{(0)},\hat{\boldsymbol{M}}^{(1)},\hat{\boldsymbol{M}}^{(1)},\hat{\boldsymbol{M}}^{(0)}\right)$ as diagonal blocks. The choice of the building blocks

$(\boldsymbol{M}_0, \boldsymbol{M}_1, \boldsymbol{M}_2, \boldsymbol{M}_3)$ and the symmetric structure ensures that $\hat{\boldsymbol{M}}^{(0)}$ and $\hat{\boldsymbol{M}}^{(1)}$ are involution matrices defined as follows:

$$\hat{\boldsymbol{M}}^{(0)}=\begin{pmatrix}\boldsymbol{M}_0 & \boldsymbol{M}_1 & \boldsymbol{M}_2 & \boldsymbol{M}_3\\ \boldsymbol{M}_1 & \boldsymbol{M}_2 & \boldsymbol{M}_3 & \boldsymbol{M}_0\\ \boldsymbol{M}_2 & \boldsymbol{M}_3 & \boldsymbol{M}_0 & \boldsymbol{M}_1\\ \boldsymbol{M}_3 & \boldsymbol{M}_0 & \boldsymbol{M}_1 & \boldsymbol{M}_2\end{pmatrix},$$

$$\hat{\boldsymbol{M}}^{(1)}=\begin{pmatrix}\boldsymbol{M}_1 & \boldsymbol{M}_2 & \boldsymbol{M}_3 & \boldsymbol{M}_0\\ \boldsymbol{M}_2 & \boldsymbol{M}_3 & \boldsymbol{M}_0 & \boldsymbol{M}_1\\ \boldsymbol{M}_3 & \boldsymbol{M}_0 & \boldsymbol{M}_1 & \boldsymbol{M}_2\\ \boldsymbol{M}_0 & \boldsymbol{M}_1 & \boldsymbol{M}_2 & \boldsymbol{M}_3\end{pmatrix}.$$

2.4 Round constant addition

In this component, the 64-bit state is XORed with RC_i (i=0, 1, …, 11). For all RC_i's, $\mathrm{RC}_i \oplus \mathrm{RC}_{11-i}=\alpha$, where α is a constant and its value is expressed in hexadecimal as 0xc0ac29b7c97c50dd. RC_0=0, $\mathrm{RC}_{11}=\alpha$, and the values of RC_6, RC_7, RC_8, RC_9, and RC_{10} can be obtained from RC_1, RC_2, RC_3, RC_4, and RC_5, respectively. Details of $\mathrm{RC}_{0\text{-to-}11}$ can be found in Borghoff et al. (2012).

2.5 Existing structures of PRINCE

Existing structures of PRINCE include three types: unrolled, low-cost, and two-cycle. Abbas et al. (2014) studied the unrolled structure and proposed a new structure to improve the performance and flexibility of PRINCE. In this unrolled structure, the intermediate process of PRINCE had been changed from $S \rightarrow \boldsymbol{M}' \rightarrow S^{-1}$ to $\mathrm{SR} \rightarrow \boldsymbol{M}' \rightarrow \mathrm{SR}^{-1}$. Details can be found in Abbas et al. (2014, 2015, 2016).

Rashidi (2020b) proposed two structures of PRINCE: low-cost and two-cycle. The low-cost structure was designed to minimize the amount of computing resources required. This structure required 64 1-bit registers, eight 2-to-1 multiplexers, one 5-to-1 multiplexer, 16 S-boxes, 16 inverse S-boxes, six 64-bit XOR gates, three multiplications ($\boldsymbol{M}'$, $\boldsymbol{M}$, $\boldsymbol{M}^{-1}$), and five 32-bit NOT gates. For latency, this structure had 11 cycles. After the plaintext was XORed with K_0 and K_1 in the first cycle, the process of $S \rightarrow \boldsymbol{M} \rightarrow K_1 \rightarrow \mathrm{RC}_i$ was executed in the first five cycles. The $S \rightarrow \boldsymbol{M} \rightarrow S^{-1} \rightarrow K_1 \rightarrow \mathrm{RC}_6$ operation was completed in the 6th cycle, and the process of $\boldsymbol{M}^{-1} \rightarrow S^{-1} \rightarrow K_1 \rightarrow \mathrm{RC}_i$ was performed in the last five cycles. Details can be found in Rashidi (2020b).

In the two-cycle structure of PRINCE, a processing element called PE was introduced. In PE, $S \rightarrow \boldsymbol{M} \rightarrow K_1 \rightarrow \mathrm{RC}_i$ and $\mathrm{RC}_i \rightarrow K_1 \rightarrow \boldsymbol{M}^{-1} \rightarrow S^{-1}$ were integrated, and they shared round key addition and round constant addition. This PE required 16 S-boxes, 16 inverse S-boxes, two 64-bit XOR gates, one multiplication of $\boldsymbol{M}$, one multiplication of $\boldsymbol{M}^{-1}$, and one 2-to-1 multiplexer. This two-cycle structure required 64 1-bit registers, five 2-to-1 multiplexers, five PEs, six 64-bit XOR gates, one multiplication of $\boldsymbol{M}'$, and five 32-bit NOT gates. For latency, this structure had only two cycles. Details can be found in Rashidi (2020b).

3 Structural optimization of the PRINCE cipher

With the goal of minimizing the area, we first study a specific implementation of the PRINCE components and propose new implementation methods. Based on these new implementation methods, three structures of PRINCE (unrolled, low-cost, and two-cycle) are constructed.

3.1 Unrolled structure

Without changing the encryption process of PRINCE, we propose a new unrolled structure. The data flow of this new structure is shown in Fig. 2. Compared with previous unrolled structures, the

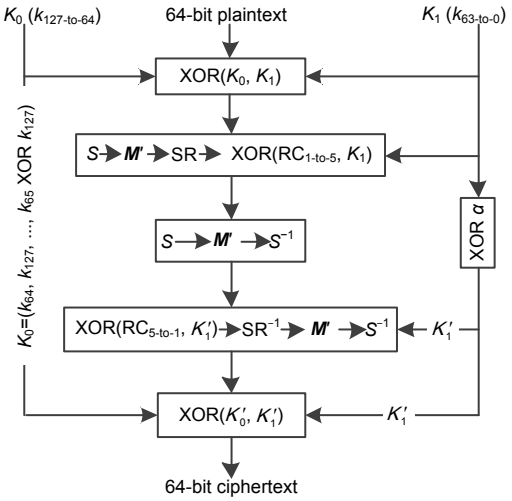

Fig. 2 Data flow of the new unrolled structure of PRINCE

encryption of this structure has only $RC_{1\text{-to-}5}$-add operations and no RC_0-add or $RC_{6\text{-to-}11}$-add operation. To ensure the correctness of PRINCE encryption result, a new variable K_1' has been introduced which replaces K_1 used after the intermediate process. The $\boldsymbol{M'}$-layer is one of the components of the PRINCE that consume a large number of logic gates. The nonlinear components of the unrolled structure are realized in the form of look-up tables (LUTs). Detailed implementation of other components is described below.

3.1.1 New structure of the multiplication by $\boldsymbol{M'}$

In the previous implementation structure of multiplication by $\boldsymbol{M'}$, every two-bit output requires four XOR operations. For example, $Mp_0=i_0\oplus i_4\oplus i_8$ and $Mp_4=i_0\oplus i_4\oplus i_{12}$. However, every two-bit output has the same XOR operation, such as Mp_0 and Mp_4, and both contain $i_0\oplus i_4$. We can first calculate the result of $i_0\oplus i_4$, and then calculate Mp_0 and Mp_4. Only three XOR gates are required in this method (Fig. 3).

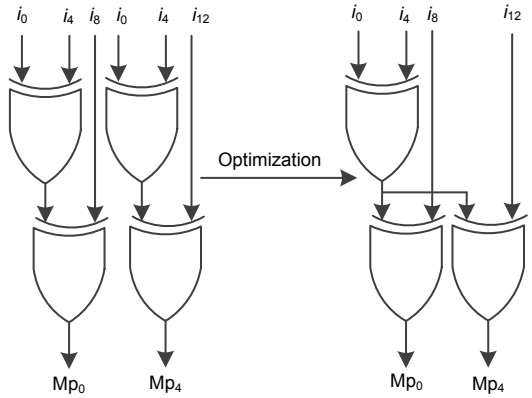

Fig. 3 Implementation structure of Mp_0 and Mp_4

For the structure of multiplication by $\boldsymbol{M'}$, there are 32 identical XOR operations. The structure of multiplication by $\boldsymbol{M'}$ can be optimized according to the optimization method of Mp_0 and Mp_4. First, 32 identical XOR operations are performed, and then the XOR operations required for each bit are performed individually. The number of XOR logic gates required in this method is reduced by 32. Fig. 4 shows the new implementation structure of multiplication by $\boldsymbol{M'}$.

3.1.2 Implementations of RC_i-add and K_1-add

For the implementation of RC_i-add, the earliest method is to store 12 round constants directly and to perform 64-bit round constant XOR operations. To reduce memory usage, researchers proposed to store only the constant values of $RC_{0\text{-to-}5}$ based on the relationship between the round constants: $RC_i\oplus RC_{11-i}=\alpha$. Assuming that the intermediate result of the PRINCE algorithm encryption is defined as "state," the addition of RC_i and K_1 in this method can be expressed using the following formula:

$$\text{state}=\begin{cases}\text{state}\oplus K_1\oplus \text{RC}_i, & i=0,1,...,5,\\ \text{state}\oplus K_1\oplus \text{RC}_{11-i}\oplus\alpha, & i=6,7,...,11.\end{cases}$$

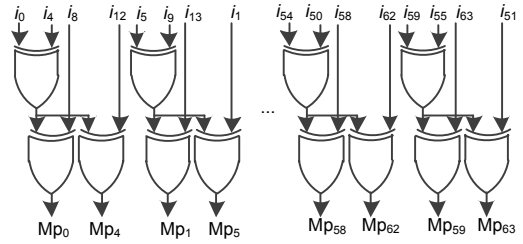

Fig. 4 Newly proposed structure of multiplication by $\boldsymbol{M'}$

Although memory usage is reduced, five 64-bit XOR operations are required to obtain $RC_{6\text{-to-}10}$, which increases the resource consumption. Later, the characteristics of $x\oplus 0=x$ and $x\oplus 1=\sim x$ are used, and an implementation method is proposed using five 32-bit NOT gates to obtain $RC_{6\text{-to-}10}$. To further reduce the area consumption, we introduce the variable $K_1'=K_1\oplus a$, which can be implemented based on the NOT gates proposed by Rashidi (2020b) (Fig. 5). The new implementations of RC_i-add and K_1-add can be expressed using the following formula:

$$\text{state}=\begin{cases}\text{state}\oplus K_1, & i=0,\\ \text{state}\oplus K_1'\oplus \text{RC}_i, & i=1,2,...,5,\\ \text{state}\oplus K_1'\oplus \text{RC}_{11-i}, & i=6,7,...,10,\\ \text{state}\oplus K_1', & i=11.\end{cases}$$

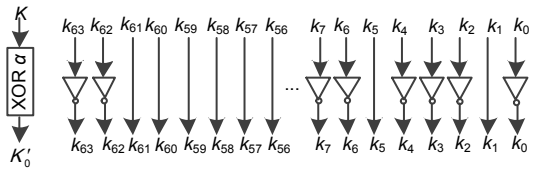

Fig. 5 Implementation structure of K_1'

3.2 Low-cost structure

The S-box (inverse S-box) layer is shared among rounds 1 to 5 (6 to 11) and the intermediate step in

previous low-cost structures. However, in PRINCE, the linear layer is also one of the components that consume a lot of resources. Therefore, we propose a new low-cost structure based on the lowest consumption of computing resources (Fig. 6). The $\boldsymbol{M}'$-layer can be shared among all the rounds (including the intermediate step). This new structure requires only 64 1-bit registers, four 2-to-1 multiplexers, one 5-to-1 multiplexer, 16 S-boxes, 16 inverse S-boxes, five 64-bit XOR gates, one 1-bit XOR gate, one multiplication of $\boldsymbol{M}'$, and one 32-bit NOT gate.

The calculation sequence of the new low-cost structure of the PRINCE cipher is shown in Fig. 7. We optimize the control logics of the PRINCE cipher. Table 3 shows the control signals used in the newly proposed structure to calculate the ciphertext in each clock cycle. In the first clock cycle, the control signals S_0, S_1, and S_2 are equal to 0, 0, and 1, respectively. In this clock cycle, $P \oplus K_0 \oplus K_1$ is applied to the structure, and the calculation of the first round is performed and the result is stored in the registers. The calculation of rounds 2 to 5 is completed in the next four clock cycles. The control signals S_0 and S_1 are set to 0 in rounds 2 to 5. In the 6th clock cycle, the calculation of the intermediate step is performed (with the control signals S_0 and S_1 set to 1). Rounds 7 to 10 are performed in the 7th to 10th clock cycles, and the control signals S_0 and S_1 are equal to 1 and 0, respectively. In the 11th round, the control signals S_0 and S_1 are set to 1 and 0, respectively. Fig. 8 shows the timing diagram of the newly proposed low-cost structure of the PRINCE cipher.

3.3 Two-cycle structure

The multiplication of $\boldsymbol{M}'$ is not shared in the $\boldsymbol{M}'/\boldsymbol{M}/\boldsymbol{M}^{-1}$-layers in existing two-cycle structures. To share K_1-add, RC_i-add, and the multiplication of $\boldsymbol{M}'$, we propose a new processing element named NPE (Fig. 9). However, we find in the subsequent hardware implementation that although the components

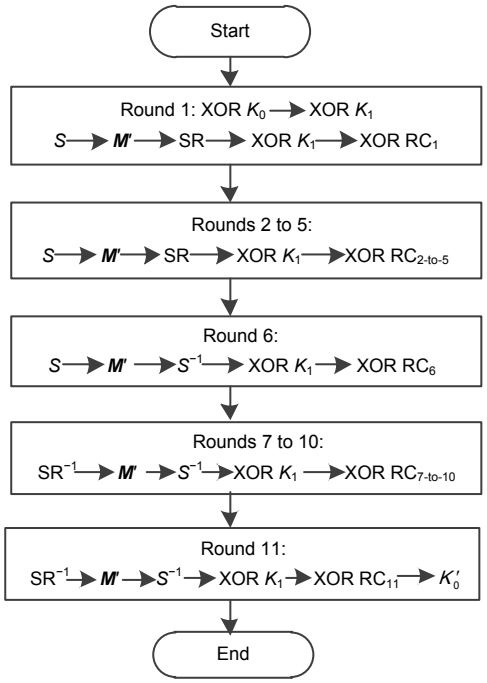

Fig. 7 Operation sequence of the new low-cost structure of PRINCE

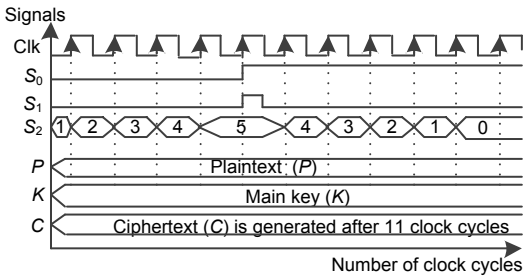

Fig. 8 Timing diagram of the newly proposed low-cost structure of the PRINCE cipher

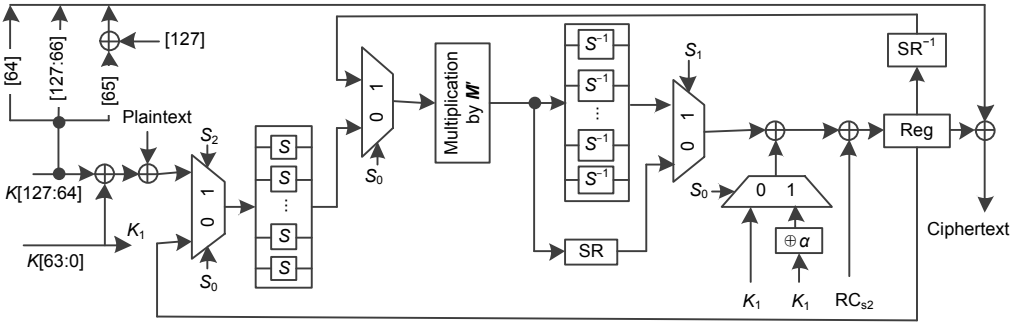

Fig. 6 New low-cost structure of PRINCE

of the PRINCE cipher have been optimally shared, due to the increase in the number of control signals, the resource required for FPGA implementation of this structure is similar to that of existing two-cycle structures. Therefore, we propose a new two-cycle structure of the PRINCE cipher based on PE (Rashidi, 2020b) (Fig. 10). This structure requires 64 1-bit registers, 10 2-to-1 multiplexers, one 32-bit NOT gates, 14 64-bit XOR gates, one 1-bit XOR gate, 96 S-boxes, 96 inverse S-boxes, and 11 multiplications of $\boldsymbol{M}'$ in $\boldsymbol{M}'/\boldsymbol{M}/\boldsymbol{M}^{-1}$-layers. Compared with previous two-cycle structures, this structure reduces the number of 32-bit NOT gates by 4 and the number of 64-bit XOR gates by 2, and the implementation of multiplication by $\boldsymbol{M}'$ in this structure reduces the 32-bit XOR gates.

Table 3 Control signals used in the new low-cost structure to calculate the ciphertext in each clock cycle

Clock cycle	S_0	S_1	S_2	RC_{s2}	K_1 or K_1'
1	0	0	1	RC_1	K_1
2	0	0	2	RC_2	K_1
3	0	0	3	RC_3	K_1
4	0	0	4	RC_4	K_1
5	0	0	5	RC_5	K_1
6	1	1	5	RC_5	K_1'
7	1	0	4	RC_4	K_1'
8	1	0	3	RC_3	K_1'
9	1	0	2	RC_2	K_1'
10	1	0	1	RC_1	K_1'
11	1	0	0		K_1'

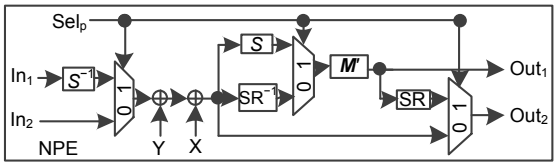

Fig. 9 New processing element (NPE) in the two-cycle structure of PRINCE

4 Hardware implementation and comparison

We have successfully verified and synthesized the new structures using Xilinx ISE 14.2 on Virtex-4 (XC4VFX12-10FF668), Virtex-5 (XC5VLX 50T-1FF1156), Virtex-6 (XC6VLX240T-2FF784), Spartan-6 (XC6SlX45T-3FGG484), and Kintex-7 (XC7K70T-3FBG484) FPGAs. The simulation results are shown in Fig. 11. When the input plaintext is 0x0000000000000000 and the input master key is 0x00000000000000000000000000000000, the value of the ciphertext is 0x818665aa0d02dfda. The results are the same as the test values of the first set in Borghoff et al. (2012), so the newly proposed structures and the optimization ideas are correct.

The new structures were synthesized on different FPGAs and evaluated by the area (number of flip flops, number of LUTs, and number of slices), maximum frequency, throughput, and efficiency indicators. Table 4 shows the results of evaluation of the

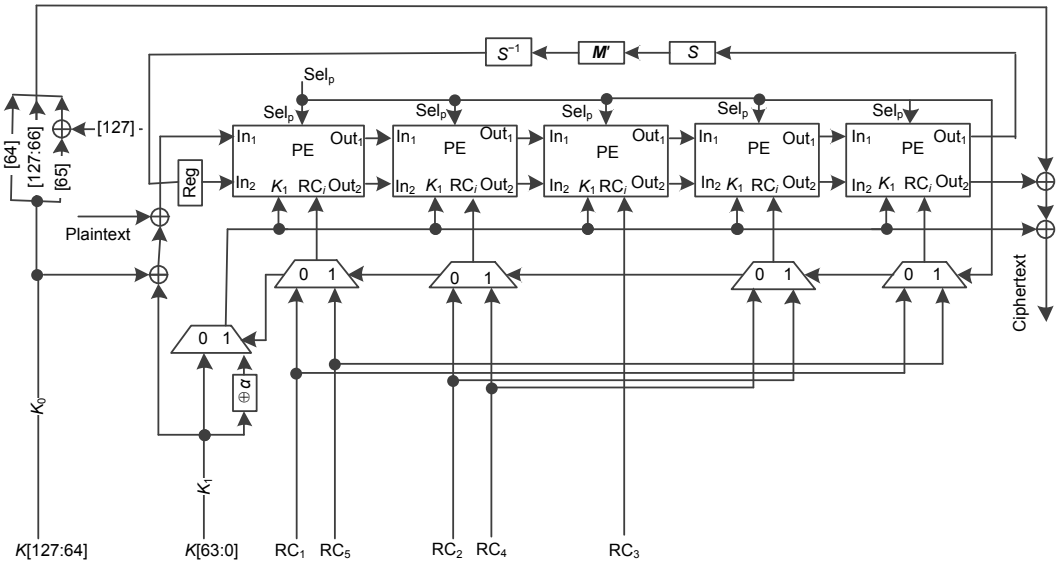

Fig. 10 New two-cycle structure of PRINCE

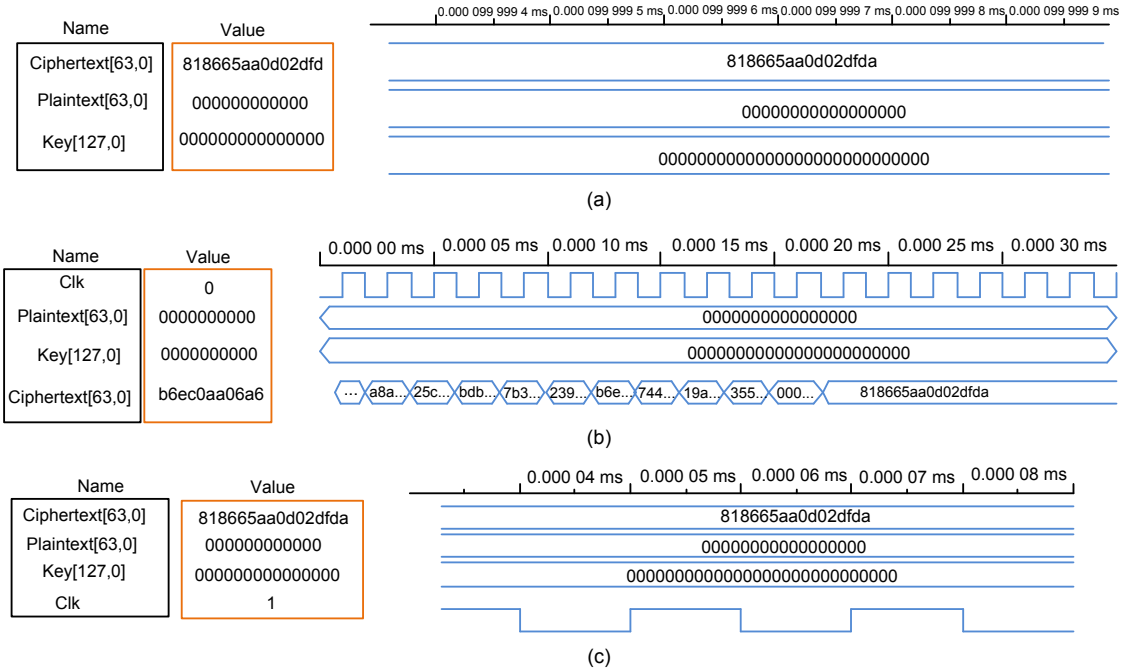

Fig. 11 Simulation results of the three proposed structures: (a) unrolled; (b) low-cost; (c) two-cycle

Table 4 Comparison of the new structures with existing structures

Structure	Device	Area			Maximum frequency (MHz)	Throughput**** (Mb/s)	Efficiency***** (Mb/s)
		Flip flop*	LUT**	Slice***			
Unrolled (Abbas et al., 2014)	Virtex-4			956	31.76	2032.64	2.13
Unrolled (this work)			1840	941	31.69	2028.16	2.16
Low-cost (Rashidi, 2020b)				387	357.78	2081.63	5.38
Low-cost (this work)		75	644	334	147.53	858.36	2.60
Two-cycle (Rashidi, 2020b)				1305	136.04	4353.15	3.34
Two-cycle (this work)		65	1920	1018	60.70	1942.40	1.91
Unrolled (this work)	Virtex-5		1489	576	32.44	2076.16	3.60
Low-cost (this work)		72	431	187	178.88	1040.76	5.57
Two-cycle (this work)		65	1634	623	115.54	3697.28	5.93
Unrolled (Abbas et al., 2014)	Virtex-6			482	65.38	4184.32	8.68
Unrolled (Maene and Verbauwhede, 2015)			1244		61.04	3902.72	3.14
Unrolled (this work)			1153	409	83.45	5340.80	13.06
Low-cost (Rashidi, 2020b)				256	465.12	2706.13	10.57
Low-cost (this work)		73	330	137	304.18	1769.77	12.92
Two-cycle (Rashidi, 2020b)				831	172.27	5512.49	6.63
Two-cycle (this work)		69	1279	451	138.63	4436.16	9.84
Unrolled (this work)	Spartan-6		1153	485	38.10	2438.40	5.03
Low-cost (this work)		77	333	127	145.10	844.22	6.65
Two-cycle (this work)		74	1236	433	68.86	2203.52	5.09
Unrolled (this work)	Kintex-7		1153	646	102.54	6562.56	10.16
Low-cost (this work)		73	330	162	374.49	2178.85	13.45
Two-cycle (this work)		65	1275	432	172.71	5526.72	12.79

* Number of flip flops; ** number of LUTs; ***: number of slices; **** throughput=(block size×maximum frequency)/number of clock cycles; ***** efficiency=throughput/number of slices

new and previous structures. For resource consumption, since the low-cost structure requires more signals, the number of flip flops was slightly larger than that in the two-cycle structure, but the numbers of slices and LUTs of the low-cost structure were much smaller than those of the unrolled and two-cycle structures. The throughput of the unrolled and two-cycle structures was higher than that of the low-cost structure. The throughput of the two-cycle was about 35% higher than that of the unrolled structure on Kintex-7, Virtex-6, and Virtex-5. The efficiency of the low-cost structure was higher than that of the unrolled and two-cycle structures. When the efficiency was used as a metric of the trade-off between area and throughput, the low-cost structure was shown to be the best choice. We can select the most suitable architectures according to the performance indicators required by different application scenarios.

In a previous study of the unrolled structure, Abbas et al. (2016) implemented the hardware IP-core of PRINCE with high speed and low resource used on the Virtex-4 FPGA. The FPGA implementation results showed that the structure required 956 slices, the execution time was 31.48 ns, and the throughput was 2.03 Gb/s. In Abbas et al. (2014), the unrolled structure was also implemented on Virtex-6. The maximum throughput of this structure was 4.18 Gb/s, the execution time was 15.29 ns, and the consumption area was 482 slices. Maene and Verbauwhede (2015) implemented the unrolled structure on Virtex-6. Their results showed that it required 1244 LUTs, the execution time was 16.4 ns, and the throughput was 3.64 Gb/s. The new unrolled structure was implemented on Virtex-6, which required 1153 LUTs and 409 slices. Its maximum throughput was 5.34 Gb/s and the execution time was 11.98 ns. On Virtex-4, 1840 LUTs and 941 slices were required, the maximum throughput was 2.03 Gb/s, and the execution time was 31.56 ns. Compared with the previous unrolled structure, the new unrolled structure required a smaller area and had higher throughput. The low-cost architecture proposed by Rashidi (2020b) needed 387 slices on Virtex-4 and 256 slices on Virtex-6, whereas our structure needed only 334 and 137 slices, respectively. For the two-cycle structure, the previous structure required 1305 slices on Virtex-4 and 831 slices on Virtex-6, while our structure needed only 1151 and 466 slices, respectively.

A comparison of the area, maximum frequency, throughput, and efficiency of FPGA (Virtex-6) for the new structures of the PRINCE cipher with those of other existing structures is shown in Fig. 12. The results showed that the new unrolled structure was low-cost and that the three new structures were better than previous structures in terms of area and efficiency.

Compared with the unrolled structure of other ciphers, the number of LUTs required by the unrolled architecture was much smaller than those of the PRINCE, PRESENT, RECTANGLE, KATAN, SIMON, and SPECK ciphers. Tables 5 and 6 show the hardware implementation results of the new structures and other ciphers. Compared with other ciphers, the low-cost structure had acceptable performance in terms of resource consumption, and its throughput was higher. Fig. 13 compares the throughput and efficiency of the low-cost structure with those of other ciphers on Spartan-6.

Table 5 Comparison of the unrolled structure and other ciphers

Structure	Device	Area		Maximum frequency (MHz)	Throughput*** (Mb/s)	Efficiency**** (Mb/s)
		LUT*	Slice**			
SIMON (Maene and Verbauwhede, 2015)	Virtex-4	2688		36.63	2344.32	
SPECK (Maene and Verbauwhede, 2015)		3594		19.88	1272.32	
PRESENT (Maene and Verbauwhede, 2015)		2089		30.67	1962.88	
RECTANGLE (Maene and Verbauwhede, 2015)		1668		38.31	2451.84	
AES (Maene and Verbauwhede, 2015)		8984		40.49	5182.72	
Unrolled (this work)		1153	409	83.45	5340.80	13.6

* Number of flip flops; ** number of slices; *** throughput=(block size×maximum frequency)/number of clock cycles; **** efficiency= throughput/number of slices

Table 6 Comparison of the low-cost structure with other ciphers

Structure	Device	Area			Maximum frequency (MHz)	Throughput**** (Mb/s)	Efficiency***** (Mb/s)
		Flip flop*	LUT**	Slice***			
ANU (Dahiphale et al, 2020)	Virtex-5	199	270	103	559.57	1432.48	13.91
RECTANGLE (Dahiphale et al., 2019)		199	395	149	529.47	1303.30	8.75
PRESENT (Rashidi, 2020a)		149	230	68	397.60	848.21	12.47
Low-cost (this work)		70	437	185	226.38	1317.12	7.12
LED (Rashidi, 2020a)	Kintex-7	70	273	122	485.79	971.58	7.96
Low-cost (this work)		73	330	162	374.49	2178.85	13.45
RECTANGLE (Dahiphale et al., 2019)	Spartan-6	199	220	73	277.03	681.92	9.34
Midori (Lara-Nino et al., 2018)		200	356	118	166.17	259.38	2.20
GIFT (Lara-Nino et al., 2018)		205	189	58	218.10	268.43	4.63
Lilliput (Singh et al., 2019)		149	198	58	237.07	505.74	8.71
Low-cost (this work)		77	333	127	145.10	844.22	6.65

* Number of flip flops; **: number of LUTs; *** number of slices; **** throughput=(block size×maximum frequency)/number of clock cycles; ***** efficiency=throughput/number of slices

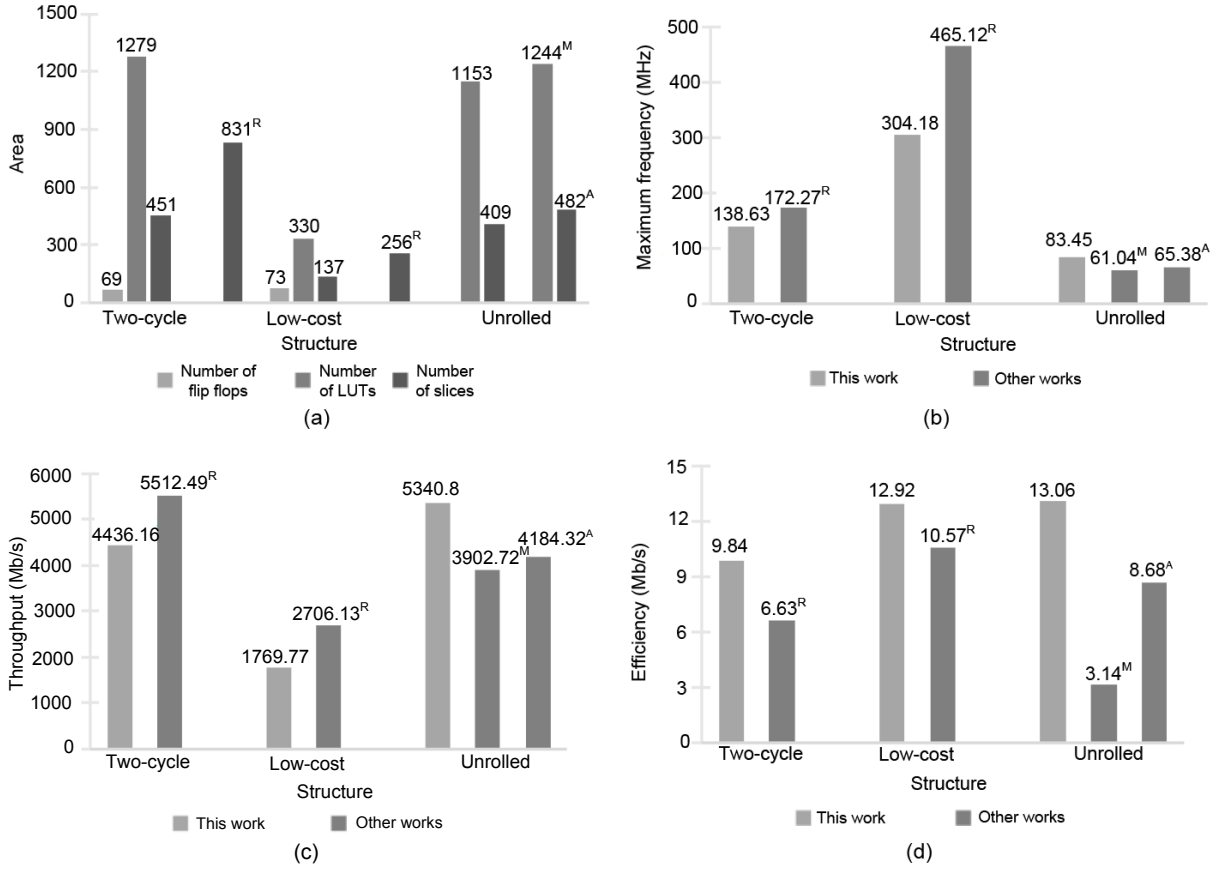

Fig. 12 Comparison of the new and existing structures in terms of area (a), maximum frequency (b), throughput (c), and efficiency (throughput/number of slices) (d)

The superscripts R, A, and M mean that the results are obtained from Rashidi (2020b), Abbas et al. (2014), and Maene and Verbauwhede (2015), respectively

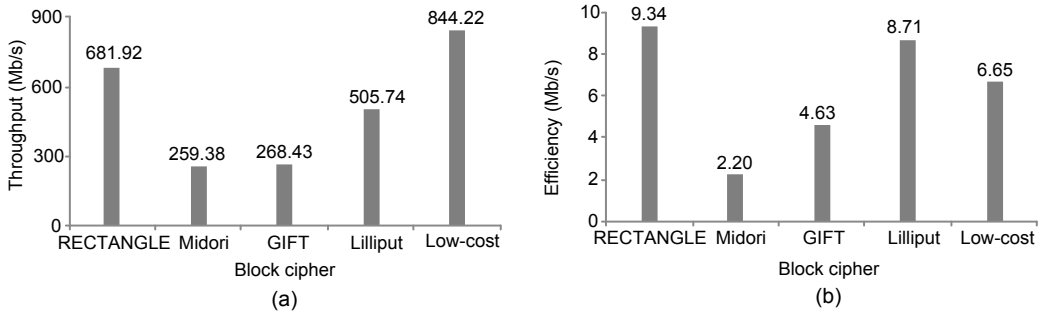

Fig. 13 Comparison of throughput (a) and efficiency (throughput/number of slices) (b) of this work (low-cost structure) and other ciphers on Spartan-6

5 Conclusions

In the era of pervasive computing, security and privacy are challenges in many data-sensitive applications. As low-resource devices are limited in terms of computing resources, the problem becomes more complicated when processing data. In this study, new unrolled, low-cost, and two-cycle hardware structures are presented. The unrolled structure includes new hardware structures for the multiplication of $\boldsymbol{M}'$ and the K_1-add and RC_i-add implementations. In the new low-cost structure, we propose a new implementation method, with the multiplication of $\boldsymbol{M}'$ shared in all rounds. Based on the optimization ideas of the unrolled and low-cost structures, we propose a new two-cycle structure. Compared with existing two-cycle structures, it is further optimized in the realization of multiplication of $\boldsymbol{M}'$ and K_1-add and RC_i-add implementations. Compared with existing structures, the newly proposed architectures are more resource-efficient and suitable for lightweight, latency-critical applications.

Contributors

Lang LI and Jingya FENG designed the research. Lang LI, Jingya FENG, and Botao LIU conducted the experiments and drafted the manuscript. Ying GUO and Qiuping LI helped organize the manuscript. Lang LI revised and finalized the paper.

Compliance with ethics guidelines

Lang LI, Jingya FENG, Botao LIU, Ying GUO, and Qiuping LI declare that they have no conflict of interest.

References

Abbas YA, Jidin R, Jamil N, et al., 2014. Implementation of PRINCE algorithm in FPGA. Proc 6th Int Conf on Information Technology and Multimedia, p.1-4. https://doi.org/10.1109/ICIMU.2014.7066593

Abbas YA, Jidin R, Jamil N, et al., 2015. PRINCE IP-core on field programmable gate arrays (FPGA). *Res J Appl Sci Eng Technol*, 10(8):914-922. https://doi.org/10.19026/rjaset.10.2447

Abbas YA, Jidin R, Jamil N, et al., 2016. Lightweight PRINCE algorithm IP core for securing GSM messaging using FPGA. *Res J Inform Technol*, 8(1):17-28. https://doi.org/10.3923/rjit.2016.17.28

Abed S, Jaffal R, Mohd B, et al., 2019. FPGA modeling and optimization of a SIMON lightweight block cipher. *Sensors*, 19(4):913. https://doi.org/10.3390/s19040913

Alioto M, 2017. Enabling the Internet of Things: from Integrated Circuits to Integrated Systems. Springer, Switzerland. https://doi.org/10.1007/978-3-319-51482-6

Ayesha N, Singh P, Acharya B, 2020. An area-optimized architecture for LiCi cipher. Int Conf on Nanoelectronics, Circuits and Communication Systems, p.157-167. https://doi.org/10.1007/978-981-15-2854-5_16

Balderas-Contreras T, Cumplido R, Feregrino-Uribe C, 2008. On the design and implementation of a RISC processor extension for the KASUMI encryption algorithm. *Comput Electr Eng*, 34(6):531-546. https://doi.org/10.1016/j.compeleceng.2007.11.003

Bansod G, Pisharoty N, Patil A, 2017. BORON: an ultra-lightweight and low power encryption design for pervasive computing. *Front Inform Technol Electron Eng*, 18(3):317-331. https://doi.org/10.1631/FITEE.1500415

Beaulieu R, Shors D, Smith J, et al., 2015. The SIMON and SPECK lightweight block ciphers. Proc 52nd Annual Design Automation Conf, p.1-6. https://doi.org/10.1145/2744769.2747946

Bogdanov A, Knudsen LR, Leander G, et al., 2007. PRESENT: an ultra-lightweight block cipher. Proc 9th Int Workshop on Cryptographic Hardware and Embedded Systems, p.450-466. https://doi.org/10.1007/978-3-540-74735-2_31

Borghoff J, Canteaut A, Güneysu T, et al., 2012. PRINCE—a low-latency block cipher for pervasive computing applications. Proc 18th Int Conf on the Theory and Application

of Cryptology and Information Security, p.208-225. https://doi.org/10.1007/978-3-642-34961-4_14

Dahiphale V, Raut H, Bansod G, 2019. Design and implementation of novel datapath designs of lightweight cipher RECTANGLE for resource constrained environment. *Multim Tools Appl*, 78(16):23659-23688. https://doi.org/10.1007/s11042-019-7587-3

Dahiphale V, Bansod G, Zambare A, et al., 2020. Design and implementation of various datapath architectures for the ANU lightweight cipher on an FPGA. *Front Inform Technol Electron Eng*, 21(4):615-628. https://doi.org/10.1631/FITEE.1800681

Dhanda S, Singh B, Jindal P, 2020. Lightweight cryptography: a solution to secure IoT. *Wirel Pers Commun*, 112(3):947-1980. https://doi.org/10.1007/s11277-020-07134-3

Feizi S, Nemati A, Ahmadi A, et al., 2015. A high-speed FPGA implementation of a bit-slice ultra-lightweight block cipher, RECTANGLE. Proc 5th Int Conf on Computer and Knowledge Engineering, p.206-211. https://doi.org/10.1109/ICCKE.2015.7365828

Kitsos P, Sklavos N, Parousi M, et al., 2012. A comparative study of hardware architectures for lightweight block ciphers. *Comput Electr Eng*, 38(1):148-160. https://doi.org/10.1016/j.compeleceng.2011.11.022

Lara-Nino CA, Díaz-Pérez A, Morales-Sandoval M, 2018. FPGA-based assessment of Midori and GIFT lightweight block ciphers. Proc 20th Int Conf on Information and Communications Security, p.745-755. https://doi.org/10.1007/978-3-030-01950-1_45

Liu BT, Li L, Wu RX, et al., 2019. Loong: a family of involutional lightweight block cipher based on SPN structure. *IEEE Access*, 7:36023-136035. https://doi.org/10.1109/ACCESS.2019.2940330

Maene P, Verbauwhede I, 2015. Single-cycle implementations of block ciphers. Proc 4th Int Workshop on Lightweight Cryptography for Security and Privacy, p.131-147. https://doi.org/10.1007/978-3-319-29078-2_8

Philip MA, Vaithiyanathan AV, 2017. A survey on lightweight ciphers for IoT devices. Int Conf on Technological Advancements in Power and Energy, p.1-4. https://doi.org/10.1109/TAPENERGY.2017.8397271

Rashidi B, 2019. High-throughput and flexible ASIC implementations of SIMON and SPECK lightweight block ciphers. *Int J Circ Theor Appl*, 47(8):1254-1268. https://doi.org/10.1002/cta.2645

Rashidi B, 2020a. Flexible structures of lightweight block ciphers PRESENT, SIMON and LED. *IET Circ Dev Syst*, 14(3):369-380. https://doi.org/10.1049/iet-cds.2019.0363

Rashidi B, 2020b. Low-cost and two-cycle hardware structures of PRINCE lightweight block cipher. *Int J Circ Theory Appl*, 48(8):1227-1243. https://doi.org/10.1002/cta.2832

Shrivastava N, Singh P, Acharya B, 2020. A novel hardware architecture for rectangle block cipher. Int Conf on Nanoelectronics, Circuits and Communication Systems, p.169-181. https://doi.org/10.1007/978-981-15-2854-5_17

Singh P, Acharya B, Chaurasiya RK, 2019. Efficient VLSI architectures of LILLIPUT block cipher for resource-constrained RFID devices. IEEE Int Conf on Electronics, Computing and Communication Technologies, p.1-6. https://doi.org/10.1109/CONECCT47791.2019.9012869

Wu WL, Zhang L, 2011. LBlock: a lightweight block cipher. Proc 9th Int Conf on Applied Cryptography and Network Security, p.327-344. https://doi.org/10.1007/978-3-642-21554-4_19

Xiong JB, Ren J, Chen L, et al., 2019. Enhancing privacy and availability for data clustering in intelligent electrical service of IoT. *IEEE Int Things J*, 6(2):530-1540. https://doi.org/10.1109/JIOT.2018.2842773

Xu T, Wendt JB, Potkonjak M, 2014. Security of IoT systems: design challenges and opportunities. IEEE/ACM Int Conf on Computer-Aided Design, p.417-423.

Yang GQ, Zhu B, Suder V, et al., 2015. The Simeck family of lightweight block ciphers. Int Workshop on Cryptographic Hardware and Embedded Systems, p.307-329. https://doi.org/10.1007/978-3-662-48324-4_16

Zhang WT, Bao ZZ, Lin DD, et al., 2015. RECTANGLE: a bit-slice lightweight block cipher suitable for multiple platforms. *Sci China Inform Sci*, 58(12):122103:1-122103:15. https://doi.org/10.1007/s11432-015-5459-7