International Journal of Information Security (2024) 23:3653–3664
https://doi.org/10.1007/s10207-024-00904-1

REGULAR CONTRIBUTION

Securing the IoT ecosystem: ASIC-based hardware realization of Ascon lightweight cipher

Safiullah Khan[1] · Kashif Inayat[2,3] · Fahad Bin Muslim[4] · Yasir Ali Shah[5] · Muhammad Atif Ur Rehman[1] · Ayesha Khalid[6] · Malik Imran[6] · Akmalbek Abdusalomov[7,8]

Published online: 29 August 2024

Abstract

The Internet of Things (IoT) nodes consist of sensors that collect environmental data and then perform data exchange with surrounding nodes and gateways. Cybersecurity attacks pose a threat to the data security that is being transmitted in any IoT network. Cryptographic primitives are widely adopted to address these threats; however, the substantial computation demands limit their applicability in the IoT ecosystem. In addition, each IoT node varies with respect to the area and throughput (TP) requirements, thus demanding flexible implementation for encryption/decryption processes. To solve these issues, this work implements the NIST lightweight cryptography standard, Ascon, on a SAED 32 nm process design kit (PDK) library by employing loop folded, loop unrolled and fully unrolled architectures. The fully unrolled architecture can achieve the highest TP but at the cost of higher area utilisation. Unrolling by a lower factor results in lower area implementations, enabling the exploration of design space to tackle the trade-off between area and TP performance of the design. The implementation results show that, for loop folded architecture, Ascon-128 and Ascon-128a require 36.7k μm^2 and 38.5k μm^2 chip area, respectively compared to 277.1k μm^2 and 306.6k μm^2 required by their fully unrolled implementations. The proposed implementation strategies can adjust the number of rounds to accommodate the varied requirements of IoT ecosystems. An implementation with an open-source 45 nm PDK library is also undertaken for enhanced generalization and reproducibility of the results.

Keywords Internet of Things · Lightweight cryptography · Ascon · ASIC implementations

✉ Safiullah Khan
safiullah.khan@mmu.ac.uk

Kashif Inayat
kashif.inayat@inu.ac.kr

Fahad Bin Muslim
fahad.muslim@giki.edu.pk

Yasir Ali Shah
y.shah@ulster.ac.uk

Muhammad Atif Ur Rehman
m.atif.ur.rehman@mmu.ac.uk

Ayesha Khalid
a.khalid@qub.ac.uk

Malik Imran
m.imran@qub.ac.uk

Akmalbek Abdusalomov
a.abdusalomov@tsue.uz

[1] Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M15 6BX, UK

[2] System on Chips Laboratory, Department of Electronics Engineering, Incheon National University, Incheon 22012, Republic of Korea

[3] Barcelona Supercomputing Center - Centro Nacional de Supercomputación, Barcelona, Spain

[4] Faculty of Computer Science and Engineering, GIK Institute, Swabi 23460, Pakistan

[5] School of Computing, Engineering and Intelligent Systems, Ulster University, Magee Campus, Londonderry, NI BT48 7JL, UK

[6] Centre for Secure Information Technologies (CSIT), Queen's University Belfast, Belfast BT7 1NN, UK

[7] Department of Computer Engineering, Gachon University, Seongnam 461-701, Republic of Korea

[8] Department of Information Systems and Technologies, Tashkent State University of Economics, 100066 Tashkent, Uzbekistan

1 Introduction

The data generated and transferred by the Internet of Things (IoT) nodes must be secure to mitigate data leakages. Several traditional cryptographic primitives can be employed, such as elliptic curve cryptography (ECC) [1] (asymmetric) or advanced encryption standard (AES) [2] (symmetric). However, the IoT nodes are inherently resource-constrained, making it hard to deploy compute-intensive algorithms. Therefore, a lighter version of cryptographic algorithms, the so-called “lightweight” cryptography (LWC), is required to meet such demands. LWC is specially designed to replace traditional algorithms in the IoT ecosystem. The compute-intensive operations have been replaced by simpler operations such as XOR, rotations, shifting and substitutions. With the help of LWC, encryption mechanisms can be established that can secure the information exchange between IoT sensor nodes and gateways while maintaining optimal performance.

To enhance the efficiency further for the LWC, one potential direction frequently pursued is using authenticated encryption (AE) [3]. The AE schemes combine the lightweight data encryption mechanisms with the authentication properties, thus providing data security and integrity simultaneously. This property is important for the IoT applications. Thus, the AE schemes allow data encryption and authentication, protecting the data against theft and unauthorized access. Modern-day AE schemes usually process the associated data as well, thus providing authenticated encryption with associated data (AEAD) [4, 5]. AEAD schemes, when combined with lightweight hash functions, can be employed to develop protocols that resist distributed denial-of-service (DDoS) and replay attacks [6, 7], crucial for the IoT ecosystem. This approach addresses the security and integrity concerns in IoT ecosystems and caters to the specific constraints and limitations of the IoT nodes.

The performance and hardware resource requirements broadly vary across different applications for the IoT sensor nodes deployed in various scenarios. Consider a real-time monitoring scenario in some surveillance applications, e.g. smart cameras demand high throughput (TP) to capture, encrypt, and transmit images at high speeds. On the other hand, many scenarios, such as battery-operated sensor nodes deployed in remote areas for roadside condition monitoring, face limitations regarding hardware area and energy consumption. It is important to design the data encryption hardware in such a way that they have moderate or small area footprints that consume minimal energy while still delivering reasonable TP. The diverse requirements of IoT applications have motivated the exploration of AEAD schemes and the proposal of efficient hardware architectures suited for various use case scenarios in the IoT ecosystem.

Over the last decade, significant emphasis has been placed on the design and research of LWC to implement and perform well in constrained devices. National Institute of Standards and Technology (NIST) initiated an LWC competition to solicit, evaluate and standardize algorithms suitable for constrained environments in 2018 [8]. The competition received 57 submissions from 25 different countries, where each package included algorithm specifications and a portable reference software implementation. Out of these submissions, 32 of the more promising candidates advanced to the second round based on their security properties. The selection was further narrowed down to ten finalists that performed significantly better. After another year of extensive analysis and performance benchmarking, the Ascon family was selected as the winner of NIST lightweight encryption contest [9]. Ascon was also selected earlier in 2019 as the primary choice for lightweight authenticated encryption in the final portfolio of the CAESAR competition [10]. The Ascon cipher has 7 variants, offering a range of functionality (including encryption, authenticated encryption, cryptographic hashing, extendable-output function etc); the finalized NIST standard may not include all of them [8].

1.1 Objective and contributions

Before detailing the aims and contributions of this work, we first highlight the limitations of the existing Ascon hardware accelerators in the upcoming text. Numerous Ascon accelerators have been demonstrated on both application-specific integrated circuits (ASIC) and field-programmable gate array (FPGA) platforms, as described in references [11–19]. Notably, ASIC-specific architectures are rarely exploited in the literature and have been detailed in references [11–13]. At the same time, FPGA-based accelerators are frequently implemented and optimized, as described in references [14–19]. However, many existing accelerators are dedicated to specific applications, highlighting the need for flexible Ascon accelerators to accommodate the diverse requirements of the IoT ecosystem. Additionally, to the best of our knowledge, there is no prior ASIC implementation of Ascon-128a.

The primary objective of this work is to implement a flexible and open sourced Ascon accelerator[1] Our main contributions are as follows:

- *Accelerator architecture of Ascon* We have proposed an efficient Ascon architecture accommodating both Ascon-128 and Ascon-128a variants. Moreover, the presented accelerator implements three different styles: loop folded, unrolled, and fully unrolled. The loop unrolled style is implemented and investigated with different unrolling factors applied to each permutation round.

[1] https://github.com/KashifInayat/ascon-lightweight. supporting both its variants by identifying the number of permutations that can be squeezed within a single clock cycle.

- *Exhaustive result evaluations* Comprehensive implementation details, evaluations, and comparisons to existing Ascon implementations are presented, examining the trade-offs among hardware utilization, power consumption, and execution time (details are given in Sect. 5).
- *Implications* Based on the analysis of the trade-offs among hardware utilization, power consumption, and execution time in our loop-unrolled Ascon accelerator, we have outlined effective strategies for the IoT community to adapt and customize their hardware designs to meet the unique requirements of each application while ensuring both security and efficiency.

The rest of the paper is organized as follows. The existing Ascon hardware accelerators are described in Sect. 2. The details about the internal architecture of Ascon is provided in Sect. 3 followed by the proposed implementation strategies in Sect. 4. The implementation results and comparison are discussed in Sect. 5. Section 6 is dedicated to highlight the applications while Sect. 7 concludes the paper.

2 Existing Ascon accelerators

2.1 ASIC-demonstrated accelerators

The existing ASIC-specific Ascon accelerators are summarized in Table 1. In [11], a low-area (LA) and high-TP (HT) architecture of Ascon is discussed. In S-box design, the area utilization is reduced as one-bit is processed in each clock cycle, at the cost of higher latency in terms of clock cycle utilization. High TP is achieved by performing one round of permutation per clock cycle. Similar techniques are adopted for hardware implementation of Ascon in [12]. Due to low area utilization, the architectures of [12] can be deployed in applications such as wireless sensor nodes, radio-frequency identification network (RFID) tags and embedded systems. The results have been demonstrated when two rounds are executed in each clock cycle in Table 1. To enable fast recovery from unplanned power failures in IoT environments, a CMOS/MRAM-based hardware implementation of Ascon is presented in [13]. This implementation is specifically employed for energy harvesting applications to achieve efficient and reliable functionality.

2.2 FPGA-accelerated architectures

Similar to ASIC accelerators, the FPGA-specific architectures for Ascon are summarized in Table 2. Reference [14] describes several lightweight AE schemes, where authors have targeted the internal bus size of 32-bit to minimize the area utilization. Moreover, the storage of secret keys, initialization vectors (IV), and intermediate results during state updates is performed using a single-port memory. The bit-slice approach is used in the implementation of the S-box, where the S-box is sliced into 64 × 5 partitions. The state value in memory is updated on word basis while basic iterative components are designed by using 64-bit internal data path. The implementation results, including TP and TP/Area ratio (TP/A), for the Spartan-6 FPGA platform are provided.

In [15], Ascon and ACORN, that were selected as the finalists for CAESAR competition are discussed, where a detailed comparison of Ascon is presented across ACRON in terms of area and TP, using the advanced encryption standard-Galois/counter mode (AES-GCM) as a reference. The main approach is to perform one permutation round in each cycle and employ the bit-sliced technique for the S-box implementation. The optimization techniques used in these FPGA implementations closely resemble those discussed in [11, 12] for ASIC implementations. Authors in [16] proposed an implementation strategy based on Ascon's iterative architecture. The authors have tried to synchronize the number of permutations to the clock cycle utilisation so that the number of clock cycles and permutations become equal. Twelve clock cycles are required for the initialization and finalization step, while six clock cycles are needed for the processing of associated data (AD) and the plaintext/ciphertext (PT/CT). A few clock cycles are needed for the rest of the operations including the synchronization of the data. However, the number of clock cycles utilized for such operations is fewer compared to the clock cycles required for the iteration process. The data path employed for the encryption/decryption process is 64-bit, in which the multiplexers are used to choose the correct values being input to the next step. Implementations have been performed for the Spartan-6 FPGA platform. Summarizing their work, the authors have optimized the number of clock cycle utilization to enhance the overall TP. A scalable and efficient implementation of Ascon suitable for diverse applications of the smart cities are presented in [18]. The data encryption for the AI-enabled IoT devices by employing Ascon has been demonstrated in [19].

2.3 Other-related implementations

Instead of ASIC and FPGA platforms, an Ascon implemented on microcontroller platform is discussed in [17], where authors have focused to optimize the permutation function for the RISC-V architecture. The underlying Boolean functions for the S-box layer are improved and efficiently implemented on RISC-V platform. The implementation results demonstrate a notable speed increase of 29% compared to reference implementations.

Table 1 Ascon ASIC-specific demonstrations from the literature

Ref	Library	Chip Area	TP (Mbps)	Energy	Power (μW)
[11] (LA)	90nm UMC	2.57 kGE	384	5706 μJ/byte	15
[11] (HT)		7.08 kGE	5524	33 μJ/byte	43
[12]	90nm UMC	10.61 kGE	8425	27 μJ/byte	72
[13]	28nm FD-SOI	4970.8 μm^2	–	1079.4 pJ	745.8

Table 2 Existing accelerators of Ascon on FPGA platform

Ref	FPGA	LUTs	Freq. (MHz)	TP (Mbps)	TP/A (Mbps/LUT)
[14]	Spartan-6	0.68 k	216.0	60.1	0.260
[15]	Spartan-6	1.64 k	146.1	114.0	0.070
[16]	Spartan-6	1.91 k	174.4	1116.4	0.584
[18]	Spartan-6	2.06 k	206.2	315.2	0.153
[19]	Virtex-7	2.70 k	270.3	721.5	0.125

Table 3 Design parameter length for Ascon-128 and Ascon-128a

Parameter size	Ascon-128	Ascon-128a
Security, Key, Nonce, Tag	128-bit	128-bit
Data processed	64-bit	128-bit
Rate/capacity	64-bit/256-bit	128-bit/192-bit
State	320-bit	320-bit
Rounds (a/b)	12/6	12/8

3 Ascon: lightweight cipher

We detail the Ascon parameters in Sect. 3.1. Following that, we delve into the internal architecture of Ascon for AE in Sect. 3.2. Lastly, we explore the Ascon permutation function in Sect. 3.3.

3.1 Supported parameters

Ascon is based on a duplex mode of operation that is identical to MonkeyDuplex [20]. However, Ascon consists of keyed initialization and finalization functions that are much stronger. To achieve compact hardware implementation, the design parameters (state size, rate, capacity, key length) and operating modes (Ascon-128 and Ascon-128a) are diligently selected. The two operating modes of Ascon and their recommended design parameters length are listed in Table 3. Ascon-128 (primary recommendation) and Ascon-128a (secondary recommendation) are ideal for lightweight use cases in IoT environments.

3.2 Ascon internal architecture

The internal architecture of Ascon AE is shown in Fig. 1. Both operating modes of Ascon are executed on a 320-bit state represented by S. The state update is performed in four stages: initialization; processing of AD; processing of PT/CT; and finalization. In both initialization and finalization stages, the permutation function f is employed 12 times while the same permutation function is employed six times in processing AD and PT/CT. The 320-bit state is split into outer (S_r) and inner parts (S_c), where r and c represents rate and capacity respectively. Table 3 provides the parameter size of r and c for both Ascon operating modes and is also expressed mathematically in Eq. (1). Five registers of 64-bit each are used to model the 320-bit internal state, when realized in hardware. The internal state is accessed starting at the most significant byte (or bit) of a_0 as byte 0 and ends at the least significant byte (or bit) of a_4 as byte 39.

$$S = Sr\|Sc = a0\|a1\|a2\|a3\|a4. \tag{1}$$

3.2.1 Initialization

Three parameters are concatenated to make the input that can act as the initial state of Ascon. These parameters include the secret key S_K, nonce N, and the pre-defined initialisation vector (IV). To maintain the confidentiality requirement, it is mandatory to refresh N for every encryption process. The initial state goes through 12 rounds of permutation f, and then is XORed with the secret key, i.e., S_K. Equations (2) and (3) represent the complete initialization process.

$$S \leftarrow IV\|S_K\|N \tag{2}$$

$$S \leftarrow f^a(S) \oplus (0^{320-S_k}\|S_K) \tag{3}$$

3.2.2 Associated data

Processing AD is only applicable when there is AD present; otherwise, it can be skipped. Ascon utilizes a chunk-based

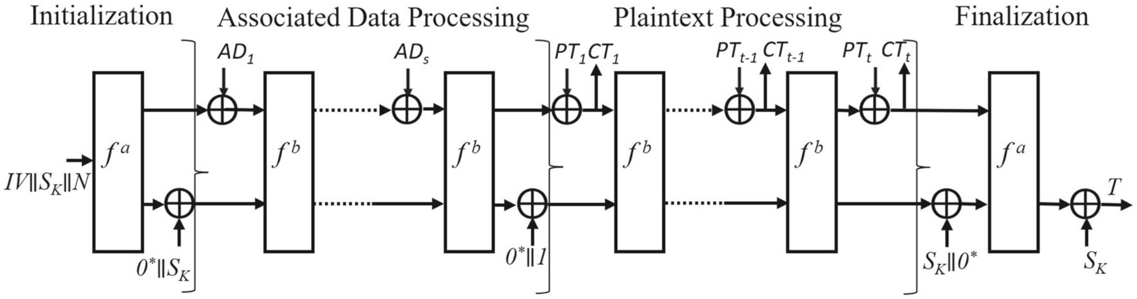

Fig. 1 Internal architecture of Ascon AE encryption. (The decryption process is the same except CT is processed to obtain PT)

processing approach for the AD, where each chunk consists of r bits. To ensure that the total length of the AD is a multiple of the r-bit size, a single 1 is appended along with the minimum number of 0s required to make it a multiple of r, known as padding. The modified AD value is then divided into i chunks of r bits. AD contains information that may not necessarily be confidential but must remain unaltered by any potential attacker. Each block of the AD is subsequently incorporated into the internal state following six permutation rounds. Once all the AD is processed, a single bit 1 is XORed to the least significant bit of the state. The mathematical representation for handling the AD processing is given in Eqs. (4) and (5), respectively.

$$S \leftarrow f^b((S_r) \oplus AD_i)\|S_c) \tag{4}$$

$$S \leftarrow S \oplus (0^{319}\|1) \tag{5}$$

3.2.3 Plaintext/ciphertext processing

The padding is also performed in this step, like the previous step, to get r-bit multiple of PT. The padded plaintext is split into chunks, then encryption is performed on each chunk. Equations (6) and (7) show the encryption process performed. An XOR operation is performed between state r-bit of state S_r and each chunk of PT_i, resulting in one block of cyphertext message CT_i. Permutation function f is applied six times to update the state, followed by processing the next chunk.

$$S_r \leftarrow S_r \oplus PT_i, \quad CT_i \leftarrow S_r \tag{6}$$

$$S \leftarrow f^b(S) \tag{7}$$

3.2.4 Finalization

In this step, 128-bit Tag T is generated. This T is used for verification in the decryption process. The output of the decryption process is valid PT if the T is verified. Otherwise, CT is tampered with and cannot be trusted. In Ascon, both encryption and decryption follow the same operations. The only difference is that in decryption, CT_i is processed instead of PT_i. In terms of the hardware, the same hardware can be used for both encryption and decryption. In the finalization step, internal state and key S_K are XORed and then 12 permutation rounds are performed. Tag T is generated when the least significant 128-bit of the key S_K is XORed with the least significant 128 bits in the state. The T and C_T are the outputs of the encryption algorithm. The mathematical representation is shown in Eqs. (8) and (9).

$$S_r \leftarrow f^a(S \oplus (0^r\|S_K\|0^{320-r-k})) \tag{8}$$

$$T \leftarrow \lceil S \rceil^{128} \oplus \lceil S_k \rceil^{128} \tag{9}$$

3.3 Permutation

The critical operation in Ascon is the permutation function, an SPN-based round transformation f that is iteratively applied to the state. As depicted in Fig. 2, the permutation function consists of three steps. The function starts with adding a constant, where in each round of permutation, the register word x_2 from the internal state is summed with round constant c_r. The next operation is applied to 64 5-bit S-boxes that are performed in the substitution layer. These S-boxes are the affine transformation of the X mapping of Keccak [21]. The use of affine transformation has improved the cryptographic features of the S-box. Furthermore, the S-box offers a high level of parallelism since it is based on a few logical operations when implemented in hardware. In the diffusion layer, diffusion operation on each register word x_i of 64-bit is performed according to the pre-defined constants established by Ascon team. The diffusion layer is linear, and it is applied to each word of the state. SHA-2 [22] uses a similar function, but in this case, rotations values for each round are different.

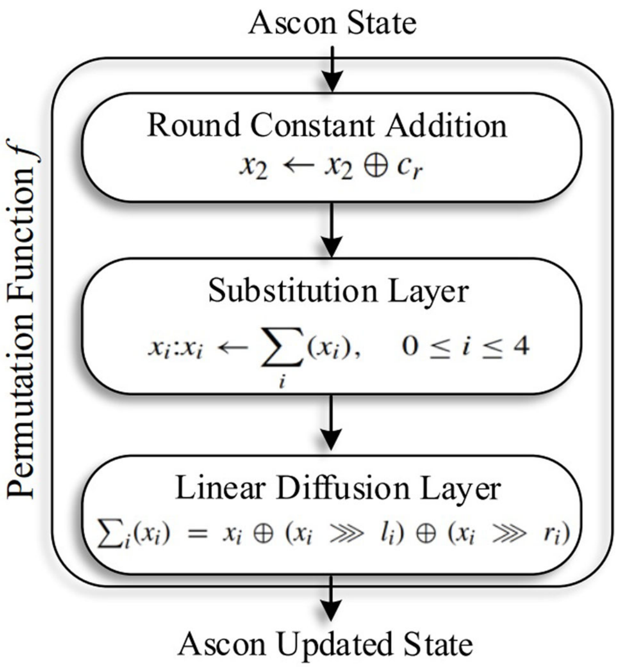
Ascon State
Permutation Function f
Round Constant Addition
$x_2 \leftarrow x_2 \oplus c_r$
Substitution Layer
$x_i{:}x_i \leftarrow \sum_i(x_i), \quad 0 \le i \le 4$
Linear Diffusion Layer
$\sum_i(x_i) = x_i \oplus (x_i \ggg l_i) \oplus (x_i \ggg r_i)$
Ascon Updated State

Fig. 2 Internal architecture of the Ascon permutation function, including three internal layers

4 Proposed Implementation Strategies

When it comes to the exploration of performance-area tradeoffs, various micro-architectures could be explored as design options. The most straightforward one is the loop folded architecture (or the standard architecture) [23]. Optimized architectures target high-performance cores or low-area (and low-power) applications. For high-performance cores, parallelization is added by loop unrolling, i.e., the permutation hardware is replicated to boast TP performance. For low-area implementation, bit-sliced designs are undertaken by economizing area/power at the expense of lower TP performance by employing resource sharing via bit-sliced architecture [24]. Several of these micro-architectures that this work undertakes are elaborated in Fig. 3.

- **Loop Folded:** A typical loop folded cipher implementation performing one permutation (or round) per clock cycle is shown in Fig. 3a. It is generally regarded as the most conservative architecture in terms of area.
- **Loop Unrolled:** The loop unrolled configuration replicates round (or permutation) resources u times to execute multiple permutations in one clock cycle, where u is the unrolling factor. Consequently, the critical path of the circuit increases, decreasing the maximum operational frequency, and the area also increases. The unrolled design reduces the clock cycles to complete the encryption by a factor of u. A higher TP performance is expected since the propagation delay and the register setup time come only once in the combinational delay for u rounds. This gain in TP is hard to enumerate without experimentation; hence, synthesis profiling is required (a twice unrolled hardware configuration is shown in Fig. 3b). A critical design point relevant to loop unrolling is the fully unrolled architecture described next.

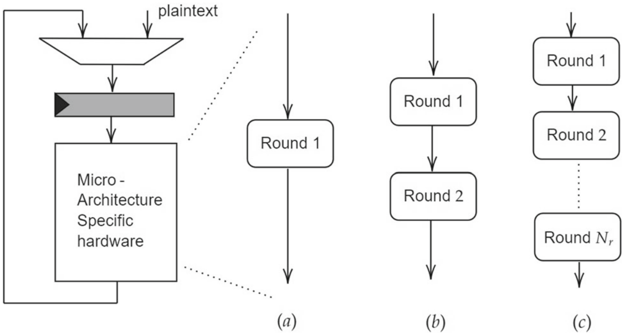

Fig. 3 Various parallel microarchitecture implementations: **a** Loop folded (with an unrolling factor of 1). **b** Loop Unrolled by a factor of 2. **c** Fully unrolled (where the total number of permutations or rounds is assumed to be N_r) [23]

- **Fully unrolled architecture:** A fully unrolled architecture with the unrolling factor u equal to the total number of permutations needed (say N_r) encrypts/decrypts data in a single cycle (as shown in Fig. 3c). A better TP is achieved since now data is processed in a single clock cycle while the decrease in operating frequency is lesser than u times as the critical path handles one register setup time only.

The detail of these architectures is provided in the following sections.

4.1 Loop folded and unrolled architectures

In order to achieve optimised results for the implementation of Ascon, the internal architecture that corresponds to the permutation round is exploited. The number of permutation rounds executed in each clock cycle is varied (from loop folded to unrolled), with the aim to achieve the best results in terms of both area consumption and computation time. The area reduction for these architectures is significant, making them an ideal choice of implementation for the applications in IoT ecosystems.

Considering the internal architecture of Ascon, several number of permutation rounds can be squeezed to execute in a single clock cycle by considering certain trade-offs. The number of clock cycles utilized can be reduced by executing more permutations in each clock cycle, however, on the other hand the hardware area consumption increases. The clock cycles consumption is inversely related to the number of permutations performed during each clock cycle. Encrypting 64-bit data requires $36/f$ clock cycles for Ascon-128 and $40/f$ clock cycles for Ascon-128a, where f is the number of permutations executed in one clock cycle. As the number of permutations during each step is either six or 12 for Ascon-128, the number of permutations in each clock cycles can always be factors of six and 12. The same goes for Ascon-128a that requires eight or 12 permutation rounds in each step and the number of permutations executed in each clock cycles must always be a factor of eight and 12. As with more permutations being executed in clock cycles, the area consumption grows linearly, making it crucial to find the balance between area and latency. Thus the optimized implementation strategies tend to generate varying TP and area consumption ideal for IoT applications.

The details of the internal architecture and working of loop folded architecture is shown in the Fig. 4, where Fig. 4a represents one permutation round and Fig. 4b represents the internal architectures being implemented in hardware. The same hardware architecture can be scaled to any number of permutations, u. The Fig. 4 shows the core part that executes the main permutation and additional circuitry that constitutes of control and data paths. As the main operations during permutation is round constant addition, S-boxes and the linear diffusion layer, a few additional operations are required to make the output of permutation as the input to the next round by re-utilizing the same hardware in a feedback mechanism.

For the loop unrolled architecture, the permutation architecture is scaled with factor u allowing to accommodate more permutations in one clock cycle. In this case, the cipher core is responsible for all the permutation operations and it constitutes majority of the area in Ascon architecture. The additional circuitry in this case constitutes negligible area compared to the permutations. When a certain number of permutation rounds that are being executed in one clock cycles are performed, the output is stored in five registers, each of which is 64-bit to hold the state of 320-bit. Upon the start of next permutation, the updated state is fed to the same hardware by utilizing the control circuitry. Main operations during encryption/decryption is performed by cipher core while control circuitry manages the data flow.

4.2 Fully unrolled architecture

Implementation of Ascon by employing fully unrolled architecture suggests to process the encryption of input data of 64-bit by a combinational circuit. This technique results in the generation of higher TP. The internal operations of the encryption process (initialization, processing AD, processing PT and finalization) are fully unrolled and as a result, the encryption/decryption of data is mapped into a combinational circuit. In order to achieve the high TP, the intermediate state values are not stored in the registers, rather they are directly used as the input for the next stages. This can be done by pre-allocating the hardware for the subsequent stages. Since the need for synchronizing the operations across different data paths have been minimized, the overall performance in terms of TP can be enhanced. However, the fully unrolled implementation approach is associated with significant hardware cost, making it less suitable for IoT applications with strict energy and hardware area constraints. It assumes that the hardware for the next round is already allocated, making it more suitable for applications where area and power utilisation is not a severe requirement.

The Ascon encryption employing the fully unrolled architecture is mapped to the hardware in a similar fashion as shown in the Fig. 1. The first step of the algorithm is initialization step, in which 12 permutation rounds are executed on a 320-bit through 5 data paths of 64-bit each. To achieve high TP, the permutation hardware f^{12} is fully unrolled with 12 rounds. This is followed by AD and PT processing that undergo a similar process, but the permutation hardware is only unrolled for six rounds (f^6). Finally, the finalization step necessitates 12 fully unrolled permutations. The hardware implementation involves the design of four separate modules, which are then connected in a series to generate the final output.

5 Implementation and results

5.1 Evaluation setup and baselines

All the designs are described in Verilog and verified using Synopsys VCS. The designs are then synthesized and mapped to SAED 32 nm and open sourced[2] 45 nm standard cell library using the Synopsys Design Compiler. The post-synthesis gate-level simulation is performed again using Synopsys VCS thereby also generating a switching activity interchange format (SAIF) file based on random vectors. The power dissipation is then estimated by the Prime Power tool by annotating the SAIF file to the netlist. The power consumption is measured at 28 ns and 14 ns clock period unless stated otherwise for 32 nm and 45 nm process design kits (PDKs), respectively. All the experiments are performed on a Linux machine with 128 GB memory. To assess the efficiency of hardware implementations, the following set of metrics are evaluated.

1. **Area**: The area of designs is estimated by the synthesis tool and is specified as μm^2. Some works in the literature used gate equivalent (GE), where one GE is equivalent to the area of a two-input NAND gate with the lowest driving strength of the appropriate technology.

[2] https://vlsiarch.ecen.okstate.edu/flow/.

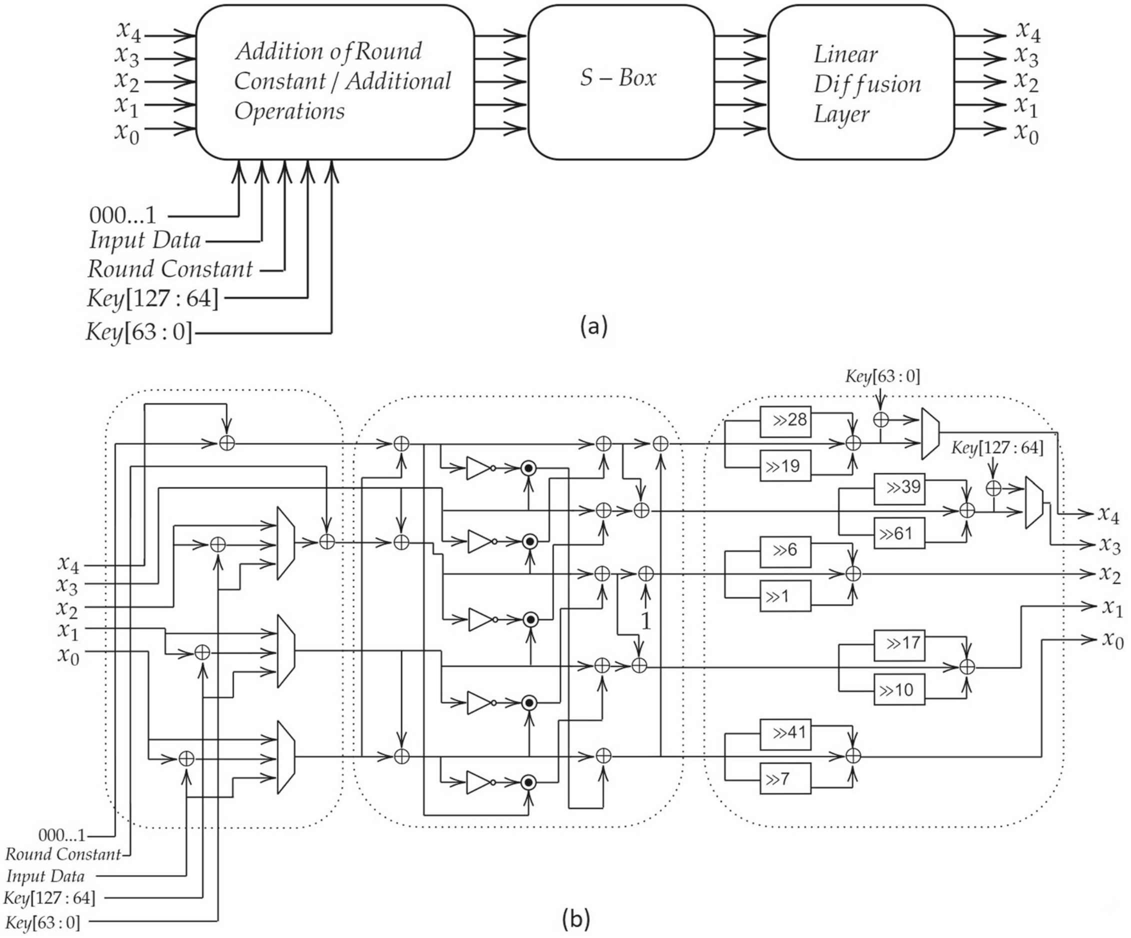

Fig. 4 **a** Ascon permutation. **b** Internal architecture for permutation round

2. **Throughput (TP)**: TP is the sustainable rate at which new output is produced. It is the product of the operating frequency and the interface bits produced per cycle. It is expressed as Byte/word/bits per second. The units we have used in this draft are Gigabits per second (Gbps).
3. **Operating Clock Period (or the Delay)**: It is the inverse of operating frequency of the design used for synthesis. The clock period unit we have used in this draft is nanosecond (ns).
4. **Power**: The power consumption on the gate level is estimated by switching activity using Synopsys PrimeTime-PX tool, the units used in this draft are milliWatt (mW).
5. **Power-delay product (PDP)**: It is the product of power and the processing clock period, its units are ns·mW.
6. **Area-delay product (ADP)**: It is the product of the processing clock period and the area of the design. The units used are ns·μm^2.
7. **Throughput per Area (TP/A)**: It is the ratio of the TP to area utilisation. It is sometimes referred as TPAR or throughput per area ratio. A higher number signifies a better metric and vice versa. Its units are Gbps/μm^2.

5.2 Implementation results and comparisons

For the fully unrolled architecture of Ascon-128 and Ascon-128a, starting from the initialization where the 12 permutations are executed with the hardware dedicated for each permutation round. The same procedure is repeated for processing AD and PT following the finalization step. The important point during the unrolled implementations is the realization of allocated hardware connected serially to generate the final output. The implementation results for Ascon-128 and Ascon-128a are shown in Table 4.

For the loop folded/unrolled implementations, a certain number of permutation rounds are executed in each clock cycle. For Ascon-128 the possible number of permutations in one clock cycles could be $\{1, 2, 3,$ and $6\}$. Similarly, for the Ascon-128a, the possible number of permutations being executed in one clock cycle is $\{1, 2,$ and $4\}$, where one round in each clock cycle is called loop folded architecture. When considering hardware implementations, it is necessary to analyse several critical parameters to evaluate performance effectively.

Table 4 Performance Results for Ascon-128 and Ascon-128*a*

Design	Parameters	Area (k) (μm^2)	Delay (ns)	Power (mW)	PDP (ns·mW)	ADP (k) (ns·μm^2)
		Implementation based on 32 nm PDK				
*Ascon*128	Loop folded	36.7	1.44	1.01	1.45	52.8
	Loop unrolled ($u = 2$)	45.8	2.17	1.14	2.47	99.3
	Loop unrolled ($u = 3$)	53.8	2.76	1.32	3.64	148.4
	Loop unrolled ($u = 6$)	76.2	4.42	1.87	8.27	336.8
	Fully unrolled	277.1	23.54	6.8	160.07	6522.9
*Ascon*128*a*	Loop folded	38.5	1.43	1.05	1.5	55.0
	Loop unrolled ($u = 2$)	48.3	2.17	1.22	2.65	104.8
	Loop unrolled ($u = 4$)	61.9	3.16	1.54	4.87	195.6
	Fully unrolled	306.6	26.22	7.93	207.92	8039.0
		Implementation based on 45 nm PDK				
*Ascon*128	Loop folded	32.4	0.66	2.70	1.79	21.4
	Loop unrolled ($u = 2$)	39.9	0.97	2.75	2.67	38.7
	Loop unrolled ($u = 3$)	50.7	1.29	2.81	3.62	65.4
	Loop unrolled ($u = 6$)	78.0	1.95	2.96	5.79	152.8
	Fully unrolled	327.3	10.87	7.97	86.74	3558.7
*Ascon*128*a*	Loop folded	34.5	0.65	2.86	1.86	22.4
	Loop unrolled ($u = 2$)	42.4	0.98	2.91	2.85	41.6
	Loop unrolled ($u = 4$)	59.1	1.39	3.07	4.18	82.1
	Fully unrolled	362.2	12.11	9.71	117.71	4386.3

Moreover, the results have been divided into two subsections, i.e. (1) based on the SAED 32 nm PDK node library and, (2) based on the open-source 45 nm standard cell library

5.2.1 SAED 32 nm PDK

Area and timing result evaluations: In case of 32 nm SAED PDK library, for the area consumption, it can be clearly observed in Table 4 that the fully unrolled implementations require the highest area for both Ascon-128 and Ascon-128*a*. This is because the hardware is pre-allocated for all the steps involved. For the folded/unrolled implementations, one of the main goal is to conserve the area using the area re-utilization strategies. For both Ascon-128 and Ascon-128*a*, when the loop folded architecture is executed, the area utilization is minimum. To generate the entire encryption process, the hardware consists of a single permutation round along with supplementary circuitry for generating the feedback path. When the hardware realization is based on executing two rounds of permutations in one clock cycle (loop unrolled with $u = 2$), the hardware area consumption increases that is depicted from the Table 4 as well. As more and more permutations are squeezed in each clock cycle, more area consumption is required making a linear relation with number of rounds and area consumption. Increasing the number of permutations being executed in one clock cycle impacts the area as well as the critical path. More number of permutations in series during each clock reflects longer critical path. A critical path refers to the longest path of logic gates or sequential elements in a circuit that determines the maximum delay between the input and output of the circuit. The results from the Table 4 depicts the same as we increased the number of permutations in the clock cycle, the critical path is elongated incorporating larger delays in the implementations.

Power-delay product and area-delay product: There is a positive correlation between power consumption and chip area. Larger chip sizes consume more power because more transistors and interconnects contribute to power dissipation. However, this correlation is not linear, as power consumption also depends on the activity and switching characteristics of the circuitry. The same pattern can be seen as the power consumption increases with the number of permutation rounds increased in each clock cycle. In order to gain a more comprehensive understanding of the operational parameters, measurements are also taken for the PDP and the ADP. The PDP increases while traversing down the column as both power and delay increase. The same trend can be seen for ADP as well. Figure 5 shows the radar chart image for the implementation results of Ascon-128 and Ascon-128a. The graphical representation depicts effectively and clearly the observed fact, making it readily understandable and comprehensible.

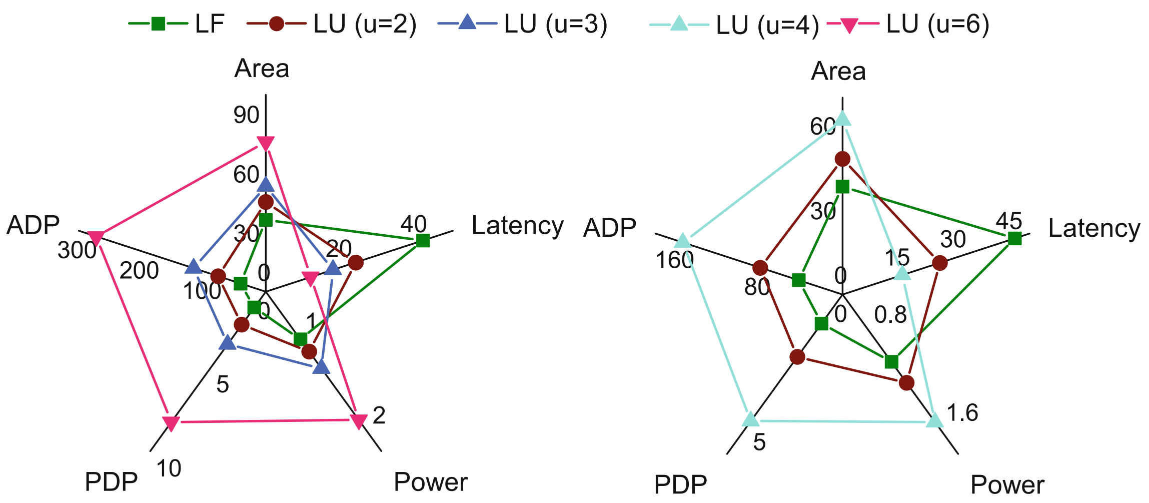

Fig. 5 Ascon-128 and Ascon-128a performance visualization for SAED 32nm PDK

Throughput and throughput/area ratio: Two important parameters to measure the performance of the circuit are TP and TP/A ratio. The TP is measured as the number of bits that can be processed in a certain time while the TP/A accommodates area utilization as well and is the ratio of TP to the area consumption. The fully unrolled implementations, being a combinational circuit, provides the highest TP for both Ascon-128 and Ascon-128*a*. Comparing the TP when implementing a number of rounds in each clock cycle, TP results are interesting. Staring from loop folded architecture, as the number of rounds executed in each clock cycle is increased the TP initially increases however, increasing the number of rounds beyond that point causes a reduction in the TP as seen from the Table 5. The explanation lies in the fact that while the delay has been increased, the latency itself hasn't experienced a significant decrease. This is because the total delay is determined by the multiplication of both delay and latency factors. Therefore the TP is maximized when three permutation rounds are processed in one clock cycle. The TP/A ratio, on the other hand is minimum for fully unrolled implementations because of their inherited area inefficiency. TP/A ratio decreases when the number of permutations are increased as the hardware footprint of the implementations increases. The TP and TP/A ratio for the Ascon-128*a* is almost double compared to Ascon-128, because Ascon-128*a* processes data in the blocks of 128-bit compared to 64-bit data processed by Ascon-128.

5.2.2 Open-source 45 nm PDK

The implementations have also been performed using the 45 nm PDK for both Ascon-128 and Ascon-128a cases. This enables a more generic and reproducible analysis due to the PDK library being open sourced. Majority of the parameters for both ASIC platforms exhibit similar characteristics. The area consumption for both ASIC platform follow the similar trend, with more permutations in one clock cycle requiring more chip area. It should be noted that the frontend synthesis tool ignores the interconnects while reporting area for the 45 nm PDK. The numbers considering 45 nm PDK thus are more of a speculative nature but indicate a certain trend dictated by the synthesis tool when using open-source PDKs. The delay again exhibits the same characteristics as higher the critical path, higher is the delay. Same is the case with power and PDP. ADP increases while traversing down the column because of the increase in chip area. The TP and TP/A results are also provided in Table 5. The TP and TP/A ratio follow the same trend when the implementations have been performed for the 32 nm ASIC PDK. The reason for such trend is already discussed in detail in the previous section. By open sourcing the design files and the usage of an open-source PDK library, it is ensured that the results and hence the overall trend is reproducible by the research community with the only requirement being the availability of the required Synopsys tool chain.

Comparison to the state-of-the-art ASIC accelerators: The majority of the implementations found in the literature targeted the FPGA platform, where comparison in terms of the performance parameters is infeasible due to platform disparities. Similarly, the ASIC implementations in the literature employ different process technologies, thus making the comparison part challenging. TP is one of the key factors for performance comparison as it incorporates latency and delay information. The ASIC implementations presented in [11] produce a TP of 384 Mbps for the low-area architecture, which is 2.75 times less than the TP of 1.058 Gbps calculated by the proposed architecture. In terms of area utilisation, the results have been presented in KGE (gate-equivalents) rather

Table 5 Performance details of Ascon-128 and Ascon-128a with different configurations

Design	Parameters	Latency cc	TP Gbps	TP/A Gbps/µm^2
	Implementation based on 32 nm PDK			
*Ascon*128	Loop folded	42	1.058	0.0288
	Loop unrolled ($u = 2$)	24	1.228	0.0268
	Loop unrolled ($u = 3$)	18	1.288	0.0239
	Loop unrolled ($u = 6$)	12	1.206	0.0158
	Fully unrolled	–	2.718	0.0098
*Ascon*128*a*	Loop folded	46	1.945	0.0505
	Loop unrolled ($u = 2$)	26	2.268	0.0469
	Loop unrolled ($u = 4$)	16	2.531	0.0408
	Fully unrolled	–	4.881	0.0159
	Implementation based on 45 nm PDK			
*Ascon*128	Loop folded	42	2.274	0.0715
	Loop unrolled ($u = 2$)	24	2.749	0.0668
	Loop unrolled ($u = 3$)	18	2.777	0.0559
	Loop unrolled ($u = 6$)	12	2.721	0.0348
	Fully unrolled	–	5.887	0.0192
*Ascon*128*a*	Loop folded	46	4.280	0.1240
	Loop unrolled ($u = 2$)	26	5.023	0.1184
	Loop unrolled ($u = 4$)	16	5.755	0.0973
	Fully unrolled	–	10.56	0.0291

than (µm^2), making it difficult to make a realistic comparison. The architecture in [12] reports higher TP compared to our proposed architecture, however it is achieved at the cost of higher area sacrifice (almost 4 times higher area consumption compared to [11]). TP has not been reported in [13], making it hard to compare in terms of performance. Again, due to different process technologies, a comparison in terms of different parameters could be misleading. In addition, none of the literature includes results for Ascon-128a, making this the first reported ASIC results for Ascon-128a to the best of the author's knowledge.

6 Applications

The sensors in an IoT ecosystem gather different kind of data which is then transferred to the gateway by employing duplex communication. The communication between the sensors and the gateway must be secured using lightweight cryptographic functions, i.e. Ascon. The amount of data generated and transmitted for each sensor is different depending on each application. The CCTV cameras used for video surveillance usually requires the images to be encrypted at a much higher speed for real time transmission. This requires the encryption process to be fast enough to meet the demands of high speed. The fully unrolled implementation strategy fits ideally to this case. Similarly the loop folded/unrolled implementations are highly efficient with respect to area consumption and power, finds its applications on IoT environments as well. The most suitable applications for these implementations include smart traffic solutions, smart parking, and city energy consumption. The data generated through these applications is not tremendous, while requiring efficiency with respect to the area and TP, thus the loop folded/unrolled implementations are best in this case.

7 Conclusion

This paper has presented an accelerator architecture for the lightweight data protection algorithm Ascon, supporting its variants Ascon-128 and Ascon-128a. The accelerator offers three implementation styles: loop folded, loop unrolled, and fully unrolled. Synthesis results are provided on a 32 nm PDK, with evaluations of area, timing, and power. Fully unrolled implementations are found to be optimal for high TP applications, albeit with higher area overhead. On the other hand, loop folded and unrolled implementations adjust the number of permutation rounds per clock cycle, demonstrating a trade-off among various parameters. This trade-off makes the implementations suitable for diverse applications within IoT ecosystems. The same trend in terms of performance parameters is also demonstrated with an open-source 45 nm PDK which together with the open sourcing of the design files enable prompt reproducibility of the results for further research.

Author Contributions S.K. initiated the idea and wrote the main Verilog code. K.I. and F.M. generated the corresponding results for the ASIC platforms. Y.S., A.A. and M.R. contributed in writing the manuscript and prepared the figures. M.I. and A.K. gave their expert opinion to elevate the quality of the paper. All the authors have reviewed the manuscript.

Data availability No datasets were generated or analysed during the current study.

Declaration

Conflict of interest The authors declare that they do not have Conflict of interest.

References

1. Amara, M., Siad, A.: Elliptic curve cryptography and its applications. In: International Workshop on Systems, Signal Processing and Their Applications, WOSSPA, pp. 247–250. IEEE (2011)
2. Heron, S.: Advanced encryption standard (AES). Netw. Secur. **2009**(12), 8–12 (2009)
3. Jimale, M.A., Z'aba, M.R., Kiah, M.L.B.M., Idris, M.Y.I., Jamil, N., Mohamad, M.S., Rohmad, M.S.: Authenticated encryption schemes: a systematic review. IEEE Access **10**, 14739–14766 (2022)
4. Rogaway, P.: Authenticated-encryption with associated-data. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, pp. 98–107 (2002)
5. Sarkar, P.: A simple and generic construction of authenticated encryption with associated data. ACM Trans. Inf. Syst. Secur. (TISSEC) **13**(4), 1–16 (2010)
6. Mansoor, K., Ghani, A., Chaudhry, S.A., Shamshirband, S., Ghayyur, S.A.K., Mosavi, A.: Securing IoT-based RFID systems: a robust authentication protocol using symmetric cryptography. Sensors **19**(21), 4752 (2019)
7. Sun, C., Liu, J., Xu, X., Ma, J.: A privacy-preserving mutual authentication resisting DoS attacks in VANETs. IEEE Access **5**, 24012–24022 (2017)
8. Turan, M.S., et al.: Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process (2023)
9. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Ascon v1.2. Submission to the CAESAR Competition **5**(6), 7 (2016)
10. Cryptographic competitions, CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness https://competitions.cr.yp.to/caesar.html. Accessed 12 Dec 2023
11. Groß, H., Wenger, E., Dobraunig, C., Ehrenhöfer, C.: Suit up!—made-to-measure hardware implementations of Ascon. In: 2015 Euromicro Conference on Digital System Design, pp. 645–652. IEEE (2015)
12. Gross, H., Wenger, E., Dobraunig, C., Ehrenhöfer, C.: Ascon hardware implementations and side-channel evaluation. Microprocess. Microsyst. **52**, 470–479 (2017)
13. Roussel, N., Potin, O., Di Pendina, G., Dutertre, J. M., Rigaud, J.B.: CMOS/STT-MRAM based Ascon LWC: A power efficient hardware implementation. In: 2022 29th IEEE International Conference on Electronics, Circuits and Systems (ICECS), pp. 1–4. IEEE (2022)
14. Yalla, P., Kaps, J. P.: Evaluation of the CAESAR hardware API for lightweight implementations. In: 2017 International Conference on ReConFigurable Computing and FPGAs (ReConFig), pp. 1–6. IEEE (2017)
15. Diehl, W., Farahmand, F., Abdulgadir, A., Kaps, J. P., Gaj, K.: Face-off between the CAESAR lightweight finalists: ACORN versus Ascon. In: 2018 International Conference on Field-Programmable Technology (FPT), pp. 330–333. IEEE (2018)
16. Rezvani, B., Coleman, F., Sachin, S., Diehl, W.: Hardware implementations of NIST lightweight cryptographic candidates: a first look. In: Cryptology ePrint Archive (2019)
17. Jellema, L.: Optimizing Ascon on RISC-V. Bachelor Thesis, Radboud University (2019)
18. Khan, S., Lee, W.K., Hwang, S.O.: Scalable and efficient hardware architectures for authenticated encryption in IoT applications. IEEE Internet of Things J. **8**(14), 11260–11275 (2021)
19. Khan, S., Lee, W. K., Hwang, S. O.: Evaluating the performance of ascon lightweight authenticated encryption for AI-enabled IoT devices. In: 2022 TRON Symposium (TRONSHOW), pp. 1–6. IEEE (2022)
20. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Permutation-based encryption, authentication and authenticated encryption. In: Directions in Authenticated Ciphers, pp. 159–170 (2012)
21. Bertoni, G., Daemen, J., Peeters, M., Van Assche, G.: Keccak specifications. Submission to NIST (Round 2) **3**(30), 320–337 (2009)
22. Standard, S. H. (1995). Secure hash standard. *FIPS PUB*, 180-1
23. Khalid, A., Paul, G., Chattopadhyay, A.: Domain Specific High-Level Synthesis for Cryptographic Workloads. Springer, Singapore (2019)
24. Khalid, A., Hassan, M., Chattopadhyay, A., Paul, G.: RapidFeinSPN: a rapid prototyping framework for Feistel and SPN-based block ciphers. In: Information Systems Security: 9th International Conference, ICISS 2013, Kolkata, India, December 16–20, 2013. Proceedings 9, pp. 169–190. Springer, Berlin (2013)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.