Suit up! Made-to-Measure Hardware Implementations of ASCON

Hannes Groß, Erich Wenger, Christoph Dobraunig, and Christoph Ehrenhöfer

Graz University of Technology,
Institute for Applied Information Processing and Communications (IAIK),
Inffeldgasse 16a, 8010 Graz, Austria
{Firstname.Lastname}@iaik.tugraz.at,
christoph.ehrenhoefer@student.tugraz.at

Abstract. Having ciphers that provide confidentiality and authenticity, that are fast in software and efficient in hardware, these are the goals of the CAESAR authenticated encryption competition. In this paper, the promising CAESAR candidate ASCON is implemented in hardware and optimized for different typical applications to fully explore ASCON's design space. Thus, we are able to present hardware implementations of Ascon suitable for RFID tags, Wireless Sensor Nodes, Embedded Systems, and applications that need maximum performance. For instance, we show that an ASCON implementation with a single unrolled round transformation is only 7 kGE large, but can process up to 5.5 Gbit/sec of data (0.75 cycles/byte), which is already enough to encrypt a Gigabit Ethernet connection. Besides, ASCON is not only fast and small, it can also be easily protected against DPA attacks. A threshold implementation of ASCON just requires about 8 kGE of chip area, which is only 3.1 times larger than the unprotected low-area optimized implementation.

Keywords: Authenticated encryption, CAESAR competition, hardware design, threshold implementation, ASCON

1 Introduction

Symmetric cryptography has a rich history of competitions to find good and secure cryptographic primitives. Winners of such competitions like Rijndael—now known as the Advanced Encryption Standard (AES)—serve as base of our modern information and communication systems. Currently, in a competition called CAESAR [2], over 45 different authenticated encryption schemes battle for entry in the final portfolio to be tomorrows information security working horses. The final portfolio will be announced at the end of 2017 and till then, the participating primitives have to survive an annual selection process.

Cryptanalysis is not enough to settle the battle for the entry in the next rounds. It is likely—as in competitions before—that we get to some point during CAESAR, where every remaining authenticated encryption scheme, has no shown cryptographic weaknesses and equivalent cryptographic properties as the other candidates. Thus, cryptanalysis results of the pure algorithms are little help at this point regarding the selection

process. Therefore, it is useful to compare other properties of the different algorithms like speed and code size of software implementations, area and throughput of hardware implementations, or the costs of protection mechanism against implementation attacks like side-channel analysis.

Providing hardware implementations of CAESAR candidates is an important task to base the decisions during CAESAR on quantifiable numbers and not on intuition. Thus, preliminary hardware implementations have been included in many submissions to the CAESAR competition like for NORX [1] and ICEPOLE [15]. However, these preliminary implementations are often only optimized for one specific design criteria (e.g. speed), and do not include any mechanisms to counteract side channel attacks. Moreover, not all CAESAR candidates already provide hardware implementation results. Therefore, the design space evaluation and comparison of the CAESAR candidates is an ongoing process [2].

In this work, we provide the first hardware implementations for the CAESAR candidate ASCON. We maximize the impact of our contribution by not focusing on only one criteria. Instead, we investigate the requirements of typical authenticated encryption applications and check the suitability of ASCON for all of them. In particular, we present implementations of ASCON suitable for RFID tags, Wireless Sensor Nodes, Embedded Systems, and applications, which need a maximum of performance. In addition, we provide protected implementations of ASCON for applications, where side-channel analysis is a threat. The resulting ASCON implementations are compared with all the hardware implementations of other CAESAR competitors available to us. This comparison shows that ASCON provides the flexibility to meet the requirements of many AE applications and is even able to compete with ciphers designed for one specific use case. To the best of our knowledge, this is the first work where such a vast amount of design spaces is evaluated and compared to other competitors regarding CAESAR.

This paper is structured as follows: Section 2 derives the requirements of today's most demanding applications for authenticated encryption. The ASCON algorithm itself is then briefly presented in Section 3. In Section 4, the different variants of ASCON hardware designs are discussed and their underlying design principles are explained in more detail. Furthermore, it is shown how the ASCON implementations can be easily protected against first-order differential power analysis (DPA) attacks. Finally, the results of the hardware implementations are discussed in Section 5, and compared to other CAESAR candidates and other authenticated encryption schemes.

2 Target Applications of Authenticated Encryption Schemes

The authenticated encryption candidates from the CAESAR competition will be implemented in software and in hardware. Although general-purpose microprocessors are constantly improving, certain demanding applications necessitate dedicated hardware designs. Software implementation would simply not perform sufficiently. In the following, a summary of typical applications is given to elaborate their respective requirements on authenticated encryption implementations.

A common concern of symmetric cryptography is **performance**. Especially servers and encrypted/authenticated high-speed network links potentially require hardware ac-

Table 1. Optimization goals for different authenticated encryption applications

Application	Optimization Goal	Interface
High performance computing	high throughput	custom
RFID tags	low power and low area	custom
Wireless sensor nodes	low energy	memory-mapped
Embedded systems	high throughput per area	memory-mapped

celerators to cope with the protected network traffic. Prominent examples for hardware accelerators are Intel's AES [10] and upcoming SHA [11] instructions. The most important characteristic of those hardware accelerators is *throughput*. However, other applications have different requirements.

One target application of authenticated encryption are passively powered **Radio Frequency Identification** (RFID) tags. These tags are wirelessly supplied by the reader field and have extremely high cost constraints in order to fulfill the requirements for mass production. The vital characteristic of these tags is the maximum read range. The limiting factor for the read range of passive RFID tags is their power consumption. Therefore, CAESAR candidates suitable for RFID should consume *very low power* and should be optimized for *low area*.

Similarly to RFID tags, **Wireless Sensor Nodes** constitute an important topic for the community. When wireless sensor nodes are deployed in the field, a single battery should ideally last for a life-cycle. Therefore, it is crucial to have CAESAR candidates, which can be optimized for *low energy* consumption. Current wireless sensor nodes come with low-energy microprocessors. Therefore, a low-energy cipher design must come with a bus interface to be usable for the microprocessor.

While servers, RFID tags, and wireless sensor nodes are very specific applications, most currently deployed systems nowadays are **Embedded Systems** that come with a microprocessor and a lot of peripherals. These microprocessors are the default solution for many consumer and industrial applications. Manufacturers add small but performant memory-mapped cryptographic coprocessors in order to fulfill versatile security tasks. On the one hand these coprocessors should be as small as possible, but on the other hand they should also be very fast which is normally a contradiction. Therefore, they can be best characterized in terms of *throughput per area*—in the following also referred to as *efficiency*.

Table 1 summarizes the optimization goals for different applications measured in traditional hardware metrics. An additional design dimension is **implementation security**. In fact, there is a whole class of practical side-channel attacks which modern devices need to be prepared for. Protecting implementations against physical attacks is not trivial. Since Kocher et al.'s [13] paper about differential power analysis, the research community struggles with the secure implementation of AES. To avoid such challenges, it is an important requirement for all CAESAR candidates to be easily protectable against implementation attacks.

In the following, it is shown that ASCON is extremely flexible and can be optimized according to the needs of all of the above mentioned applications. Furthermore, it is demonstrated that ASCON can be efficiently protected against power analysis attacks.

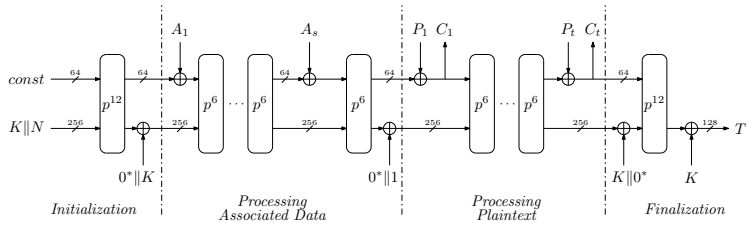


Fig. 1. The encryption of ASCON-128 (taken from [8])

3 ASCON

ASCON [8] has a sponge-like mode of operation as depicted in Figure 1. Its state size, the permutation p and mode of operation are chosen in a way that allows compact hardware implementations, while still providing high throughput. ASCON comes in two different versions, namely ASCON-128 and ASCON-96 with 128 and 96 bit security level, respectively. This paper focuses on ASCON-128, while the modifications for the faster ASCON-96—essentially a doubling of the rate—are straight-forward.

3.1 Mode of Operation

ASCON has a state size of 320 bits (consisting of five 64-bit words $x_0, \ldots, x_4$) that are updated in four phases: *Initialization*, *Processing of Associated Data*, *Processing of Plaintext/Ciphertext*, and *Finalization*. All phases use the same permutation function p that is applied twelve times in the *Initialization* and *Finalization* phase. The lighter variant of p with six rounds is used for processing the data and ensures high performance. The data is handled in 64-bit blocks.

The *Initialization* phase takes the secret key K (128 bits) and the public nonce N (128 bits). This nonce has to be fresh for every encryption and must not be used twice. If the nonce is used twice or multiple times, then the confidentiality is jeopardized.

After the *Initialization* phase the optional associated data A_i is processed. Associated data is information, which does not need to be confidential, but must not be altered by an attacker. Each block A_i is added to the secret state. If there is no associated data to process, the whole step can be omitted.

In the *Encryption* phase, each plaintext block P_i is xored with the secret state to produce one ciphertext block C_i. Six consecutive round transformations p are executed for each of the 64-bit data blocks.

After the generation of the ciphertext, the *Finalization* starts. The output of the *Finalization* is the 128-bit tag T. With the help of this tag, modifications of the ciphertext and the associated data can be detected during decryption (validation). Decryption is very similar to encryption. Just the part, where the ciphertext is processed instead of the plaintext differs slightly. Thus, no inverse of the permutation is needed for decryption. So, both encryption and decryption can be implemented with just a slight overhead compared to encryption only.

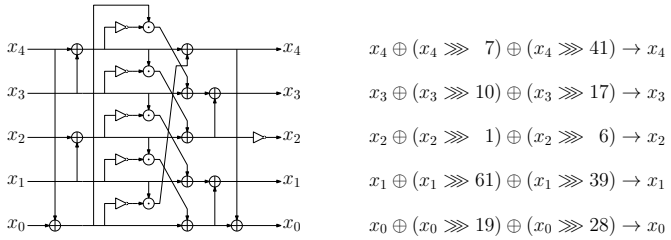


Fig. 2. Substitution layer with 5-bit S-box (left) and linear layer (right) (taken from [8])

3.2 Permutation

ASCON-128 uses two permutations p^6 and p^{12}. The two permutations are the 6 and 12 iterative execution of the round transformation p. The round transformation p consists of a constant addition to x_2, followed by an application of a substitution layer, and a linear layer.

The substitution layer is the parallel application of 64 5-bit S-boxes. The S-boxes used for ASCON are an affine transformation of the χ mapping of Keccak [3]. This affine mapping improves some cryptographic properties of the Keccak's χ mapping, while still leaving the core of the S-box and therefore the algebraic degree of 2 intact. Moreover, the ASCON S-box can be implemented, using only a few logical operations, which are highly parallelizeable (Figure 2).

The linear layer consists of five applications of the function $\Sigma_{l_i,r_i}(x_i) = x_i \oplus (x_i \ggg l_i) \oplus (x_i \ggg r_i)$ to each 64-bit word of the state $(x_0, \ldots, x_4)$. The Σ function is similar to the one used in SHA-2 [16], except that other rotation values (l_i, r_i) are used. The rotation values (l_i, r_i) are different for every 64-bit word in one round and are: $(19, 28)$ for x_0, $(61, 39)$ for x_1, $(1, 6)$ for x_2, $(10, 17)$ for x_3, and $(7, 41)$ for x_4.

3.3 Hardware Security Properties of ASCON

In order to protect ASCON against side-channel attacks it is important to reflect on the properties of ASCON that make the life of an attacker hard. ASCON uses a mode of operation which is based on MonkeyDuplex [7]. In contrast to MonkeyDuplex, ASCON uses a keyed *Initialization* and *Finalization*. This has the effect that a state recovery during the processing of data neither leads to the recovery of the secret key, nor allows universal forgeries.

Therefore, Power-analysis attacks on the data processing phase may be applied in order to recover the internal states, but do not allow the attacker to recover the key. In addition, power-analysis attacks on the *Finalization* are hard, since the attacker would have to attack both the key and many unknown state bits that have no influence on the emitted tag. In fact, the most vulnerable phase is the *Initialization*. For ASCON-128, three out of five input bits of the first S-boxes are publicly known and the other two bits belong to the secret key.

To prevent side-channel analysis, ASCON allows to apply well known protection mechanisms. The ASCON S-box is an affine transformation of the Keccak S-box.

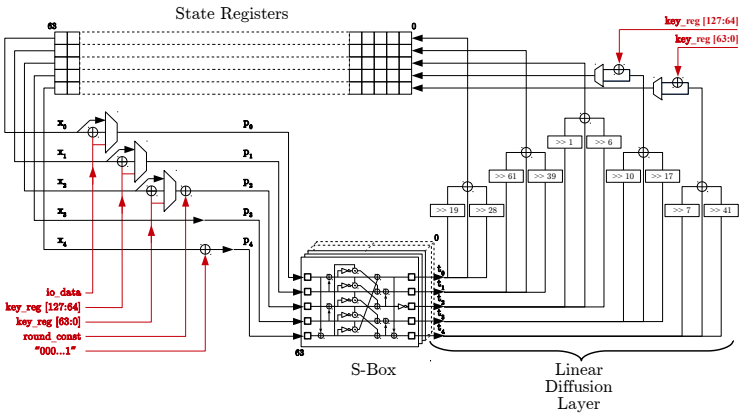


Fig. 3. Datapath of the *fast* variant of ASCON with one round transformation per cycle

Hence, it is possible to reuse essential parts of techniques for side channel protection like the threshold implementation of Keccak from Bilgin et al. [4] as discussed in Section 4.4. Therefore, ASCON can be protected against first-order DPA attacks by using only three shares and four random bits per round.

4 Hardware Designs

ASCON allows to be optimized for many practical applications, where both confidentiality and authenticity are required. These include all applications mentioned in Section 2 with their different optimization goals. In the following, three implementation variants with different design goals are introduced.

4.1 High Throughput Design (ASCON-fast)

Due to the low complexity of ASCON's round transformation, it is possible to fully unroll a complete round transformation and still achieve high frequencies. As it turns out, the round transformations are so hardware-friendly that even multiple rounds can be computed in a single clock cycle.

The ASCON-fast variants aim at a maximal data throughput with a minimum of processing delay. Therefore, at least one round transformation is performed in every clock cycle and no pipelining stages are used. Each ASCON-fast variant uses a different number of unrolled round transformations. The datapath of ASCON-fast as it is shown in Figure 3 mainly consists of the unrolled round transformations (five 64-bit state registers, 64 parallel S-boxes, and the linear diffusion layer). Only a few additional multiplexers and XOR-gates are needed to connect the unrolled round transformation with the data bus and the key register.

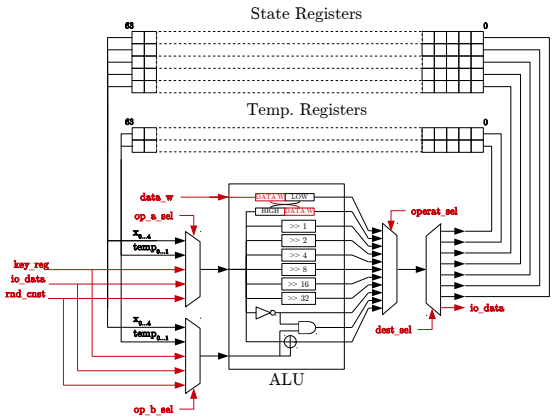


Fig. 4. Datapath of the *64-bit* variant of ASCON

4.2 64-bit Datapath Design (ASCON-64-bit)

The design idea behind the ASCON-64-bit implementation is based on the inherent 64-bit structure of ASCON. Instead of a concrete implementation of the S-box and the linear diffusion layer, this design uses an arithmetic logic unit (ALU) comparable to a microcontroller design. Consequently, the controlpath works similar to a sequential program code that is executed by the datapath in Figure 4. Besides the five state registers, there exist also two temporary registers, which—together with the inputs from the controlpath—form the input operands of the ALU. The ALU itself consists of an iterative barrel-shifter unit, three logic operations, and a data-storage unit that takes the 64-bit bus data input and stores it either in the high or the low part of the selected operand. On the output of the ALU the result of the operation is selected that is then applied to the destination register. During the execution, the S-box and linear layer are iteratively calculated using the operations of the ALU. Thus, one round operation takes 59 clock cycles.

4.3 Low Area Design (ASCON-x-low-area)

The datapath of the so-called ASCON-x-low-area variant (see Figure 5) uses a radical low-area approach, which can be summarized as “one bit operation per cycle”. The state consists of five clock gated shift registers with independent shift-enable inputs. For the S-box calculation, all state registers are activated and shifted bit-slice-wise through the single S-box instance. The result is stored in the least-significant bits of the state. Accordingly, the whole S-box layer operation consumes 64 cycles. The subsequent linear diffusion layer is split up into five interleaved subiterations in which each state register is updated individually. As a single state bit depends on two other bits of the same state row, the linear layer cannot be calculated without temporarily storing either the results or the state row itself, respectively. Thus, another (temporary) shift register is needed that in one iteration holds the result of the current linear layer operation and in the next

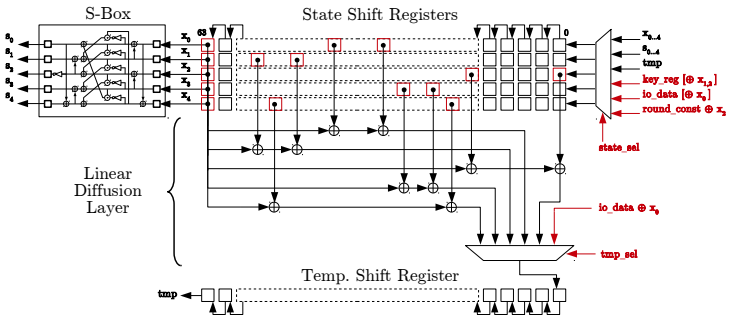


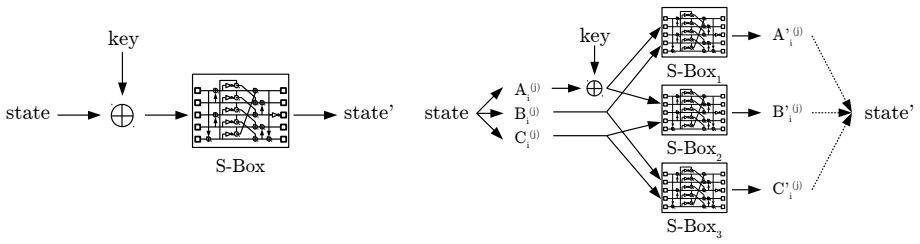
Fig. 5. Datapath of the *x-low-area* variant of ASCON

iteration is used to write the result back. Once the first subiteration of the linear layer is finished, the calculation of the next state row and the write back operation can be done in parallel. This uncompromising low-area approach results in 512 clock cycles per round transformation.

4.4 Protecting ASCON against First-Order Side Channel Analysis Attacks

In practice it is not sufficient to have fast, small, low-power, or low-energy designs. In addition, those designs need to be be protected against the serious threat of implementation attacks like power-analysis attacks. In 2006, Nikova et al. [17] introduced the concept of a threshold implementation (TI) scheme that is provably resistant against first-order DPA attacks. The basic principle behind TI is that the calculations are never performed directly on the security critical data but indirectly by applying modified transformations on so-called shares (named A, B and C in Figure 6.b). Linear transformations, like key additions or ASCON's linear layer, can be performed on each share separately. Non-linear operations, like S-box lookups, have to be carefully transformed to remain valid. To withstand first-order DPA, the new individual S-boxes have to fulfill the correctness, non-completeness, and uniformity properties. A sharing of an S-box is said to be correct if the sum of the resulting output shares equals the result of the S-box function applied to the sum of the input shares (see Figure 6.b). Also the non-completeness property is represented in Figure 6.b by having three S-box component functions which are each independent of at least one input share. The last property which is required for a valid TI is the so-called uniformity which results for an S-box implementation in the equivalent requirement for each component function to be invertible.

One of the best explored algorithm regarding threshold implementations is the new SHA-3 standard Keccak [3]. In 2014, Bilgin et al. [4] not only showed how to share the Keccak's S-box function χ very efficiently with three shares, but also how to reduce the high amount of random bits required to recover the uniformity property. ASCON uses an affine transformation of Keccak's χ function. Therefore, the current findings of Bilgin et al. and all future findings on the Keccak S-box can be directly applied to ASCON. Since, the sharing of the χ function is already well explained by Bilgin et al.,



(a) Key addition and S-box lookup (b) According transformation with three shares

Fig. 6. The principle of secret sharing explained on a simple transformation

the details of ASCON's shared S-box are summarized in Appendix A for the interested reader.

In order to investigate the additional costs for protecting ASCON against first-order DPA attacks, three-share TI versions of the *fast* and the *x-low-area* variants are implemented. ASCON-fast-TI is assumed to be used in an embedded system scenario as a cryptographic co-processor that is controlled by a microcontroller. The microcontroller is responsible for performing the initial sharing of the state and for distributing its source of randomness. The necessary random bits (four fresh random bits per round transformation) are delivered over the bus interface in advance of each round transformation and stored in an internal register.

ASCON-x-low-TI, on the other hand, is designed to be used within a microcontroller-free design, e.g., a commercial RFID tag. Therefore, a direct access to a random number generator is assumed to be available, like it is used in EPC Gen2 UHF tags. Because the ASCON-x-low-TI implementation only performs one S-box operation per cycle, the output of the random number generator is directly used and does not need to be stored in advance.

In the next Section we evaluate how the implemented ASCON designs perform compared to each other and compared to related work.

5 Results

All ASCON designs are implemented in VHDL and evaluated using a Cadence-based ASIC design-flow. The VHDL code is open-source and is available online.[1] This will facilitate future comparisons by fellow researchers using different ASIC or FPGA design flows. For the following results, a 90 nm UMC standard performance low-K library from Faraday is used with a global clock of 1 MHz and a 1 V power supply. The designs are compiled with the Cadence Encounter RTL compiler version v08.10-s28_1 and routed with Cadence NanoRoute v08.10-s155.

The results of all practical evaluations are collected in Table 2. Some implementations can process up to 8 bytes of data in a single clock cycle (0.125 cycles per

[1] http://ascon.iaik.tugraz.at/implementation.html

Table 2. Characteristics of the ASCON-128 hardware implementations

Design	Chip Area w/o interface	Chip Area w/ interface	Throughput		Power at 1 MHz	Energy
	[kGE]	*[kGE]*	*[cycles/byte]*	*[Mbps]*	*[µW]*	*[µJ/byte]*
Unprotected Implementations						
ASCON-fast						
1 round	7.08	7.95	0.75	5,524	43	33
2 rounds	10.61	11.48	0.38	8,425	72	27
3 rounds	14.26	15.13	0.25	10,407	102	25
6 rounds	24.93	25.80	**0.13**	**13,218**	184	**23**
ASCON-64-bit	4.99	5.86	44.25	72	32	1,397
ASCON-x-low-area	**2.57**	**3.75**	384.00	14	**15**	5,706
Threshold Implementations						
ASCON-fast-TI						
1 round	28.61	30.42	0.75	3,774	183	137
2 rounds	47.46	49.13	0.38	6,289	315	118
3 rounds	66.45	68.27	0.25	7,143	447	112
6 rounds	123.52	125.19	**0.13**	**9,018**	830	**104**
ASCON-x-low-TI	**7.97**	**9.19**	384.00	15	**45**	17,234

byte) and other implementations are as small as 2.57 kGE. Especially the characteristics of ASCON-fast with one unrolled round are impressive. This implementation needs 7.08 kGE (7.95 kGE with key register and 64-bit bus interface), reaches a maximum clock frequency of 517 MHz, and can therefore process up to 5.5 Gbit per second. This means that the most straightforward design is easily sufficient to encrypt a gigabit Ethernet connection on the fly. At 100 MHz, the design only needs 529 µW (38 µW static leakage and 4.9µW/MHz dynamic power) and is therefore also suitable for mobile applications. As it also provides the best performance per throughput, it is perfectly suitable for *embedded systems*.

If higher throughput is required, ASCON-fast with six unrolled rounds can process more than 13 Gbit/sec at 206 MHz. This is more than sufficient even for 10 gigabit network connections. For *RFID* applications, where size as well as power matter, ASCON-x-low is only 2.57 kGE large and requires as little as 15 µW for a 1 MHz clock source. In an for RFID tags more suitable, power saving 130 nm low-leakage UMC technology, the power consumption is reduced to 4.1 µW.

Energy, the most critical characteristic of *wireless sensor nodes*, is the product of power and runtime. The ASCON-fast implementations require the least amount of energy because of their low runtimes. Even though six unrolled rounds give the best energy results it probably would be more reasonable to use ASCON-fast with one unrolled round for *wireless sensor nodes*.

Table 3. Characteristics of related implementations capable of authenticated encryption

Design	Chip Area	Throughput	Power	Technology
	[kGE]	*[Mbps]*	*[µW/MHz]*	
AES-CCM [6]	3.77	57	5.12	STM 65 nm
AES-OCB2 [6]	5.92	113	8.11	STM 65 nm
AES-ALE [6]	2.70	244	10.55	STM 65 nm
Minalpher [19] low-area	2.81	369	—	
SILC [12] V1	15.70	764	—	
AES-OCB [18]	22.55	854	—	TSMC 90 nm
SILC [12] V2	23.10	2,635	—	
Scream ED [9] 1 Round	6.23	4,577	—	STM 65 nm
Keccak MonkeyDupl. [20]	5.90	4,900	42	
Scream ED [9] 2 Round	8.31	5,190	—	STM 65 nm
Minalpher [19] high-speed	14.32	6,104	—	
Norx [1]	59.00	10,000	—	UMC 180 nm
ICEPOLE [15]	—	41,364	—	FPGA (Xilinx Virtex 6)

5.1 Related Work

A fair comparison of hardware designs is a difficult task. Different designers make diverging assumptions about, e.g., key registers or bus interfaces. Additionally, results are highly dependent on the used manufacturing technology, the used toolchain (e.g., Cadence, Synopsis, Menthor, Xilinx, or Altera), and the external operating conditions (e.g., power supply voltage or ambient temperature). Therefore, the following comparison with related work has to be interpreted carefully.

In Table 3, the hardware results of several authenticated encryption designs are listed. There are standardized AES-based implementations [6, 18] and CAESAR candidates based on sponge constructions [1, 15, 20] and block ciphers [9, 12, 19] depicted. Figure 7 visually combines Tables 2 and 3. The horizontal axis shows the throughput and the vertical axis depicts the area footprint. The dashed equi-efficiency lines indicate a constant throughput per area ratio.

It seems that ASCON provides, together with Keccak MonkeyDuplex [20], excellent performance per area. All AES implementations are a magnitude slower than even the slowest ASCON-fast implementation. Only Norx [1] and ICEPOLE [15] achieve similar or higher performance. However, Norx is more than twice as large (59 kGE) as the largest ASCON design (26 kGE). There are no ASIC results available for ICEPOLE so far. In terms of size, ASCON needs six times fewer registers to store the state and the processed data ($1280+1024$ vs. $320+64$) than ICEPOLE. ICEPOLE S-boxes also have an algebraic degree of four and are thus potentially harder to share than the ASCON S-boxes which probably results in a higher overhead for protected implementations.

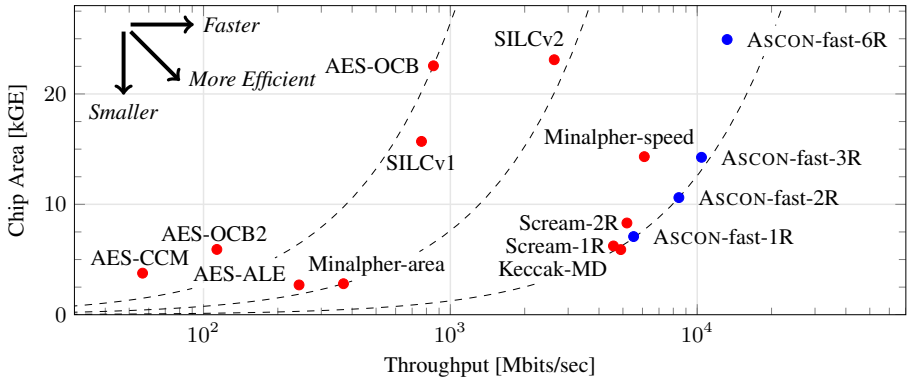


Fig. 7. Throughput versus area comparison

5.2 First-Order DPA Resistant Threshold Implementations of ASCON

Compared to the unprotected versions, the state size intuitively triples for the TI variants ASCON-fast-TI and ASCON-x-low-TI because of the three state shares. Furthermore, also the logic inside the datapath is implemented in triplicate, but the S-box sharing results in a increased resource consumption because of the more complex component functions. This increase in the resource requirements affects the ASCON-fast-TI more than the ASCON-x-low-TI because the latter only uses one S-box instance. As a result, ASCON-fast-TI is 4.0 times larger than ASCON-fast. Nevertheless, it still offers a throughput of 3.77 Gbit/sec. ASCON-x-low-TI, on the other hand, has a comparably low overhead factor of only 3.1.

The authors of this paper are not aware of the existence of any other side-channel protected CAESAR candidates so far to compare ASCON to. Due to its prevalence, a lot of effort of the hardware security community was spent on the question on how to protect AES against DPA attacks. In 2011, Moradi et al. [14] (later improved by Bilgin et al.[5]) implemented a low-area version of AES (encryption only), which is protected by the TI masking scheme. As it turned out, the overhead factor for protecting their AES core against first-order DPA attacks is about 4.6 (2.4 kGE vs. 11 kGE). The major reason for this implementation overhead is the S-box, which is very difficult to protect compared to the S-box of ASCON.

6 Conclusions

In this work we exhaustively evaluated that ASCON is ready for all challenges that today's authenticated encryption applications pose. We demonstrated how to implement high-throughput designs that can process up to 13.2 gigabit of data per second. A design designated for RFID applications requires as little as 2.6 kGE and 15 µW of power at 1 MHz. ASCON-fast also fulfills all requirements of wireless sensor nodes in particular and embedded systems in general. It requires only 7.1 kGE and 33 µJ of energy per byte

and can easily encrypt and authenticate the traffic of an one-gigabit Ethernet connection. As additional contribution, we demonstrated the ease of protecting ASCON against power-analysis attacks. ASCON-fast-TI is a power-analysis protected threshold design that is still able to process 3.6 gigabit of data per second. The comparison with other authenticated-encryption designs showed that ASCON is ready for all challenges the CAESAR competition might hold. And there is still room for improvement: a pipelined ASCON design can process more data at higher frequencies. Such a design can easily be derived from the released Apache-licensed source codes.[2]

Acknowledgements. This work has been supported by the FFG research program SeCoS (project number 836628), the European Commission through the FP7 program (project MATTHEW, project number 610436), and by the Austrian Science Fund (project P26494-N15).

References

1. J. Aumasson, P. Jovanovic, and S. Neves. NORX. Submission to the CAESAR competition: `http://competitions.cr.yp.to/round1/norxv1.pdf`, 2014.
2. D. Bernstein. CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. Available online at `http://competitions.cr.yp.to/caesar.html`, 2014.
3. G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. Keccak Specifications. Submission to NIST (Round 3), 2011.
4. B. Bilgin, J. Daemen, V. Nikov, S. Nikova, V. Rijmen, and G. Van Assche. Efficient and First-Order DPA Resistant Implementations of Keccak. In A. Francillon and P. Rohatgi, editors, *Smart Card Research and Advanced Applications*, Lecture Notes in Computer Science, pages 187–199. Springer International Publishing, 2014.
5. B. Bilgin, B. Gierlichs, S. Nikova, V. Nikov, and V. Rijmen. A More Efficient AES Threshold Implementation. In D. Pointcheval and D. Vergnaud, editors, *Progress in Cryptology - AFRICACRYPT 2014 - 7th International Conference on Cryptology in Africa, Marrakesh, Morocco, May 28-30, 2014. Proceedings*, volume 8469 of *Lecture Notes in Computer Science*, pages 267–284. Springer, 2014.
6. A. Bogdanov, F. Mendel, F. Regazzoni, V. Rijmen, and E. Tischhauser. ALE: AES-Based Lightweight Authenticated Encryption. In S. Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 447–466. Springer, 2013.
7. J. Daemen. Permutation-based Encryption, Authentication and Authenticated Encryption. DIAC – Directions in Authenticated Ciphers, July 2012.
8. C. Dobraunig, M. Eichlseder, F. Mendel, and M. Schläffer. Ascon. Submission to the CAESAR competition: `http://ascon.iaik.tugraz.at`, 2014.
9. V. Grosso, G. Leurent, F. Standaert, K. Varici, F. Durvaux, L. Gaspar, and S. Kerckhof. CAESAR candidate SCREAM Side-Channel Resistant Authenticated Encryption with Masking, 8 2014. DIAC 2014: Directions in Authenticated Ciphers, Santa Barbara, USA [Accessed: 2014 09 30].

[2] http://ascon.iaik.tugraz.at/implementation.html

10. Intel Corporation. Intel Advanced Encryption Standard Instructions (AES-NI), March 2011. Available online: http://software.intel.com/en-us/articles/intel-advanced-encryption-standard-instructions-aes-ni/.
11. Intel Corporation. New Instructions Supporting the Secure Hash Algorithm on Intel Architecture Processors, July 2013. Available online: https://software.intel.com/en-us/articles/intel-sha-extensions.
12. T. Iwata, K. Minematsu, J. Guo, S. Morioka, and E. Kobayashi. SILC: SImple Lightweight CFB, 8 2014. DIAC 2014: Directions in Authenticated Ciphers, Santa Barbara, USA [Accessed: 2014 09 30].
13. P. C. Kocher, J. Jaffe, and B. Jun. Differential Power Analysis. In *Proceedings of the 19th Annual International Cryptology Conference on Advances in Cryptology*, CRYPTO '99, pages 388–397, London, UK, UK, 1999. Springer-Verlag.
14. A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. Pushing the Limits: A Very Compact and a Threshold Implementation of AES. In *Proceedings of the 30th Annual International Conference on Theory and Applications of Cryptographic Techniques: Advances in Cryptology*, EUROCRYPT'11, pages 69–88, Berlin, Heidelberg, 2011. Springer-Verlag.
15. P. Morawiecki, K. Gaj, E. Homsirikamol, K. Matusiewicz, J. Pieprzyk, M. Rogawski, M. Srebrny, and M. Wójcik. ICEPOLE: High-Speed, Hardware-Oriented Authenticated Encryption. In L. Batina and M. Robshaw, editors, *CHES 2014 - 16th International Workshop, Busan, South Korea, September 23-26, 2014. Proceedings*, volume 8731 of *Lecture Notes in Computer Science*, pages 392–413. Springer, 2014.
16. National Institute of Standards and Technology. FIPS PUB 180-4: Secure Hash Standard. Federal Information Processing Standards Publication 180-4, U.S. Department of Commerce, March 2012. Available online: http://www.itl.nist.gov.
17. S. Nikova, C. Rechberger, and V. Rijmen. Threshold Implementations Against Side-channel Attacks and Glitches. In *Proceedings of the 8th International Conference on Information and Communications Security*, ICICS'06, pages 529–545, Berlin, Heidelberg, 2006. Springer-Verlag.
18. M. Parelkar. *Authenticated Encryption in Hardware*. George Mason University, 2005.
19. Y. Sasaki, Y. Todo, K. Aoki, Y. Naito, T. Sugawara, Y. Murakami, M. Matsui, and S. Hirose. Minalpher, 8 2014. DIAC 2014: Directions in Authenticated Ciphers, Santa Barbara, USA [Accessed: 2014 09 30].
20. T. Yalçin and E. B. Kavun. On the Implementation Aspects of Sponge-based Authenticated Encryption for Pervasive Devices. In *Proceedings of the 11th International Conference on Smart Card Research and Advanced Applications*, CARDIS'12, pages 141–157, Berlin, Heidelberg, 2013. Springer-Verlag.

A Threshold Implementation of ASCON's S-box

The efficient implementation of a non-linear function like the S-box of ASCON is not trivial. Quite nicely, ASCON is using an affine transformation of the χ function used in Keccak [3], the new SHA-3 standard. In 2014, Bilgin et al. [4] showed a shared implementation of the Keccak S-box based on three shares with direct-sharing property. This property results in a very compact shared representation of the S-box χ with low hardware costs.

In the following, the same notation as in Bilgin et al.'s paper is used on ASCON. The state shares are represented by the capital letters A, B and C, with the bit index i (modulo 5) inside one of the 64 5-bit state slices with index j.

$$
\begin{aligned}
A'_i &\leftarrow \chi'_i(B,C) \triangleq B_i + (B_{i+1}+1)\,B_{i+2} + B_{i+1}C_{i+2} + B_{i+2}C_{i+1} \\
B'_i &\leftarrow \chi'_i(C,A) \triangleq C_i + (C_{i+1}+1)\,C_{i+2} + C_{i+1}A_{i+2} + C_{i+2}A_{i+1} \\
C'_i &\leftarrow \chi'_i(A,B) \triangleq A_i + (A_{i+1}+1)\,A_{i+2} + A_{i+1}B_{i+2} + A_{i+2}B_{i+1}
\end{aligned} \tag{1}
$$

The shared S-box implementation in Equation 1 fulfills the correctness and the non-completeness properties of TI implementations, but the component functions are not invertible and therefore also not uniform. However, because of the similarities to the Keccak S-box, the findings of Bilgin et al. can also be applied to the ASCON S-box. A straight-forward way to recover the uniformity is the insertion of additional random bits in form of the virtual variables S and P in Equation 2. Even though this ensures the uniformity of the component functions, this would require 640 fresh high-quality random bits per cipher round which is not quite practical!

$$
\begin{aligned}
A'_i &\leftarrow \chi'_i(B,C) + P_i + S_i \\
B'_i &\leftarrow \chi'_i(C,A) + P_i \\
C'_i &\leftarrow \chi'_i(A,B) + S_i
\end{aligned} \tag{2}
$$

Bilgin et al. showed that when only three out of the five S-box output bits are considered, the joint uniformity is already satisfied. Consequently, only the two remaining state slice bits need to be repaired by using Equation 2. Thus, for bit slice positions $i \in \{0,1,2\}$ Equation 1 is sufficient.

The requirement for fresh random bits can be decreased further when the relation between the individual state slices is considered. Since, each sharing of one state slice is independent of the other shared slices, these independent shares can be used to recover the uniformity property of a different state slice sharing. It only needs to be ensured that the independence between the state slices is not violated. Equation 3 can then be applied to all bits of the state slices where the bit index $i \in \{3,4\}$ except the one slice with the state slice index $j = 0$ which requires fresh randomness in order to not violate the independence requirement. The total amount of fresh random bits per cipher round is then reduced to only four bits. The rest of the bits required for the virtual variables is taken from independent bit slices.

$$
\begin{aligned}
A'^{(j)}_i &\leftarrow \chi'_i(B^{(j)}, C^{(j)}) + A^{(j-1)}_i + B^{(j-1)}_i \\
B'^{(j)}_i &\leftarrow \chi'_i(C^{(j)}, A^{(j)}) + A^{(j-1)}_i \\
C'^{(j)}_i &\leftarrow \chi'_i(A^{(j)}, B^{(j)}) + B^{(j-1)}_i
\end{aligned} \tag{3}
$$