NIST Update: ISAP v2.0

Christoph Dobraunig, Maria Eichlseder, Stefan Mangard, Florian Mendel,
Bart Mennink, Robert Primas and Thomas Unterluggauer

https://isap.iaik.tugraz.at
isap@iaik.tugraz.at

30 September 2022

1 Introduction

ISAP [13] is a family of lightweight authenticated encryption algorithms designed with a focus on robustness against implementation attacks. ISAP is of particular interest for applications like firmware updates where robustness against power analysis and fault attacks is crucial and code size and a small footprint in hardware matters. In this short update, we first discuss the implications of new proofs and security arguments on ISAP. Second, we give an overview on implementations of ISAP and how they compare against other authenticated encryption schemes like AES-GCM. After that, we discuss third-party analysis involving ASCON-p and KECCAK-p[400] and its implications on ISAP. Finally, we compare ISAP with current NIST standards and discuss its target applications.

2 Proofs

The mode of ISAP combines various ideas and constructions from the unkeyed sponge and the keyed sponge and duplex. The main workhorse in ISAP is the keyed duplex construction, security of which was proven by Daemen et al. [10]. This generic provable security result concerned black-box security, where the underlying permutations are assumed to be perfectly random and leak-free. Dobraunig and Mennink [16] considered the leakage resilience of the duplex construction, and showed that the full-state keyed duplex is still secure if a limited amount of leakage per duplex call takes place, provided that adversarial state manipulation is restricted wisely. As one of the applications, Dobraunig and Mennink demonstrated that the ISAPENC mode (including the call to ISAPRK) is leakage resilient. In a separate work, Dobraunig and Mennink [18] considered leakage resilience of the suffix sponge, a generalization of ISAPMAC with the function ISAPRK abstracted, in a comparable leakage model as before (tightness of their analysis was discussed by Dobraunig and Mennink [19]). Dobraunig and Mennink [17] united these two works and showed that they, jointly, imply leakage resilience of the ISAP mode.

Leakage resilience and misuse. For more information on the leakage resilience of the mode of ISAP, we refer to the ToSC paper about ISAP v2.0 [13]. We remark that, within this proof, the analysis of authenticity does not rely on uniqueness of the nonce. This means that the security proof in [13] in particular implies that the ISAP mode achieves leakage-resilient authenticity under nonce-reuse. In addition, as the ISAP mode follows the encrypt-than-MAC construction, it by design also achieves security under release of unverified plaintexts. The ISAP mode also guarantees key committing security: decryption of any ciphertext under two different keys will likely fail due to the usage of ISAPRK within ISAPMAC, up to collisions in the output of ISAPRK, or the tag.

Since the previous status report on Isap, there have been two new publications on the generic security of (aspects of) the Isap mode.

1. In [20], Dobraunig and Mennink introduced and analyzed leakage resilient value comparison, and particularly the "PVP" construction. By using this construction, one does not perform plain tag comparison, but rather processes them through a permutation first (as was already suggested in the original specification of Isap [14]). They in particular presented the combined SuKS-then-PVP (StP) construction as it would appear in Isap [20].

2. In [21], Dobraunig et al. proposed a more meaningful tamper and leakage resilience model, i.e., one that is able to more accurately capture the leakage as it occurs in implementations as well as faults. They showcased their model alongside the "asakey" encryption mode that resembles the encryption mode of Isap.

Hash function. In the original specification [14, Section 2.6], we suggested that Isap can be combined with the Ascon hash function. This Ascon hash function follows the sponge construction, and generic security of this construction follows from the indifferentiability of the sponge [3] up to at most $2^{c/2}$ permutation evaluations. This result, in particular, implies that finding collisions, preimage, or second preimages for the construction of Ascon hash is as hard as finding them for a random oracle, provided the attacker can make at most $2^{c/2}$ evaluations. Recently, Lefevre and Mennink [43] noticed that, while this bound is tight for collision and second preimage resistance, it is not tight for preimage resistance, and they derived an improved bound. For the mode of Ascon hash, with parameters $(b, c, r, n) = (320, 256, 64, 256)$, their result implies that it achieves *generic* preimage resistance up to even 2^{192} queries, instead of the former bound of 2^{128} queries implied by the indifferentiability result.

Bounds on characteristics. Several teams have worked on proving improved bounds on characteristics for the Ascon permutation. Erlacher et al. [27] proved bounds on the minimum number of linearly or differentially active S-boxes for 4, 6, 8, 12 rounds using SAT solvers. For the 12-round permutation, any characteristic has differential probability or squared correlation $\leq 2^{-216}$; this bound is likely not tight, but provides ample security margin. For more details, we refer to the Ascon update document. Additionally, Erlacher [26] proved that any collision-producing differential characteristic for the Isap rekeying phase with 1-bit rate and 1 round has probability $\leq 2^{-128}$ and covers 5 or more rounds. The best known characteristic covers 5 such rounds (absorbing 6 bits, 3 of them active) with 105 active S-boxes, i.e., its probability is less than 2^{-210} [26].

3 Implementation Security

In this section, we outline the properties of the Isap mode that make it suitable for cryptographic applications that require strong protection/hardening against implementation attacks including, but not limited to, firmware updates. Isap's protection/hardening against physical attacks like DPA [40], DFA [5], SFA [28], and SIFA [12] has already been discussed in a previous update document [15]. In the context of profiling/SPA attacks, which cannot entirely be covered by Isap's mode-level hardening and need to be addressed on implementation level, there exist several third-party works evaluating Keccak-p[400]-based instances of Isap on low-end, low-noise general-purpose microcontrollers [2, 62, 38]. While such works show that immensely powerful adversaries are still able to conduct side-channel attacks on completely unprotected implementations of Isap with high efforts, they also show that Isap's baseline of defense against physical attacks is generally much

higher compared to other AEAD schemes. Moreover, adding additional hardening against SPA is generally considered to be about an order of magnitude cheaper compared to algorithmic hardening against DPA [2, 65, 30, 32], while SPA hardening may not even be needed on ASIC/FPGA platforms that use a datapath width ≥ 64 bits [58, 8]. So far, submissions to the call for protected implementations by GMU[1] were primarily tested for DPA protection using masking and hence should not be considered to resist profiling/SPA attacks by powerful adversaries on low-noise platforms [9].

In the remainder of this section, we discuss properties of Isap that are not necessarily covered by published protected implementations of other AEAD schemes. In the context of DPA attacks, masked implementations of AEAD schemes typically only provide protection against DPA-based key recovery attacks. However, a device can also be compromised by DPA-style attacks targeting other parts of the algorithm. In the following, we discuss two such attacks and how Isap can provide corresponding protection.

DPA-based Tag Recovery. If the tag comparison operation during authenticated decryption does not offer DPA protection, an attacker may query the decryption with related ciphertexts and perform a DPA-style attack on the leaky tag comparison to forge corresponding valid tags [2]. These tags may then allow the attacker to issue, e.g., a firmware update using a modified firmware image without knowledge about the used encryption key. To address this issue on algorithm-level, one needs to add masking to the tag comparison operation, e.g., using 127 masked AND operations with a multiplicative depth of 7 [2]. In case of Isap, one can make use of the PVP construction [20, 37] to achieve the same effect without noticeable increase of area or code-size.

DPA-based Plaintext Recovery. In case of an online single-pass AEAD scheme, an attacker could query the decryption with a constant nonce and varying ciphertexts, thereby forcing constant key stream blocks that get combined with varying ciphertext blocks [63]. Such an attack does not require direct extraction of cryptographic keys but recovers plaintext messages that may carry, e.g., cryptographic keys or IP in case of firmware updates. To address this issue on algorithm-level, one needs to employ masking whenever key stream blocks get combined with ciphertext blocks – a consideration that is especially relevant for so-called leveled implementations [52]. In the case of Isap, the two-pass construction prevents this scenario by verifying the tag before decryption.

4 Implementations

In this section, we summarize implementation results for Isap. First, we provide a list of the main updates on Isap hardware/software implementations since the publication of the most recent Isap specification in the previous year. We then present performance metrics of Isap on platforms that are particularly relevant for lightweight cryptography such as microcontrollers, co-processors, FPGAs, and ASICs. On other higher-performance platforms such as 64-bit CPUs, and when processing longer messages, Isap generally comes with a runtime increase of around a factor 3 compared to Ascon-128 [25]. We perform our comparisons with the existing NIST standard AES-GCM whenever possible, or alternatively with other AES-based ciphers. Since the Isap mode is designed with a focus on robustness against implementation attacks, and thus features protection/hardening against physical attacks like DPA [40], DFA [5], SFA [28], and SIFA [12], we also include comparisons with AES-based ciphers that feature algorithmic countermeasures against attacks like DPA [40].

[1] https://cryptography.gmu.edu/athena/index.php?id=LWC

4.1 Implementation Updates

The main updates on Isap hardware/software implementations since the publication of the most recent Isap specification include the following:

- Added Isap software implementations that feature more platform-specific optimizations [35].
- Added `opt_32_stp` software implementation [35] that includes a leakage-resilient tag comparison as proposed in [20, 37].
- Software implementations of Isap-A-128a optionally also support Ascon-Hash [36].
- Updated all Isap hardware implementations to LWC Hardware API 1.2.0 [36].
- Added `v1_stp` hardware implementation that includes a leakage-resilient tag comparison as proposed in [20, 37].
- Added `v1_lowlatency` hardware implementation that achieves higher throughput and cuts latency in half by performing 2 permutation rounds per clock cycle [36].
- All hardware implementations of Isap-A-128a also support Ascon-Hash [36].

4.2 Software Results

In the following, we compare the performance of Isap against optimized implementations of AES-CTR and AES-GCM on 32-bit ARM Cortex M4 microprocessors from Schwabe et al. [54] and Mbed TLS [51]. We also include performance numbers of an optimized 2-share implementation of AES-CTR due to a lack of published results for masked AES-GCM implementations.

Generally speaking, as shown in Table 1, the performance of Isap-A-128a roughly matches that of AES-GCM. To get an impression of how much overhead an algorithmic countermeasure against DPA causes for AES-based ciphers, we can have a look at the result for first-order masking of AES-CTR in [54, 55]. There, the runtime for processing large messages using the `2-shares` implementation is reported to be increased by more than a factor of 5 compared to the unprotected `bitsliced` implementation.[2] We additionally point out that the `2-shares` implementation (1) needs to precompute and store all masked round keys to achieve this performance, (2) does not contain any device-specific fixes to counteract potential unintended mask combinations of the microarchitecture and lacks formal verification or a practical evaluation. Hence, more modification and runtime overhead is likely required for this implementation to achieve first-order DPA protection in practice. If one further desires some sort of redundancy against DFA, SFA, or SIFA for the AES-based designs, the runtime is generally expected to be additionally increased by at least another factor of 2.

4.3 Co-Processor Results

In the following, we present performance numbers for Ascon and Isap which can be achieved using a 32-bit RISC-V microprocessor that features a recently proposed compact hardware accelerator for Ascon-p [58]. The accelerator requires only 4.7 kGE, or about half the area of dedicated hardware accelerator designs, and is easy to integrate into low-end embedded devices like 32-bit ARM Cortex-M or RISC-V microprocessors. As can be seen in Table 2, with Isap and Ascon's family of modes for AEAD and hashing, we can perform cryptographic computations with a performance of about 2 cycles/byte, or

[2] For our comparison with AES-CTR from Schwabe et al. [54] we use the most up-to-date (and corrected) performance numbers from the corresponding git repository [55].

Table 1: Software performance metrics on 32-bit ARM Cortex M4 microprocessors when processing x message bytes and 0 bytes of associated data. We indicate protection/hardening against physical side-channel attacks (SCA) or fault injection (FI).

Scheme	Type	SCA	FI	Cycles/Byte		
				64 B	1024 B	long
AES-CTR `bitsliced` [55, 54]	Enc.	✗	✗	165.7	117.2	101.1
AES-CTR `2-shares` [55, 54]	Enc.	✓	✗	-	-	≈ 540
AES-GCM `mbedtls -O3` [51]	AEAD	✗	✗	412.6	201.2	185.9
Isap-A-128a `opt_32_armv67m -O3` [51]	AEAD	✓	✓	614.0	218.2	191.8
Isap-K-128a `opt_32_armv7m -O3` [51]	AEAD	✓	✓	2163.6	461.0	347.5

about 4 cycles/byte if protection/hardening against fault attacks and power analysis is desired. Put differently, the hardware accelerator achieves speed-up factors of about 50 to 80, when compared to corresponding pure software implementations.

Table 2: Performance metrics of a low-end 32-bit RISC-V microprocessor with/without 1-round hardware acceleration for Ascon-p (`Acc.`). We indicate protection/hardening against physical side-channel attacks (SCA) or fault injection (FI).

Scheme	Implementation	SCA	FI	Cycles/Byte			Binary Size
				64 B	1536 B	long	[Bytes]
Ascon-128	C `-O3`	✗	✗	162.0	110.8	106.5	11 716
Ascon-128	C `-Os`	✗	✗	248.5	171.6	168.3	2 104
Ascon-128	ASM + `Acc.`	✗	✗	4.2	2.2	2.1	888
Ascon-Hash	ASM + `Acc.`	✗	✗	4.6	2.6	2.5	484
Isap-A-128a	ASM + `Acc.`	✓	✓	29.1	5.2	4.2	1 844

4.4 FPGA Results

In the following, we compare the performance of Isap to AES-GCM on FPGA platforms. To allow for an easier comparison, all presented performance metrics are derived from 7-series Xilinx FPGA platforms. As can be seen in Table 3, area and performance of unprotected AES-GCM implementations are roughly on par with Isap. To get an impression of how much overhead an algorithmic countermeasure against DPA causes for AES-based ciphers, we can have a look at the result for first-order Threshold Implementations (TI) of AES-GCM in [48]. There, the area is reported to be increased by more than a factor of 4 while the throughput drops by an even larger amount. If one further desires some sort of redundancy against DFA, SFA, or SIFA, one either needs to expect an additional doubling of the area (spatial redundancy) or a further reduction in throughput (temporal redundancy) for the AES-based designs.

4.5 ASIC Results

In the following, we compare the performance of Isap to AES-GCM on ASIC platforms. The presented performance metrics from Aagaard et al. [1] are derived for two different cell libraries and two different synthesizer workflows. As can be seen in Table 4, the performance of unprotected AES-GCM implementations is roughly on par with Isap.

Table 3: FPGA metrics of Isap compared to the NIST standardized AES-GCM mode. We indicate protection/hardening against physical side-channel attacks (SCA) or fault injection (FI).

Scheme	FPGA	SCA	FI	Throughput [Mbit/s]	Throughput /Slices	Area [Slices]
AES-GCM [33]	Artix-7	✗	✗	700	1.78	393
AES-GCM [33]	Artix-7	✗	✗	2 200	2.81	781
AES-GCM TI⋆ [48]	Virtex-7	✓	✗	180	0.05	3 422⋆
Isap-A-128a [36]	Artix-7	✓	✓	1 110	1.78	622
Isap-K-128a [36]	Artix-7	✓	✓	1 560	1.68	924

⋆ Area (Slices) do not include RNG.

In terms of area, implementations of Isap are only about half the size of AES-GCM implementations. For AES-based cipher designs we generally expect a similar increase of runtime/area for algorithmic countermeasures against physical attacks as reported in the previous section on FPGA results. For example, works like [49, 31] report an area increase by about a factor of 4 for 1st-order masked AES designs. In terms of throughput, masked AES hardware designs generally require about 5 to 8 times as many cycles given that this many additional register stages are usually required to correctly implement a masked AES S-box [49, 31]. The difference in throughput between unmasked/masked AES designs may be smaller, e.g., in case of pipelined designs that process AES super boxes sequentially and thus generally do not focus on high throughput or low latency [49, 31]. We again expect an additional doubling of the area (spatial redundancy) or a further reduction in throughput (temporal redundancy) to achieve additional hardening against DFA, SFA, or SIFA for the AES-based designs.

Table 4: ASIC metrics for Isap and AES-GCM when processing x message bytes and 0 bytes of associated data as reported in [1]. We indicate protection/hardening against physical side-channel attacks (SCA) or fault injection (FI).

Scheme	Interface	Cell Lib	Synthesizer	SCA	FI	Throughput [Cycles/Byte]	Area [KGE]
AES-GCM	32-bit	STM_{65nm}	SDC/CE	✗	✗	2.1	53.0
AES-GCM	32-bit	STM_{65nm}	CG/CI	✗	✗	2.1	27.0
AES-GCM	32-bit	$TSMC_{65nm}$	SDC/CE	✗	✗	2.1	25.8
AES-GCM	32-bit	$TSMC_{65nm}$	CG/CI	✗	✗	2.1	26.2
Isap-A-128a	32-bit	STM_{65nm}	SDC/CE	✓	✓	3.2	17.2
Isap-A-128a	32-bit	STM_{65nm}	CG/CI	✓	✓	3.2	12.9
Isap-A-128a	32-bit	$TSMC_{65nm}$	SDC/CE	✓	✓	3.2	11.4
Isap-A-128a	32-bit	$TSMC_{65nm}$	CG/CI	✓	✓	3.2	12.0
Isap-K-128a	16-bit	STM_{65nm}	SDC/CE	✓	✓	2.3	19.6
Isap-K-128a	16-bit	STM_{65nm}	CG/CI	✓	✓	2.3	14.0
Isap-K-128a	16-bit	$TSMC_{65nm}$	SDC/CE	✓	✓	2.3	13.4
Isap-K-128a	16-bit	$TSMC_{65nm}$	CG/CI	✓	✓	2.3	13.0

⋆ SPC = Synopsys Design Compiler vP-2019.03, CE = Cadence Encounter v14.13
CG = Cadence Genus v18.10, CI = Cadence Innovus v18.10

Table 5: Throughput (cycles/byte) of Isap-A-128a in hardware. "Urol" indicates the number of permutation rounds that are executed within one clock cycle.

		MAC (0+x)			AEAD (x+0)		
Algorithm	Urol	64 B	1 536 B	long	64 B	1 536 B	long
Isap-A-128a	1	5.1	1.9	1.8	8.5	3.0	2.8
Isap-A-128a (StP)	1	5.1	1.9	1.8	8.5	3.0	2.8
Isap-A-128a	2	2.8	1.0	1.0	4.6	1.8	1.6
Isap-A-128a ⋆	4	1.6	0.7	0.6	2.7	1.1	1.1

⋆ The performance for Urol = 4 is extrapolated from the other designs.

Finally, we present a more comprehensive overview of the performance of Isap hardware implementations in applications that require high performance/throughput. The numbers in Table 5 are derived from our own hardware implementations [36]. Besides that, a recent work that compares Isap to other (protected) AEAD implementations in term of area/latency in hardware is available at [64].

5 Overview of Third-Party Cryptanalysis

5.1 Ascon-based instances

5.1.1 Mode of Operation

Since Isap-A-128a and Isap-A-128 share the same underlying permutations, it is possible to derive conclusions on the security of Isap-A-128a and Isap-A-128 from third-party analysis of members of the Ascon family. In particular, the results of the authenticated encryption mode of Table 6 can be mapped to the encryption part IsapEnc and the results on Ascon-Hash of Table 7 are relevant for IsapMac.

Table 6: Overview of the best third-party analysis of Ascon achieved by only reducing the number of rounds of their underlying permutation.

Type	Target	Rounds	Time	Method	Reference
Key recovery	Ascon initialization	7 / 12	2^{123}	Cube	[53]
	Ascon initialization	6 / 12	2^{40}	Cube-like	[46]
	Ascon initialization	5 / 12	2^{31}	Diff.-linear	[60]
State recovery	Ascon-128a iteration	3 / 8	2^{117}	Differential	[29]
	Ascon-128a iteration	2 / 8	–	Sat-Solver	[24]

Table 7: Overview of the best third-party analysis of Ascon-Hash.

Type	Target	Output size	Rounds	Time	Method	Reference
Collision	Ascon-Hash	256	2 / 12	2^{125}	Differential	[66]
	Ascon-Hash	256	2 / 12	2^{103}	Differential	[29]

The key recovery attacks targeting Ascon's initialization target the mixing of the key with the nonce by observing the behavior of the first output block for different choices of the nonce. As Table 6 shows, third-party analyses can only exploit this behavior if

the number of rounds are reduced from 12 to 7. If we have a look at the encryption of ISAP, the last absorbed bit of the nonce and the first keystream block are separated by 18 rounds in the case of ISAP-A-128A and 24 rounds in the case of ISAP-A-128. Hence, we conclude that we have a very comfortable security margin against attacks aiming to exploit the mixture of key and nonce via such attack vectors.

As shown in Table 6, the internal state of ASCON-128a can be recovered in a nonce-respecting scenario if the number of rounds of the iteration between two keystream blocks is reduced to 2. For ISAP-A-128A and ISAP-A-128, we use 6 and 12 rounds between extracting two keystream blocks. In addition, the rate is reduced to 64 bits compared to the 128 bits of ASCON-128a. Hence, there is a comfortable security margin against this attack vector.

If we consider ISAPMAC, the analysis of round-reduced ASCON-HASH in Table 7 is of interest. In particular, the analysis shown in Table 7 aims for an internal collision during the absorption of message blocks and is hence directly applicable to round-reduced (from 12 to 2) ISAP-A-128A and ISAP-A-128. Still, there is a comfortable security margin.

5.1.2 Permutation

We want to emphasize that we do not require ideal properties for the underlying permutations of our designs. Non-random properties of the underlying permutations are known and do not affect the claimed security properties of the entire algorithms. Nevertheless, for designing algorithms based on ASCON's permutation, it is necessary to understand its properties. Hence, we give an overview of third-party results regarding ASCON's permutation in Table 8.

Table 8: Overview of the third-party analysis of ASCON's permutation. (👁 = non-black-box distinguisher)

Type	Target	Rounds	Time	Method	Reference
Distinguisher	Permutation	12 / 12	2^{55} 👁	Zero-sum	[34]
	Permutation	8 / 12	2^{46}	Integral	[34]
	Permutation	7 / 12	2^{65}	Integral	[61]
	Permutation	7 / 12	2^{60}	Integral	[53]
	Permutation	7 / 12	2^{34} 👁	Limited-Birthday	[29]
	Permutation	5 / 12	2^{109}	Truncated Differential	[59]
	Permutation	5 / 12	2^{80}	Rectangle	[29]
	Permutation	3 / 12	–	Subspace Trails	[42]

5.2 Keccak-based instances

5.2.1 Mode of Operation

Similar to the ASCON case, we can draw conclusions on the security of ISAP-K-128A and ISAP-K-128 based on the analysis of other encryption and hashing schemes that use KECCAK-p[400] as their underlying permutation.

When looking for cryptanalytic results covering the use of KECCAK-p[400] for confidentiality, it seems that KETJE Sr [4], a participant of CAESAR, drew the most attention. As we can see in Table 9, at most 7 rounds of the initialization of KETJE Sr can be attacked. Although the rate we use in ISAP's encryption with 144 bits is larger than the 32 bits of KETJE Sr, the number of rounds separating the last absorbed bit of the nonce from the first encryption output of 16 for ISAP-K-128A and 24 for ISAP-K-128 is very large. Hence, we do not expect a threat from this attack vector.

Table 9: Overview of third-party analysis of KETJE Sr v1, and KETJE Sr v2 achieved by only reducing the number of rounds of their underlying permutation.

Type	Target	Rounds	Time	Method	Reference
Key recovery	KETJE Sr v1	7 / 13	2^{115}	Cube	[22]
	KETJE Sr v1	7 / 13	2^{91}	Cube	[57]
	KETJE Sr v1	7 / 13	2^{75}	Cube	[45]
	KETJE Sr v2	7 / 13	2^{113}	Cube	[22]
	KETJE Sr v2	7 / 13	2^{99}	Cube	[56]
	KETJE Sr v2	7 / 13	2^{77}	Cube	[45]

When looking for cryptanalytic results covering the use of KECCAK-p[400] in ISAP's MAC, it seems that the results for the Keccak hash function with the KECCAK-p[400] permutation are the most relevant. While for small capacities and output sizes, preimage and collision attacks have been shown for up to 3 and 4 rounds with practical complexity, only 2 rounds of the hash function with a sufficiently large capacity for a meaningful security level could be attacked with high complexity (see Table 10). This result matches with the rates used for ISAP's MAC. Since it also targets internal collisions during the absorption of the message, it is directly applicable to variants of ISAP-K-128A and ISAP-K-128 reduced to 2 rounds. Since ISAP-K-128A uses 16 rounds and ISAP-K-128 uses 20 rounds for the MAC, there is a comfortable security margin.

Table 10: Overview of third-party analysis of hashes based on KECCAK-p[400].

Type	Target	Output size	Rounds	Time	Method	Reference
Preimage	KECCAK[240,160]	80	3 / 20	-	Algebraic	[39, 44]
Collision	KECCAK[240,160]	160	4 / 20	-	Differential	[39, 41]
	KECCAK[144,256]	Arbitrary	2 / 20	2^{101}	Algebraic	[6]

5.2.2 Permutation

We want emphasize that we do not require ideal properties for the underlying permutations of our designs. Non-random properties of the underlying permutations do not affect the claimed security properties of the entire algorithms. In the following, we give an brief overview of third-party results regarding KECCAK-p[400].

Table 11: Overview of the third-party analysis of KECCAK-p[400]. (👁 = non-black-box distinguisher)

Type	Target	Rounds	Time	Method	Reference
Distinguisher	Permutation	20 / 20	2^{396} 👁	Zero-sum	[7]
	Permutation	12 / 20	2^{396}	Integral	[7]
	Permutation	7 / 20	2^{84} 👁	Limited-Birthday	[23]
	Permutation	6 / 20	2^{278}	Differential	[47]

6 Platforms and metrics in which the candidate performs better than current NIST standards

6.1 Authenticated encryption with associated data

In Section 4, we already provide implementation comparisons of variants of Isap with the current widely-used NIST standard AES-GCM [11, 50]. We see that Isap performs already very well compared to unprotected hardware implementations of AES-GCM. Once implementation countermeasures have to be factored in for AES-GCM, we can expect Isap to perform better. In particular, the stronger the attackers one needs to defend against, the more performance advantage one gets over using cryptographic algorithms that solely rely on implementation-level countermeasures like higher-order masking. Since Isap has been designed to be robust even in the presence of active and passive physical adversaries, it is also resilient against implementation mistakes. This means that the authenticity is preserved in case of nonce reuse or release of unverified plaintext (see Section 2).

6.2 Hashing

We do not give a separate definition for a hash function based on Ascon-p and Keccak-p[400] because hash functions based on these permutations are already well-specified [14, Section 2.6]. This is an advantage of permutation-based constructions that we share with SHA-3. However, for Isap, we use much smaller permutations compared to the 1600-bit Keccak permutation, since this leads to a smaller footprint in software and hardware. However, this advantage of flexibility does not stop at a combination of AEAD and hashing. For instance, IsapMac can also be a very efficient stand-alone leakage resilient message authentication code.

7 Target applications and use cases for which the candidate is optimized

Isap strives to perform well in environments where implementation attacks like side-channel and fault attacks are a threat. Especially if hardware support for round-based implementations is available, the resilience against implementation attacks is high. In addition, the overhead that is needed to achieve resilience against implementation attacks is largely independent from the length of the processed data. This means that Isap is especially suited for tasks like in-field firmware updates of lightweight devices, or the protection of stored encrypted data like FPGA bitstreams.

References

[1] M. D. Aagaard and N. Zidaric. “ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process”. In: (accessed: 09/2022). URL: https://eprint.iacr.org/2021/049.pdf.

[2] D. Bellizia, O. Bronchain, G. Cassiers, V. Grosso, C. Guo, C. Momin, O. Pereira, T. Peters, and F.-X. Standaert. “Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography - A Practical Guide Through the Leakage-Resistance Jungle”. In: *CRYPTO 2020*. Ed. by D. Micciancio and T. Ristenpart. Vol. 12170. LNCS. Springer, 2020, pp. 369–400. URL: https://doi.org/10.1007/978-3-030-56784-2_13.

[3] G. Bertoni, J. Daemen, M. Peeters, and G. Van Assche. "On the Indifferentiability of the Sponge Construction". In: *EUROCRYPT 2008*. Ed. by N. P. Smart. Vol. 4965. LNCS. Springer, 2008, pp. 181–197. URL: http://keccak.team/files/SpongeIndifferentiability.pdf.

[4] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer. *Ketje v2*. Submission to Round 3 of the CAESAR competition. 2016. URL: https://competitions.cr.yp.to/round3/ketjev2.pdf.

[5] E. Biham and A. Shamir. "Differential Fault Analysis of Secret Key Cryptosystems". In: *Advances in Cryptology – CRYPTO '97*. Ed. by B. S. Kaliski Jr. Vol. 1294. LNCS. Springer, 1997, pp. 513–525. URL: https://doi.org/10.1007/BFb0052259.

[6] R. H. Boissier, C. Noûs, and Y. Rotella. "Algebraic Collision Attacks on Keccak". In: *IACR Transactions on Symmetric Cryptology* 2021.1 (2021), pp. 239–268. URL: https://doi.org/10.46586/tosc.v2021.i1.239-268.

[7] C. Boura, A. Canteaut, and C. D. Cannière. "Higher-Order Differential Properties of Keccak and *Luffa*". In: *FSE*. Vol. 6733. Lecture Notes in Computer Science. Springer, 2011, pp. 252–269.

[8] R. Breuer, F.-X. Standaert, and I. Levi. "Fully-Digital Randomization Based Side-Channel Security - Toward Ultra-Low Cost-per-Security". In: *IEEE Access* 10 (2022), pp. 68440–68449. URL: https://doi.org/10.1109/ACCESS.2022.3185995.

[9] O. Bronchain and F.-X. Standaert. "Breaking Masked Implementations with Many Shares on 32-bit Software Platforms or When the Security Order Does Not Matter". In: *IACR Trans. Cryptogr. Hardw. Embed. Syst.* 2021.3 (2021), pp. 202–234.

[10] J. Daemen, B. Mennink, and G. Van Assche. "Full-State Keyed Duplex with Built-In Multi-user Support". In: *ASIACRYPT (2)*. Vol. 10625. LNCS. Springer, 2017, pp. 606–637.

[11] J. Daemen and V. Rijmen. *The Design of Rijndael - The Advanced Encryption Standard (AES), Second Edition*. Information Security and Cryptography. Springer, 2020. ISBN: 978-3-662-60768-8. URL: https://doi.org/10.1007/978-3-662-60769-5.

[12] C. Dobraunig, M. Eichlseder, T. Korak, S. Mangard, F. Mendel, and R. Primas. "SIFA: Exploiting Ineffective Fault Inductions on Symmetric Cryptography". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2018.3 (2018), pp. 547–572. URL: https://doi.org/10.13154/tches.v2018.i3.547-572.

[13] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. "Isap v2.0". In: *IACR Transactions on Symmetric Cryptology* 2020.S1 (2020), pp. 390–416. URL: https://doi.org/10.13154/tosc.v2020.iS1.390-416.

[14] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. *Isap v2.0 (Submission to NIST)*. Finalist of NIST lightweight cryptography standardization process. 2021. URL: https://csrc.nist.gov/Projects/Lightweight-Cryptography/.

[15] C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, and T. Unterluggauer. *NIST Update: ISAP v2.0*. (accessed: 09/2022). URL: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/ISAP_update_isap.pdf.

[16] C. Dobraunig and B. Mennink. "Leakage Resilience of the Duplex Construction". In: *ASIACRYPT 2019*. Ed. by S. D. Galbraith and S. Moriai. Vol. 11923. LNCS. Springer, 2019, pp. 225–255. URL: https://doi.org/10.1007/978-3-030-34618-8_8.

[17] C. Dobraunig and B. Mennink. *Leakage Resilience of the ISAP Mode: a Vulgarized Summary*. NIST Lightweight Cryptography Workshop 2019. 2019.

[18] C. Dobraunig and B. Mennink. "Security of the Suffix Keyed Sponge". In: *IACR Transactions on Symmetric Cryptology* 2019.4 (2019), pp. 223–248. URL: https://doi.org/10.13154/tosc.v2019.i4.223-248.

[19] C. Dobraunig and B. Mennink. "Tightness of the Suffix Keyed Sponge Bound". In: *IACR Transactions on Symmetric Cryptology* 2020.4 (2020), pp. 195–212. URL: https://doi.org/10.46586/tosc.v2020.i4.195-212.

[20] C. Dobraunig and B. Mennink. "Leakage Resilient Value Comparison with Application to Message Authentication". In: *EUROCRYPT (2)*. Vol. 12697. LNCS. Springer, 2021, pp. 377–407.

[21] C. Dobraunig, B. Mennink, and R. Primas. "Leakage and Tamper Resilient Permutation-Based Cryptography". In: *IACR Cryptol. ePrint Arch.* (2020). to appear at ACM CCS 2022, p. 200. URL: https://eprint.iacr.org/2020/200.

[22] X. Dong, Z. Li, X. Wang, and L. Qin. "Cube-like Attack on Round-Reduced Initialization of Ketje Sr". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 259–280. URL: https://doi.org/10.13154/tosc.v2017.i1.259-280.

[23] A. Duc, J. Guo, T. Peyrin, and L. Wei. "Unaligned Rebound Attack: Application to Keccak". In: *FSE*. Vol. 7549. LNCS. Springer, 2012, pp. 402–421.

[24] A. D. Dwivedi, M. Klouček, P. Morawiecki, I. Nikolič, J. Pieprzyk, and S. Wójtowicz. "SAT-based Cryptanalysis of Authenticated Ciphers from the CAESAR Competition". In: *SECRYPT ICETE 2017*. Ed. by P. Samarati, M. S. Obaidat, and E. Cabello. SciTePress, 2017, pp. 237–246. eprint: 2016/1053.

[25] eBACS Team. "eBACS: ECRYPT Benchmarking of Cryptographic Systems". In: (accessed: 09/2022). URL: https://bench.cr.yp.to/ebaead.html.

[26] J. Erlacher. *SAT-Supported Linear and Differential Cryptanalysis of Ascon and Related Designs*. Master's thesis. 2022.

[27] J. Erlacher, F. Mendel, and M. Eichlseder. "Bounds for the Security of Ascon against Differential and Linear Cryptanalysis". In: *IACR Transactions on Symmetric Cryptology* 2022.1 (2022), pp. 64–87.

[28] T. Fuhr, É. Jaulmes, V. Lomné, and A. Thillard. "Fault Attacks on AES with Faulty Ciphertexts Only". In: *Workshop on Fault Diagnosis and Tolerance in Cryptography – FDTC 2013*. Ed. by W. Fischer and J. Schmidt. IEEE Computer Society, 2013, pp. 108–118. URL: https://doi.org/10.1109/FDTC.2013.18.

[29] D. Gérault, T. Peyrin, and Q. Q. Tan. "Exploring Differential-Based Distinguishers and Forgeries for ASCON". In: *IACR Transactions on Symmetric Cryptology* 2021.3 (2021), pp. 102–136.

[30] D. Goudarzi and M. Rivain. "How Fast Can Higher-Order Masking Be in Software?" In: *EUROCRYPT (1)*. Vol. 10210. Lecture Notes in Computer Science. 2017, pp. 567–597.

[31] H. Groß, S. Mangard, and T. Korak. "Domain-Oriented Masking: Compact Masked Hardware Implementations with Arbitrary Protection Order". In: *TIS@CCS*. ACM, 2016, p. 3.

[32] H. Groß, S. Mangard, and T. Korak. "An Efficient Side-Channel Protected AES Implementation with Arbitrary Protection Order". In: *CT-RSA*. Vol. 10159. Lecture Notes in Computer Science. Springer, 2017, pp. 95–112.

[33] Helion Technology Limited. "AES-GCM cores". In: (accessed: 09/2022). URL: https://www.heliontech.com/aes_gcm.htm.

[34] K. Hu and T. Peyrin. *Revisiting Higher-Order Differential(-Linear) Attacks from an Algebraic Perspective: Applications to Ascon, Grain v1, Xoodoo, and ChaCha*. NIST LWC Workshop 2022. 2022.

[35] ISAP Team. "ISAP Code Package". In: (accessed: 09/2022). URL: https://github.com/isap-lwc/isap-code-package.

[36] ISAP Team. "ISAP Hardware Package". In: (accessed: 09/2022). URL: https://github.com/isap-lwc/isap-hardware-package.

[37] ISAP Team. "Update on the Performance and Mode-level Properties of ISAP". In: (accessed: 09/2022). URL: https://csrc.nist.gov/csrc/media/Presentations/2022/update-on-the-performance-and-mode-level-propertie/images-media/session-5-mennink-primas-update-isap.pdf.

[38] M. J. Kannwischer, P. Pessl, and R. Primas. "Single-Trace Attacks on Keccak". In: *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2020.3 (2020), pp. 243–268. URL: https://doi.org/10.13154/tches.v2020.i3.243-268.

[39] Keccak Team. "The Keccak Crunchy Crypto Collision and Pre-image Contest". In: (accessed: 09/2022). URL: https://keccak.team/crunchy_contest.html.

[40] P. C. Kocher, J. Jaffe, and B. Jun. "Differential Power Analysis". In: *CRYPTO*. Vol. 1666. LNCS. Springer, 1999, pp. 388–397.

[41] S. Kölbl, F. Mendel, T. Nad, and M. Schläffer. "Differential Cryptanalysis of Keccak Variants". In: *IMACC*. Vol. 8308. LNCS. Springer, 2013, pp. 141–157.

[42] G. Leander, C. Tezcan, and F. Wiemer. "Searching for Subspace Trails and Truncated Differentials". In: *IACR Transactions on Symmetric Cryptology* 2018.1 (2018), pp. 74–100.

[43] C. Lefevre and B. Mennink. *Tight Preimage Resistance of the Sponge Construction*. Cryptology ePrint Archive, Paper 2022/734, to appear in CRYPTO 2022. 2022. URL: https://eprint.iacr.org/2022/734.

[44] T. Li, Y. Sun, M. Liao, and D. Wang. "Preimage Attacks on the Round-reduced Keccak with Cross-linear Structures". In: *IACR Transactions on Symmetric Cryptology* 2017.4 (2017), pp. 39–57.

[45] Z. Li, X. Dong, W. Bi, K. Jia, X. Wang, and W. Meier. "New Conditional Cube Attack on Keccak Keyed Modes". In: *IACR Transactions on Symmetric Cryptology* 2019.2 (2019), pp. 94–124. URL: https://doi.org/10.13154/tosc.v2019.i2.94-124.

[46] Z. Li, X. Dong, and X. Wang. "Conditional Cube Attack on Round-Reduced ASCON". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 175–202. ISSN: 2519-173X. eprint: 2017/160. URL: https://github.com/lizhengcn/Ascon_test.

[47] S. Mella, J. Daemen, and G. V. Assche. "New techniques for trail bounds and application to differential trails in Keccak". In: *IACR Transactions on Symmetric Cryptology* 2017.1 (2017), pp. 329–357.

[48] N. Mentens, V. Miskovsky, M. Novotny, and J. Vliegen. *High-speed Side-channel-protected Encryption and Authentication in Hardware*. Cryptology ePrint Archive, Report 2018/1088. https://eprint.iacr.org/2018/1088. 2018.

[49] A. Moradi, A. Poschmann, S. Ling, C. Paar, and H. Wang. "Pushing the Limits: A Very Compact and a Threshold Implementation of AES". In: *EUROCRYPT*. Vol. 6632. LNCS. Springer, 2011, pp. 69–88.

[50] National Institute of Standards and Technology. *NIST Special Publication 800-38D: Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC*. 2007. URL: https://doi.org/10.6028/NIST.SP.800-38D.

[51] NIST LWC Team. "Lightweight-Cryptography-Benchmarking". In: (accessed: 09/2022). URL: https://github.com/usnistgov/Lightweight-Cryptography-Benchmarking/blob/main/benchmarks/results_nano33ble_aead_all.csv.

[52] O. Pereira, F.-X. Standaert, and S. Vivek. "Leakage-Resilient Authentication and Encryption from Symmetric Cryptographic Primitives". In: *CCS*. ACM, 2015, pp. 96–108.

[53] R. Rohit, K. Hu, S. Sarkar, and S. Sun. "Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon". In: *IACR Transactions of Symmetric Cryptology* 2021.1 (2021), pp. 130–155.

[54] P. Schwabe and K. Stoffelen. "All the AES You Need on Cortex-M3 and M4". In: *SAC 2016*. Vol. 10532. LNCS. Springer, 2016, pp. 180–194.

[55] P. Schwabe, K. Stoffelen, and M. J. Kannwischer. "Fast AES on ARM Cortex-M3 and M4". In: (accessed: 09/2022). URL: https://github.com/Ko-/aes-armcortexm.

[56] L. Song and J. Guo. "Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP". In: *IACR Transactions on Symmetric Cryptology* 2018.3 (2018), pp. 182–214. URL: https://doi.org/10.13154/tosc.v2018.i3.182-214.

[57] L. Song, J. Guo, D. Shi, and S. Ling. "New MILP Modeling: Improved Conditional Cube Attacks on Keccak-Based Constructions". In: *ASIACRYPT 2018*. Ed. by T. Peyrin and S. D. Galbraith. Vol. 11273. LNCS. Springer, 2018, pp. 65–95. URL: https://doi.org/10.1007/978-3-030-03329-3_3.

[58] S. Steinegger and R. Primas. "A Fast and Compact RISC-V Accelerator for Ascon and Friends". In: *CARDIS*. Vol. 12609. LNCS. Springer, 2020, pp. 53–67.

[59] C. Tezcan. "Truncated, Impossible, and Improbable Differential Analysis of Ascon". In: *ICISSP 2016*. Ed. by O. Camp, S. Furnell, and P. Mori. SciTePress, 2016, pp. 325–332. eprint: 2016/490.

[60] C. Tezcan. "Analysis of Ascon, DryGASCON, and Shamash Permutations". In: *International Journal of Information Security Science* 9.3 (2020), pp. 172–187. URL: https://www.ijiss.org/ijiss/index.php/ijiss/article/view/762.

[61] Y. Todo. "Structural Evaluation by Generalized Integral Property". In: *EUROCRYPT 2015*. Ed. by E. Oswald and M. Fischlin. Vol. 9056. LNCS. Springer, 2015, pp. 287–314. eprint: 2015/090.

[62] B. Udvarhelyi, O. Bronchain, and F.-X. Standaert. "Security Analysis of Deterministic Re-keying with Masking and Shuffling: Application to ISAP". In: *COSADE 2021*. Ed. by S. Bhasin and F. D. Santis. Vol. 12910. LNCS. Springer, 2021, pp. 168–183. URL: https://doi.org/10.1007/978-3-030-89915-8_8.

[63] T. Unterluggauer, M. Werner, and S. Mangard. "Side-channel plaintext-recovery attacks on leakage-resilient encryption". In: *DATE*. IEEE, 2017, pp. 1318–1323.

[64] C. Verhamme, G. Cassiers, and F.-X. Standaert. "Analyzing the Leakage Resistance of the NIST's Lightweight Crypto Competition's Finalists". In: *CARDIS*. Lecture Notes in Computer Science. https://perso.uclouvain.be/fstandae/PUBLIS/279b.pdf. Springer, 2022.

[65] N. Veyrat-Charvillon, M. Medwed, S. Kerckhof, and F.-X. Standaert. "Shuffling against Side-Channel Attacks: A Comprehensive Study with Cautionary Note". In: *ASIACRYPT*. Vol. 7658. Lecture Notes in Computer Science. Springer, 2012, pp. 740–757.

[66] R. Zong, X. Dong, and X. Wang. *Collision Attacks on Round-Reduced Gimli-Hash/Ascon-Xof/Ascon-Hash*. IACR Cryptology ePrint Archive, Report 2019/1115. 2019. eprint: 2019/1115.