Check for updates

NIST Internal Report
NIST IR 8454

Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process

Meltem Sönmez Turan
Kerry McKay
Donghoon Chang
Lawrence E. Bassham
Jinkeon Kang
Noah D. Waller
John M. Kelsey
Deukjo Hong

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8454

NIST NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
U.S. DEPARTMENT OF COMMERCE

NIST Internal Report
NIST IR 8454

Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process

Meltem Sönmez Turan
Kerry McKay
Donghoon Chang
Lawrence E. Bassham
Jinkeon Kang
Noah D. Waller
John M. Kelsey
Deukjo Hong
Computer Security Division
Information Technology Laboratory

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8454

June 2023

U.S. Department of Commerce
Gina M. Raimondo, Secretary

National Institute of Standards and Technology
Laurie E. Locascio, NIST Director and Under Secretary of Commerce for Standards and Technology

Certain commercial equipment, instruments, software, or materials, commercial or non-commercial, are identified in this paper in order to specify the experimental procedure adequately. Such identification does not imply recommendation or endorsement of any product or service by NIST, nor does it imply that the materials or equipment identified are necessarily the best available for the purpose.

There may be references in this publication to other publications currently under development by NIST in accordance with its assigned statutory responsibilities. The information in this publication, including concepts and methodologies, may be used by federal agencies even before the completion of such companion publications. Thus, until each publication is completed, current requirements, guidelines, and procedures, where they exist, remain operative. For planning and transition purposes, federal agencies may wish to closely follow the development of these new publications by NIST.

Organizations are encouraged to review all draft publications during public comment periods and provide feedback to NIST. Many NIST cybersecurity publications, other than the ones noted above, are available at https://csrc.nist.gov/publications.

NIST Technical Series Policies
Copyright, Use, and Licensing Statements
NIST Technical Series Publication Identifier Syntax

Publication History
Approved by the NIST Editorial Review Board on 2023-06-03

How to cite this NIST Technical Series Publication:
Sönmez Turan M, McKay K, Chang D, Bassham LE, Kang J, Waller ND, Kelsey JM, Hong D (2023) Status Report on the Final Round of the NIST Lightweight Cryptography Standardization Process. (National Institute of Standards and Technology, Gaithersburg, MD), NIST Interagency or Internal Report (IR) NIST IR 8454. https://doi.org/10.6028/NIST.IR.8454

Author ORCID iDs

Meltem Sönmez Turan:	0000-0002-1950-7130
Kerry McKay:	0000-0002-5956-587X
Donghoon Chang:	0000-0003-1249-2869
Lawrence E. Bassham:	0000-0003-0856-9990
Jinkeon Kang:	0000-0003-2142-8236
Noah D. Waller:	0000-0002-6979-9725
John M. Kelsey:	0000-0002-3427-1744
Deukjo Hong:	0000-0002-0998-2958

Contact Information
lightweight-crypto@nist.gov

All comments are subject to release under the Freedom of Information Act (FOIA).

Abstract

The National Institute of Standards and Technology (NIST) initiated a public standardization process to select one or more schemes that provide Authenticated Encryption with Associated Data (AEAD) and optional hashing functionalities and are suitable for constrained environments. In February 2019, 57 candidates were submitted to NIST for consideration. Among these, 56 were accepted as first-round candidates in April 2019. After four months, NIST selected 32 of the candidates for the second round. In March 2021, NIST announced 10 finalists – namely ASCON, Elephant, GIFT-COFB, Grain-128AEAD, ISAP, PHOTON-Beetle, Romulus, SPARKLE, TinyJAMBU, and Xoodyak – to move forward to the final round of the selection process. On February 7, 2023, NIST announced the decision to standardize the ASCON family for lightweight cryptography applications. This report describes the evaluation criteria and selection process, which is based on public feedback and internal review of the finalists.

Keywords

authenticated encryption; constrained devices; cryptography; hash functions; lightweight cryptography; standardization.

Reports on Computer Systems Technology

The Information Technology Laboratory (ITL) at the National Institute of Standards and Technology (NIST) promotes the U.S. economy and public welfare by providing technical leadership for the Nation's measurement and standards infrastructure. ITL develops tests, test methods, reference data, proof of concept implementations, and technical analyses to advance the development and productive use of information technology. ITL's responsibilities include the development of management, administrative, technical, and physical standards and guidelines for the cost-effective security and privacy of other than national security-related information in federal information systems.

Acknowledgments

NIST thanks the submission teams, who developed and designed the candidates, and the cryptographic community, who analyzed the candidates, shared their comments through the `lwc-forum`, and published papers on various technical aspects of the candidates.

NIST also thanks the developers, who provided optimized implementations of the finalists as well as the hardware and software benchmarking initiatives, for their contribution to the understanding of the performance characteristics of the algorithms on various target platforms.

The authors of this report acknowledge and appreciate contributions from their colleagues at NIST – Lily Chen, Dustin Moody, Andrew Regenscheid, Sara Kerman, Isabel Van Wyk, and Michael Fagan, who provided technical and administrative support and contributed to discussions.

Table of Contents

List of Tables

List of Figures

1. Introduction

The deployment of small computing devices such as RFID tags, industrial controllers, sensor nodes and smart cards is becoming much more common. The shift from desktop computers to small devices brings a wide range of new security and privacy concerns. In many conventional cryptographic standards, the tradeoff between security, performance and resource requirements was optimized for desktop and server environments. As a result, implementing the current cryptography standards (e.g. AES-GCM [1, 2], and SHA-2 [3]) in resource-constrained devices becomes challenging due to the inherent limitations of such devices. When they can be implemented, their performance may not be acceptable.

In 2015, the National Institute of Standards and Technology (NIST) initiated the lightweight cryptography standardization process to select one or more schemes for Authenticated Encryption with Associated Data (AEAD) and optional hashing functionalities that are suitable for use in constrained environments. NIST sought a pairing of AEAD and hashing schemes with shared components in order to reduce implementation size for supporting both functionalities.

In February 2023, NIST announced the decision to standardize the ASCON family for lightweight cryptography applications. The aim of this report is to provide a public record of the third round of the standardization process and explain the evaluation of the finalists to be selected for standardization.

1.1. Background

After hosting two public workshops (in 2015 and 2016), NIST published the submission requirements and evaluation criteria [4] in 2018 and received 57 submissions in response to the call. In April 2019, NIST announced 56 first-round candidates. In August 2019, NIST announced 32 second-round candidates (see Table 1) and published NIST Internal Report (NIST IR) 8268 [5] to explain the evaluation criteria and selection of the second-round candidates.

Table 1. List of second-round candidates

Second-round Candidates
ACE, ASCON, COMET, DryGASCON, Elephant, ESTATE, ForkAE, GIFT-COFB, Gimli, Grain-128AEAD, HyENA, ISAP, KNOT, LOTUS-AEAD and LOCUS-AEAD, mixFeed, ORANGE, Oribatida, PHOTON-Beetle, Pyjamask, Romulus, SAEAES, Saturnin, SKINNY-AEAD, SPARKLE, SPIX, SpoC, Spook, Subterranean 2.0, SUNDAE-GIFT, TinyJAMBU, WAGE, Xoodyak

In August 2020, NIST invited the submitters of the second-round candidates to provide

short updates on their algorithms. During the second round of the process, NIST hosted the third and fourth lightweight cryptography workshops to discuss various aspects of the second-round candidates and obtain valuable feedback for the selection of the finalists. NIST announced the 10 finalists in March 2021 (see Table 2 and 3) and published NIST IR 8369 [6] to explain the selection of the finalists.

Table 2. List of finalists

Finalists
ASCON, Elephant, GIFT-COFB, Grain-128AEAD, ISAP, PHOTON-Beetle, Romulus, SPARKLE, TinyJAMBU, Xoodyak

NIST hosted the fifth workshop (virtual) in May 2022 and received status updates from the designers in September 2022. The timeline of the standardization process is summarized in Table 4.

1.2. Organization

Section 2 provides information about the evaluation criteria and the selection process. Section 3 includes overviews of the finalists, including the design principles, security claims, and summaries of third-party analyses. Section 4 summarizes the software and hardware benchmarking initiatives, including the protected implementations. Section 5 explains the plans for next steps. Appendices include a list of acronyms and detailed results on NIST's internal software benchmarking.

Table 3. The submission teams of the finalists

Finalist	*Team*
ASCON [7–11]	C. Dobraunig, M. Eichlseder, F. Mendel, M. Schläffer
Elephant [12–16]	T. Beyne, Y. Long Chen, C. Dobraunig, B. Mennink
GIFT-COFB [17–21]	S. Banik, A. Chakraborti, T. Iwata, K. Minematsu, M. Nandi, T. Peyrin, Y. Sasaki, S. M. Sim, Y. Todo, A. Inoue [1]
Grain-128AEAD [22–25]	M. Hell, T. Johansson, A. Maximov [2], W. Meier, J. Sönnerup, H. Yoshida
ISAP [26–30]	C. Dobraunig, M. Eichlseder, S. Mangard, F. Mendel, B. Mennink, R. Primas, T. Unterluggauer
PHOTON-Beetle [31–34]	Z. Bao, A. Chakraborti, N. Datta, J. Guo, M. Nandi, T. Peyrin, K. Yasuda
Romulus [35–39]	C. Guo [3], T. Iwata, M. Khairallah, K. Minematsu, T. Peyrin
SPARKLE [40–44]	C. Beierle, A. Biryukov, L. Cardoso dos Santos, J. Großschädl, A. Moradi [4], L. Perrin, A. Rezaei Shahmirzadi [4], A. Udovenko, V. Velichkov, Q. Wang
TinyJAMBU [45–49]	H. Wu, T. Huang
Xoodyak [50–54]	J. Daemen, S. Hoffert, S. Mella [5], M. Peeters, G. Van Assche, R. Van Keer

[1] A. Inoue joined the GIFT-COFB team during the third round.[2] A. Maximov joined the Grain-128AEAD team during the third round. [3] C. Guo joined the Romulus team during the third round. [4] A. Moradi and A. Rezaei Shahmirzadi joined the SPARKLE team during the third round. [5] S. Mella joined the Xoodyak team during the second round.

2. Evaluation Criteria and Selection Process

2.1. Evaluation Criteria

In addition to the submission requirements, the *call for submissions* [4] also listed the evaluation criteria for the standardization process. These criteria were further discussed and clarified during the NIST lightweight cryptography workshops. This section summarizes the evaluation criteria used during the third round of the standardization process.

The cryptographic *security* of the finalists is the most important criterion. The security of the finalists was evaluated based on the analysis available in the submission packages, the security claims of the designers, security proofs, publicly available third-party analysis, and observations. The security evaluation of each finalists are summarized in Section 3. Although not explicitly required by the submission call, there are some additional considerations, such as *nonce-misuse security*, *releasing unverified plaintext (RUP) security*, the *impact of state recovery*, and *post-quantum security* of the candidates.

Another criterion is the *hardware and software performance* of the finalists in constrained environments, which is evaluated and compared in terms of various performance and cost metrics. The finalists are expected to perform significantly better than the current NIST standards for authenticated encryption and hashing, in particular AES-GCM [1, 2] and SHA-2 [3]. Performance comparisons of the finalists are provided in Section 4 and Appendix B.

Resistance to side-channel and *fault attacks* is listed as another criterion. While the submitted implementations were not required to provide side-channel resistance, the ability to provide it easily and at low cost is highly desired. The results on side-channel resistance and fault attacks are provided in Section 4.3.

Intellectual property statements are also part of the evaluation criteria. In principle, NIST does not object to algorithms or implementations that may require the use of a patent claim. However, when technical reasons justify this approach, NIST considers any factors that could hinder adoption in the evaluation process (see Section 2.2).

2.2. Selection Process

Fairly evaluating the finalists and selecting algorithms to be standardized and used long-term was a challenging task. Part of the challenge was due to the variability of the finalists in their functionalities, security claims, underlying building blocks, supported parameter sizes, design approaches, the number of variants (see Table 5 and 6), different amount of third-party security analyses and optimized implementations that were available for consideration.

Since the announcements of the first-round candidates in 2019, the NIST lightweight cryptography team had weekly meetings to discuss the security and the performance of the submissions. The NIST team primarily evaluated candidates based on the submission packages, status updates, publicly available third-party security analysis papers, implementation and benchmarking results, and feedback received during workshops. The `lwc-forum` email forum (with over 750 members) served as an additional venue to receive comments and share ideas. NIST did not consider any other source that was not publicly available during the selection process.

The NIST team published NIST IR 8268 [5] and NIST IR 8369 [6] to explain the selection process for the first and the second rounds of evaluation, respectively. The NIST team also published security analysis papers on some of the candidates (e.g., [58–63]) and performed software benchmarking on microcontrollers (see Appendix B).

Target Applications and Profiles: During the early stages of the standardization process, NIST asked for public feedback on target applications and identified the following two profiles [56]:

- Profile I – AEAD and hashing for constrained software and hardware environments,

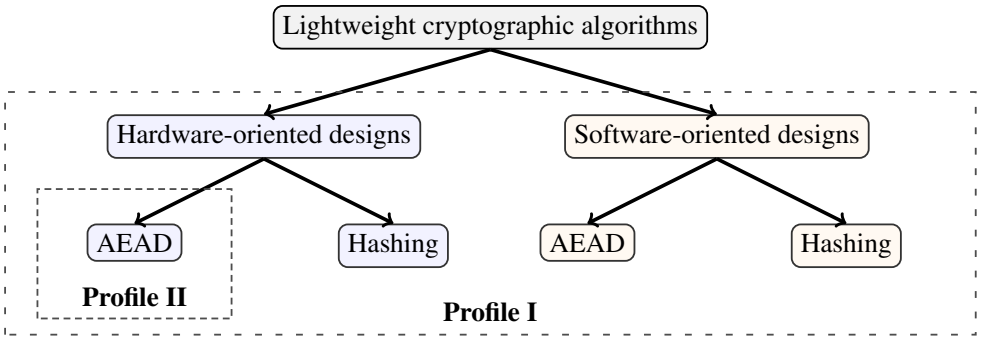


Fig. 1. Profiles for lightweight cryptography applications

and

- Profile II – AEAD for constrained hardware environments,

as shown in Figure 1. Although a single call for algorithms that covered both profiles was published, NIST also considered selecting multiple algorithms for standardization (e.g., one for each profile).

Security Evaluation: The finalists received a large number of third-party security analyses that challenged the correctness of the security claims provided in the submission packages. A summary of the third-party analyses is provided in Section 3. Table 7 provides a list of selected results on the classical security of AEAD variants. None of the publicly available analyses invalidate the claims of the submitters in single-key and nonce-respecting settings, and most candidates have comfortable security margins. [1]

Variants: The submissions were allowed to include multiple variants (maximum of 10 for AEAD and hashing) that support different input/output sizes and/or have different underlying building blocks. NIST asked the teams to identify a *primary* variant for AEAD and hashing with specific input/output sizes so that a fair comparison of the finalists would be possible. Although the submission call only asked for AEAD and hashing functionalities, some of the submissions (e.g., ASCON, SPARKLE, and Xoodyak) also included eXtendable Output Function (XOF) variants, which were not considered as official variants. However, the flexibility of providing a XOF functionality was considered to be an advantage of the design during the selection process.

[1] Note that determining the security margins of the finalists is not straightforward, as some of the finalists have a different number of rounds for different parts of the cipher (e.g., initialization, message/AD processing and finalization) or full-round distinguishers for the underlying components (e.g., permutation) do not necessarily mean that there is no security margin.

Table 4. Timeline of the NIST lightweight cryptography standardization process

Date	***Event***
July 2015	First Lightweight Cryptography Workshop at NIST
October 2016	Second Lightweight Cryptography Workshop at NIST
March 2017	NIST IR 8114 Report on Lightweight Cryptography [55]
April 2017	(draft) Profiles for Lightweight Cryptography Standardization Process [56]
August 2018	Federal Register Notice [57]
	Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process [4]
February 2019	Submission deadline
April 2019	Announcement of the first-round candidates
August 2019	Announcement of the second-round candidates
October 2019	NIST IR 8268, Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process [5]
November 2019	Third Lightweight Cryptography Workshop at NIST
September 2020	Submission deadline for optional status updates
October 2020	Fourth Lightweight Cryptography Workshop (virtual)
March 2021	Announcement of the finalists
July 2021	NIST IR 8369, Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process [6]
May 2022	Fifth Lightweight Cryptography Workshop (virtual)
September 2022	Submission deadline for optional status updates
February 2023	Selection announcement
June 2023	Sixth Lightweight Cryptography Workshop (virtual)

Table 5. Overview of the AEAD variants

Finalists	*Variant*	*Building Block*	*Mode*	*Key size*	*Nonce Size*	*Tag Size*
ASCON	ASCON-128	ASCON Permutation	MonkeyDuplex	128	128	128
	ASCON-128a			128	128	128
	ASCON-80pq			160	128	128
Elephant	Dumbo	Spongent-π[160]	Encrypt-then-MAC	128	96	64
	Jumbo	Spongent-π[176]		128	96	64
	Delirium	KECCAK-f[200]		128	96	128
GIFT-COFB	GIFT-COFB	GIFT-128	Combined Feedback	128	128	128
Grain-128AEAD	Grain-128AEAD	Feedback shift register	Encrypt-and-MAC	128	96	64
ISAP	ISAP-A-128a	ASCON Permutation	Encrypt-then-MAC	128	128	128
	ISAP-K-128a	KECCAK-f[400]		128	128	128
	ISAP-A-128	ASCON Permutation		128	128	128
	ISAP-K-128	KECCAK-f[400]		128	128	128
PHOTON-Beetle	PHOTON-Beetle-AEAD[128]	PHOTON_{256} Permutation	Sponge with Combined Feedback	128	128	128
	PHOTON-Beetle-AEAD[32]			128	128	128
Romulus	Romulus-N	Skinny-128-384+ Tweakable Block Cipher	Combined Feedback	128	128	128
	Romulus-M		MAC-then-Encrypt	128	128	128
	Romulus-T		Encrypt-then-MAC	128	128	128
SPARKLE	SCHWAEMM256-128	SPARKLE_{384}	Sponge with Combined Feedback	128	256	128
	SCHWAEMM128-128	SPARKLE_{256}		128	128	128
	SCHWAEMM192-192	SPARKLE_{384}		192	192	192
	SCHWAEMM256-256	SPARKLE_{512}		256	256	256
TinyJAMBU	TinyJAMBU-128	Keyed Permutation	Sponge	128	96	64
	TinyJAMBU-192			192	96	64
	TinyJAMBU-256			256	96	64
Xoodyak	Xoodyakv1	Xoodoo Permutation	Sponge-variant Cyclist	128	128	128

Table 6. Overview of the hash function variants

Finalists	*Variant*	*Building Block*	*Mode*	*Digest size*
ASCON	ASCON-Hash	ASCON Permutation	Sponge	256
	ASCON-Hasha			256
PHOTON-Beetle	PHOTON-Beetle-Hash[32]	$PHOTON_{256}$ Permutation	Sponge	256
Romulus	Romulus-H	Skinny-128-384+	MDPH[1]	256
SPARKLE	ESCH256	$SPARKLE_{384}$	Sponge	256
	ESCH384	$SPARKLE_{512}$		384
Xoodyak	Xoodyak	Xoodoo Permutation	Sponge	256

[1] MDPH stands for Merkle-Damgård with Permutation using Hirose's DBL compression function [64, 65]

Design Tweaks: In the beginning of the final round, the submitters were allowed to make design modifications (i.e., tweaks) to improve the security or the performance of their candidates. NIST expected these modifications to be relatively minor and not to invalidate previous security analyses. There were no design tweaks for ASCON, GIFT-COFB, ISAP, PHOTON-Beetle, and SPARKLE. New variants were added to the families of ASCON and Romulus, and some of the existing variants of the Romulus family were withdrawn. The assignments of the primary variants were changed for ISAP and SPARKLE. Romulus and Xoodyak were modified to improve their performance. The Elephant design was slightly modified to achieve authenticity under nonce-misuse. The only two finalists that were modified in response to third-party security analysis were Grain-128AEAD and TinyJAMBU.

Performance Benchmarking: The finalists are expected to perform significantly better than current NIST standards, particularly AES-GCM and SHA-2. One of the key elements of lightweight cryptography is the ability for implementors to make trade-offs that best tailor the implementation for a specific use. In such a context, it is essential that not only the fastest (or smallest) implementations are reported, but several metrics are needed to understand the potential for making implementation trade-offs. Therefore, benchmarking efforts provided valuable information to compare the performance of the finalists. For software benchmarking, the finalists ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak showed performance advantages in various platforms, as presented in Section 4.1 and Appendix B. For hardware benchmarking, the best-performing finalists were ASCON, Xoodyak, and TinyJAMBU, as summarized in Section 4.2.

For protected implementations, NIST considered the resistance of the finalists to side-channel and fault attacks, and the implementation overhead needed to mitigate such attacks. The finalists ASCON, ISAP, Xoodyak, and TinyJAMBU demonstrated strong performance, as summarized in Section 4.3.1, which provides insight into the cost of side-channel protection for each of the finalists.

Post-Quantum Security: Although providing security against quantum threats is not one of the main concerns of the lightweight cryptography standardization process, it was also considered during evaluation. In general, most symmetric cryptosystems are considered

Table 7. Selected key recovery, forgery, and distinguishing attacks on the AEAD variants in the nonce-respecting setting

Finalists	*Variant*	*Selected Results*
ASCON	ASCON-128	KR (7 out of 12 rounds) [66]
	ASCON-128a	KR (7 out of 12 rounds) [66]
	ASCON-80pq	KR (7 out of 12 rounds) [66]
Elephant	Dumbo	Distinguisher (40 out of 80 rounds) [67]
	Jumbo	Distinguisher (46 out of 90 rounds) [68]
	Delirium	KR (8 out of 18 rounds) [69]
GIFT-COFB	GIFT-COFB	KR of GIFT-128 (27 out of 40 rounds) [70]
Grain-128AEAD	Grain-128AEAD	KR (192 out of 512 rounds for Initialization) [71]
ISAP	ISAP-A-128a	Forgery (4 out of 12 rounds) [72]
	ISAP-K-128a	Forgery (4 out of 16 rounds) [73]
	ISAP-A-128	Forgery (4 out of 12 rounds) [72]
	ISAP-K-128	Forgery (4 out of 20 rounds) [73]
PHOTON-Beetle	PHOTON-Beetle-AEAD[128]	Distinguisher (10 out of 12 rounds) [74]
	PHOTON-Beetle-AEAD[32]	Distinguisher (10 out of 12 rounds) [74]
Romulus	Romulus-N	RKR of Skinny-128-384+ (32 out of 40 rounds) [75, 76]
	Romulus-M	RKR of Skinny-128-384+ (32 out of 40 rounds) [75, 76]
	Romulus-T	RKR of Skinny-128-384+ (32 out of 40 rounds) [75, 76]
SPARKLE	SCHWAEMM256-128	KR (4.5 out of 11 steps for Initialization) [42]
	SCHWAEMM128-128	KR (4.5 out of 10 steps for Initialization) [42]
	SCHWAEMM192-192	KR (4.5 out of 11 steps for Initialization) [42]
	SCHWAEMM256-256	KR (3.5 out of 8 steps for Message Processing) [42]
TinyJAMBU	TinyJAMBU-128	WKR (476 out of 1024 rounds) [77]
	TinyJAMBU-192	RK-Forgery (full rounds) [78]
	TinyJAMBU-256	RK-Forgery (full rounds) [78]
Xoodyak	Xoodyak	KR (6 out of 12 rounds) [79]

KR: Key Recovery, RK: Related Key, RKR: Related Key Recovery, WKR: Weak Key Recovery

to be relatively secure against quantum threats. The best generic attack against symmetric ciphers is Grover's algorithm [80], which provides a quadratic speedup for exhaustive key search (or finding collisions in hash functions). To avoid the attack, variants with larger key sizes (or larger digest sizes) are preferred. Among the finalists, three of the candidates supported keys longer than 128 bits. In particular, the SPARKLE and TinyJAMBU families included AEAD variants with 192-bit and 256-bit keys and one ASCON variant supported 160-bit keys. Note that, due to the requirement of running Grover's algorithm with sequential queries, the practical implications of the attack may be limited. Additionally, there are some results that exploit the internal structure of symmetric ciphers, particularly the Even-Mansour construction [81, 82], which may impact the quantum security of Elephant.

Intellectual Property Statements: The initial call for submissions [4] stated the goal of worldwide, royalty-free availability for selected algorithms. NIST required that algorithm submitters identify all known intellectual property that could be infringed by implementing their candidate algorithm. Among the finalists, applicable patents were only identified for PHOTON-Beetle [31]. After the review process was completed, intellectual property considerations did not factor into decisions made during the selection process.

2.2.1. Selection of ASCON

After evaluating the finalists according to the criteria presented above, NIST has selected the ASCON family for standardization.

The ASCON family includes AEAD and hash functions, as well as additional XOFs. This allows it to satisfy a wide range of application needs and there is low additional cost to implement additional functionalities thanks to its permutation-based design.

ASCON is the most mature of the finalists in terms of security. While some of the other finalists were not published prior to the lightweight standardization process, the AEAD variants of the ASCON family had already been presented and analyzed as part of the CAESAR competition.[2] Three profiles were created during the competition, including one for lightweight authenticated encryption. Ultimately, the AEAD variants of ASCON were selected as the primary choice for lightweight applications in the final CAESAR portfolio. ASCON's maturity can also be seen in the tweaks for the final round, where there were additional variants added but none of the second-round variants were modified. This is in contrast to some other finalists that included design tweaks to address attacks.

With ASCON's long history comes a wealth of analyses. It was the submission with the most third-party analysis and implementations. Despite the head-start on cryptanalytical attacks, ASCON has remained strong. AEAD variants of the ASCON family provide a high

[2]The Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) was organized by an international cryptologic research community to identify a portfolio of authenticated encryption schemes that offer advantages over AES-GCM and are suitable for widespread adoption. The final portfolio of the competition was announced in February 2019. More information is available at https://competitions.cr.yp.to/caesar.html.

security margin in the nonce-respecting setting and also provide high integrity assurances in the nonce-misuse setting. Additionally, the AEAD mode provides a mode-level protection mechanism for security against leakage.

Performance in constrained environments, such as dedicated hardware and embedded systems, was a significant factor in the decision. ASCON performed very well in hardware and software, demonstrated implementation flexibility supporting various trade-offs between cost and performance, and showed performance advantages over current NIST AEAD and hash standards in a variety of hardware and software platforms with limited resources. ASCON was also shown to incur a lower additional cost for protected implementations over unprotected ones.

Another finalist, ISAP, also had two AEAD variants that relied on the ASCON permutation. It was ultimately deemed less flexible than ASCON, as its mode-level leakage resistance caused implementations to be larger and slower.

One important limitation of the ASCON variants studied in the lightweight standardization process is the lack of an option for 256-bit keys. This can be an issue when 128-bit security against quantum attacks is needed. However, NIST emphasizes that the main purpose of this selection process was for lightweight AEAD and hashing. When post-quantum security and 256-bit keys are required, AES-GCM can be used. NIST may also consider additional variants providing higher post-quantum security at a later date.

NIST believes that the ASCON family will provide sufficient security in target environments for the foreseeable future. Further, NIST has decided that a secondary algorithm is not necessary at this time as the performance of ASCON is expected to be acceptable for target devices and applications.

3. Finalists

This section provides an overview of each finalist and focuses on their design principles, security claims, and summaries of third-party analyses (See also [83, 84]).

3.1. ASCON

3.1.1. Overview of the Design

ASCON [9] is a permutation-based AEAD and a hashing scheme. The main component of the ASCON family is a 320-bit permutation instantiated with different constants and number of rounds for different variants. ASCON-AEAD uses the monkeyDuplex construction [85] with additional key additions during initialization and finalization, whereas ASCON-Hash uses the sponge construction [86]. The ASCON family [87], including ASCON-128 and ASCON-128a, was selected as the primary choice for lightweight authenticated encryption in the final portfolio of the CAESAR competition.

Submission updates. A new hash function – ASCON-Hasha – and an extendable output function – ASCON-Xofa – were added to the ASCON family in the final round.

Variants. The AEAD variants of ASCON are provided below. The number of rounds for each AEAD variant is represented as a 3-tuple, which corresponds to the number of rounds during initialization, Associated Data (AD) and message processing, and finalization, respectively. Ascon team also defined ASCON-80pq to provide stronger resistance against quantum key recovery attacks.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*Block size* (in bits)	*#Rounds*
ASCON-128	128	128	128	64	12/6/12
ASCON-128a	128	128	128	128	12/8/12
ASCON-80pq	160	128	128	64	12/6/12

The hash and XOF variants of ASCON are provided below. The number of rounds for hashing variants is represented as a 4-tuple, which corresponds to the number of rounds during initialization, absorbing message, squeezing the first block, and squeezing the remaining blocks, respectively.

Hash variants	*Digest size* (in bits)	*Rate* (in bits)	*Capacity* (in bits)	*#Rounds*
ASCON-Hash	256	64	256	12/12/12/12
ASCON-Hasha	256	64	256	12/8/12/8
XOF variants				
ASCON-Xof	any	64	256	12/12/12/12
ASCON-Xofa	any	64	256	12/8/12/8

Security Claims. Submitters made the following security claims:

- All three ASCON AEAD variants provide 128-bit security for the confidentiality of plaintext and the integrity of plaintext, AD, and nonce in the nonce-respecting setting, where the number of processed plaintext and AD blocks protected by the encryption algorithm is limited to a total of 2^{64} blocks per key.
- ASCON-Hash and ASCON-Hasha provide 128-bit security against collision attacks and (second) pre-image attacks. ASCON-Xof and ASCON-Xofa with an output size of ℓ bits provide $\min(128, \ell/2)$-bit security against collision attacks and $\min(128, \ell)$-bit security against (second) pre-image attacks.

3.1.2. Security Analysis

The ASCON family has received a significant amount of third-party security analysis. A summary of the results is provided below (also, see [11]).

The following papers studied the security of the AEAD variants in the nonce-respecting setting.

- Rohit and Sarkar [88] presented a weak-key recovery cube-like attack and a weak-key distinguisher on 7-round ASCON initialization in the nonce-respecting setting with a time complexity of 2^{97} and a data complexity of 2^{64} for $2^{116.34}$ keys and with a time complexity of 2^{33} and a data complexity of 2^{33} for 2^{63} keys, respectively.
- Tezcan [89] presented a differential-linear key-recovery attack on 4-round ASCON initialization with bias 2^{-15}.
- Tezcan [90] provided differential-linear key-recovery attacks on 4-round and 5-round ASCON initialization with a time complexity of 2^{15} and $2^{31.44}$, respectively.
- Li et al. [91] presented a key-recovery attack on 7-round ASCON-128 and a weak-key-recovery attack on ASCON-128a with time complexity $2^{103.9}$ and 2^{77}, respectively, where the size of the weak-key class is 2^{117}.
- Dobraunig et al. [92] presented a key recovery cube-like attack on 6-round ASCON initialization with a time complexity of 2^{66}, and a forgery attack (using a differential char-

acteristic) on 4-round ASCON finalization with a time complexity of 2^{101}.

- Tezcan [93] presented truncated and improbable differential distinguishers on 5-round ASCON-permutation with a data complexity of 2^{109} and a 5-round impossible differential distinguisher on it with a data complexity of 2^{256}.
- Dwivedi et al. [94] presented a state recovery SAT-based attack on ASCON-128a with 2-round permutation during the encryption phase with a time complexity of 2^{32}.
- Gérault et al. [95] presented new forgery attacks on four rounds of the finalization of ASCON-128 with a data complexity of $2^{96.61}$ and three rounds of the finalization of ASCON-128a with a data complexity of 2^{20}. They also presented a state recovery attack on ASCON-128a with 3-round permutation during the encryption phase with a time complexity of 2^{117}.
- Hu and Peyrin [96] presented a conditional higher-order differential-linear attack on 6-round ASCON initialization with time and data complexities of 2^{74}.
- Rohit et al. [66] presented a key recovery cube attack on 7-round ASCON in the nonce-respecting setting with a data complexity of 2^{64} and a time complexity of 2^{123}.
- Liu et al. [97] presented a differential-linear key-recovery attack on 5-round ASCON-128 initialization with time complexity 2^{26} and data complexity 2^{26}. They mentioned that this attack is also applicable to ASCON-128a.
- Halak et al. [98] described how they could insert a hardware Trojan to reduce the number of rounds from 12 to five during the initialization phase and perform a cube attack with a time complexity of 2^{24}.

The following papers studied the security of the AEAD variants in the nonce-misuse setting.

- Chang et al. [62] presented a key-recovery conditional-cube attack and a state-recovery conditional-cube attack on ASCON-128a with 7-round permutation during the encryption phase with a time complexity of 2^{118} and a data complexity of 2^{117}.
- Li et al. [99] presented a key recovery cube-like attack on 7-round ASCON initialization with a time complexity of 2^{97}. Authors also presented a forgery attack (using cube tester) on 6-round ASCON-128 finalization with a time complexity of 2^{33} and a state recovery cube-like attack on ASCON-128 with 6-round permutation during the encryption phase with a time complexity of 2^{66}.
- Chang et al. [58] presented a key-recovery conditional-cube attack on ASCON-80pq with 6-round permutation during the encryption phase with a time complexity of 2^{130} and a data complexity of $2^{44.8}$. They also presented a 192-bit partial-state recovery conditional-cube attack on ASCON-128 with 6-round permutation during the encryption phase with a time complexity of $2^{44.8}$ and a data complexity of $2^{44.8}$.

- Baudrin et al. [100] presented a state-recovery conditional-cube attack on ASCON-128 with 6-round permutation during the encryption phase in the nonce-misuse setting with a time complexity of about 2^{40} and a data complexity of about 2^{40}.

The following papers studied the security of the underlying permutation against some distinguishing attacks.

- Dobraunig et al. [92] presented a zero-sum distinguisher on 12-round ASCON permutation with a time complexity of 2^{130}.
- Todo [101] presented integral distinguishers on r-round ASCON-permutation with 2^{65}, 2^{130}, 2^{258}, 2^{300}, and 2^{315} chosen plaintext for $r = 7, \ldots, 11$, respectively.
- Baksi et al. [102] found two all-in-one differential distinguishers of 3-round ASCON-permutation by using machine learning with 2^{19} training data.
- Dobraunig et al. [103] presented a heuristic tool for finding linear characteristics and showed that the minimum number of active S-boxes for 5-round ASCON-permutation without any restriction is 67 with bias 2^{-94}, and the minimum number of active S-boxes for 4-round ASCON-permutation with a restriction that active mask bits have to be in the outer (rate) part of the state is 61 with bias 2^{-83}.
- Leander et al. [104] provided generic algorithms that search subspace trails for SPN ciphers and permutation ciphers and were applied to ASCON-permutation with three rounds covered for encryption with 298 dimension and one round covered for decryption with dimension 125.
- Gérault et al. [95] found non-black-box limited-birthday distinguishers for the 7-round ASCON-permutation with time complexity 2^{34}.
- Hu and Peyrin [96] presented a zero-sum distinguisher on 12-round ASCON-permutation with a time complexity of 2^{55} and a higher-order differential on the 8-round permutation with a time complexity of 2^{46}.
- Rohit et al. [66] presented a division-property-based distinguisher on 7-round ASCON-permutation with a data complexity of 2^{60} and a time complexity of 2^{60}.
- Erlacher et al. [105] proved that any single characteristic on a 4-round ASCON permutation has a differential probability or squared correlation of at most 2^{-72}, six rounds at most 2^{-108}, eight rounds at most 2^{-144}, and 12 rounds at most 2^{-216}.
- Hirch et al. [106] presented a dedicated tool for trail search in ASCON and proved bounds beyond 2^{-128} for six rounds and beyond 2^{-256} for 12 rounds of both differential and linear trails.
- Sommervoll [107] proposed the *phantom gradient attack*, which replaces discrete operations with differentiable functions and represents a target cipher as a neural network in

order to recover a secret key. The author applied the technique to individual operations in ASCON.

Results on ASCON-Hash. The following papers studied the collision resistance of the hash function variants.

- Gérault et al. [95] provided a collision attack on 2-round ASCON-Hash and ASCON-Hasha with a time complexity of 2^{103}.
- Dobraunig et al. [108] found a practical semi-free-start collision for four rounds of ASCON-Hash and ASCON-Xof. They also considered preimage attacks on 2-round and 3-round ASCON-Xof when the hash value is truncated to 64 bits.
- Zong et al. [109] provided collision attacks on 2-round ASCON-Xof, whose output size is 64 bits, with a time complexity of 2^{15} and 2-round ASCON-Hash with a time complexity of 2^{125}. However, Yu et al. [110] reported that one of the 2-round differential characteristic used in [109] is invalid.

The following papers studied the preimage resistance of the hash function variants:

- Qin et al. [72] introduced bit-level MILP-based automatic tools and gave a preimage attack on 4-round ASCON-Xof, whose output size is 128 bits, with time $2^{126.4}$ and memory 2^{45}.
- Lefevre et al. [111] proved that ASCON-Hash can have 192-bit preimage security with an assumption that ASCON-permutation is ideal.

Results in the quantum setting. Lee et al. [112] provided estimated quantum resources for a quantum preimage attack on ASCON-Hash.

Security Margin. None of the existing security analyses violate the security claims of the submitters. The best key-recovery attacks are on the AEAD variants of ASCON with 7-round (out of 12) initialization [66]. The best attack on hash variants of ASCON is a preimage attack covering 4 rounds. Considering these results, ASCON family has a high security margin.

3.2. Elephant

3.2.1. Overview of the Design

Elephant [14] is a permutation-based AEAD scheme that follows a nonce-based encrypt-then-MAC construction in which encryption is done using counter mode and authentication using a variant of the protected counter sum. Elephant is the only finalist that is based on a parallel mode, which is instantiated using either Spongent [113] or KECCAK [114] permutations.

Submission updates. In the final round, the mode was modified from the Wegman-Carter-Shoup MAC to a protected counter sum MAC to achieve authenticity under nonce-misuse.

Variants. The AEAD variants of the Elephant family are listed below.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*Block size* (in bits)	*Permutation*	*#Rounds*
Dumbo	128	96	64	160	160-bit Spongent	80
Jumbo	128	96	64	176	176-bit Spongent	90
Delirium	128	96	128	200	KECCAK-f[200]	18

Security Claims. Submitters claim that Dumbo, Jumbo, and Delirium provide 112-bit, 127-bit, and 127-bit security in the nonce-respecting setting with data limits of $2^{45.68}$, $2^{45.54}$, and $2^{45.36}$ blocks per key, respectively.

3.2.2. Security Analysis

The third-party analyses on Elephant are summarized below.

- Zhou et al. [69] presented an interpolation key-recovery attack on 8-round (out of 18) Delirium in the nonce-respecting setting with 2^{70} data complexity, $2^{98.3}$ XOR operations, and 2^{70} memory complexity.
- Vialar [115] presented an efficient side-channel key recovery attack against Dumbo by using correlation power analysis on the first round of the Spongent permutation during the absorption of the first block of associated data.
- Beyne et al. [116] proved the multi-user security of Elephant v2 under the assumption that the keys of all instances are mutually independent and that the underlying permutation is random. They also showed that Elephant v2 ensures authenticity under nonce misuse.

Results on Spongent permutations. The following papers studied the security of Spongent permutations.

- Bogdanov et al. [67] presented differential distinguishers on the 160-bit Spongent permutation covering 40 rounds (out of 80) with probability 2^{-160}, the 176-bit Spongent permutation over 44 rounds (out of 90) with probability 2^{-176}, linear distinguishers on the 160-bit Spongent permutation over 80 rounds (out of 80) with correlation 2^{-160}, and the 176-bit Spongentpermutation over 90 rounds (out of 90) with correlation 2^{-180}.
- Zhang and Liu [68] presented a truncated-differential distinguisher on 176-bit Spongent-permutation of 46-round (out of 90) with probability $2^{-174.415}$.
- Sun et al. [117] presented a zero-sum distinguishing attack on 176-bit Spongent permutation of 21-round (out of 90) with 2^{159} time complexity.

Results on KECCAK permutations. The underlying permutation of SHA-3 hash functions [114], namely KECCAK-f[1600], has received a significant amount of third-party analysis (e.g., [118–137]) Some of these results also apply to the KECCAK permutation with smaller sizes. The best preimage and collision attacks on KECCAK and KECCAK variants cover up to four rounds [124–131] and six rounds [132–135], respectively. There are polynomial enumeration method-based (second) preimage attacks on 6-round, 7-round, 8-round, and 9-round KECCAK that require complexity close to brute force [136, 137]. There are zero-sum distinguishers on full 24-round KECCAK-f[1600] with very high complexities [138–141]. When the complexity is bounded by 2^{200}, zero-sum distinguishing attacks on KECCAK-f[1600] are known up to 14 rounds with 9-round forward and 5-round backward directions [125].

The KECCAK crunchy crypto collision and preimage contest [73] lists practical preimage (up to one round by Boyar and Peralta) and collision results (up to two rounds by Westfeld) on KECCAK-f[200], where capacity is 160-bit.

Results in the quantum setting. A summary of results in quantum setting is provided below.

- Bonnetain and Jaques [142] presented a quantum circuit for an implementation of the offline Simon's algorithm [143] with $O(n/3)$ classical queries and $O(n/3)$ quantum time, and estimated its cost to recover a key of Elephant, where n is the block size.
- Alagic et al. [144] provided a post-quantum security proof for a variant of Elephant mode, in the public random permutation model, showing that the offline Simon's algorithm matches the upper bound of the security proof.
- Shi et al. [145] proposed a quantum key recovery attack on Elephant in a quantum setting, where the adversary is allowed to make superposition queries to the encryption or decryption.

Security Margin. None of the existing security analyses violate the security claims of the submitters. The best key-recovery attack on Delirium (based on the 200-bit KECCAK permutation) covers eight rounds (out of 18) [69]. There are no dedicated cryptanalysis results on Dumbo (based on the 160-bit Spongent permutation) or Jumbo (based on the 176-bit

Spongent permutation). Instead, there is a differential distinguisher [67] on 40 rounds (out of 80) of the 160-bit Spongent permutation with probability 2^{-160} and a truncated-differential distinguisher [68] on 46 rounds (out of 90) of the 176-bit Spongent permutation with probability $2^{-174.415}$. Considering these results, Elephant variants have around 50% of security margin.

3.3. GIFT-COFB

3.3.1. Overview of the Design

GIFT-COFB is a block-cipher based AEAD scheme, where the underlying block cipher is GIFT-128 [146, 147] and the mode is a variation of COFB (COmbined FeedBack) mode, which was introduced in CHES 2017 [148, 149].

Submission updates. The GIFT-COFB team did not propose a design tweak.

Variants. The single variant of the GIFT-COFB family is listed below. The underlying block cipher `GIFT-128` consists of 40 rounds.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*#Rounds*
GIFT-COFB	128	128	128	40

Although the GIFT-COFB submission does not include an official hash function variant, the designers proposed constructing a 256-bit hash functions using GIFT-128 in the double-block-length hashing developed by Mennink [150] if hashing functionality is desired.

Security Claims. The submitters claimed that GIFT-COFB has 64-bit IND-CPA security (privacy) and 58-bit INT-CTXT security (authenticity) in the nonce-misuse setting.

3.3.2. Security Analysis

The underlying block cipher GIFT has received a large number of third-party analyses. An extended list of third-party analyses on GIFT-128 is provided in [21].

- Cao and Zhang [151] presented two related-key differential characteristics for the 7-round and 10-round GIFT-128 with probabilities of $2^{-15.83}$ and $2^{-72.66}$, respectively.
- Cui et al. [152] presented a linear attack on 20-round GIFT-128 with time complexity $2^{112.28}$ using a 16-round linear characteristic with correlation 2^{-62}.
- Eskandari et al. [153] used their tool to find an integral distinguisher on 11-round GIFT-128 with data complexity 2^{127}.
- Ji et al. [154] improved Matsui's branch and bound search algorithm and applied it to the GIFT family. Their algorithm found the best differential trails for GIFT-128 up to 19 rounds and the best linear trails for GIFT-128 up to 10 rounds and GIFT-64 up to 15 rounds.
- Ji et al. [155] presented three attacks: (1) a related-key boomerang attack on 22-round GIFT-128 with time $2^{112.63}$, data $2^{112.63}$, and memory 2^{52}; (2) a related-key rectangle attack on 23-round GIFT-128 with time $2^{126.89}$, data $2^{121.31}$, and memory $2^{121.63}$; and (3)

a differential attack on 26-round GIFT-128 with time $2^{123.245}$, data $2^{123.245}$, and memory 2^{109}.

- Khalesi and Ahmadian [156] showed that the minimum data complexity of the integral distinguisher on 11-round GIFT-128 is 2^{127} and confirmed that 11-round distinguisher found by Eskandari et al. [153] has the minimum data complexity.
- Li et al. [157] presented a differential attack on 26-round GIFT-128 with time $2^{124.415}$, data 2^{109}, and memory 2^{109} using a 20-round differential characteristic with probability $2^{-121.415}$.
- Liu et al. [158] proposed general STP-based models searching for differential and linear trails and found differential trails of 9-round, 10-round, 11-round, 12-round, 13-round, 18-round, and 21-round GIFT-128 with probability $2^{-45.4}$, $2^{-49.4}$, $2^{-54.4}$, $2^{-60.4}$, $2^{-67.8}$, $2^{-103.4}$, and $2^{-126.4}$, respectively.
- Liu and Sasaki [159] presented a related-key boomerang attack on 21-round GIFT-128 with time $2^{126.6}$, data $2^{126.6}$, and memory $2^{126.6}$ using a 19-round boomerang distinguisher with probability $2^{-121.2}$.
- Sun et al. [160] presented a linear cryptanlaysis of GIFT-COFB by using a SAT-based trail search technique that allows a key recovery attack on GIFT-COFB with 16-round GIFT-128. They also presented a 24-round key recovery attack on GIFT-128 with a 19-round linear approximation.
- Sun et al. [161] presented a linear attack on 25-round GIFT-128 with time $2^{124.75}$, data $2^{126.77}$, and memory 2^{96} by appending one more round after a new 19-round linear approximation that they discovered.
- Zhu et al. [162] presented a differential attack on 22-round GIFT-128 with time 2^{114}, data 2^{114}, and memory 2^{53}.
- Zong et al. [70] presented a two-step strategy to search for advantageous distinguishers. They used 20-round differentials to give the differential attack on 27 rounds of GIFT-128 and two 17-round linear trails to give the linear hull attack on 22 rounds of GIFT-128. They also presented the linear cryptanalysis of GIFT-COFB with 15-round GIFT-128 using a 9-round linear trail.
- Hu et al. [163] showed that there is no impossible differential for 8-round GIFT-128 with patterns that have an active superbox in the plaintext and ciphertext.
- Baksi [164] presented the optimal linear bounds for 11-round and 12-round GIFT-128, extending from the best-known result of 10 rounds.
- Sun et al. [165] showed that 22 rounds for GIFT-128 are required to prevent efficient DC/LC trails, which was reconfirmed by Kim et al. [166].

- Bijwe et al. [167] provided the implementation costs of Grover's key search algorithm for GIFT-64 and GIFT-128.

The following results study the security of the mode of GIFT-COFB and its related variants:

- Inoue et al. [168] presented an attack on GIFT-COFB using q_e encryption queries and no decryption query to break privacy (IND-CPA). The success probability is $O(q_e/2^{n/2})$ for n-bit block, while the claimed bound contains $O(q_e^2/2^n)$. It does not invalidate the 64-bit IND-CPA claim of GIFT-COFB.
- Inoue and Minematsu [169] presented a forgery attack on GIFT-COFB using $2^{n/2}$ encryption queries and a single decryption query. It shows the tightness of the provable security bounds of GIFT-COFB.
- Khairallah [170] presented the IND-CCA attack with complexity $2^{n/2}$ and the forgery attack with complexity $2^{n/2}$ to operate with a single encryption query.
- Khairallah [171] presented a forgery attack on GIFT-COFB mode using a mask collision. The attack requires $O(2^{n/4})$ encryption queries and $O(2^n)$ decryption queries. If the adversary can guess the colliding pair of masks successfully, it leads to successful forgery with probability 1. However, the success probability of such a guess is $2^{-n/2}$.
- Rajan et al. [172] presented distinguishing attacks on GIFT-COFB reduced to two to six rounds by building a multi-layer perceptron network.
- Inoue et al. [173] showed that GIFT-COFB has 32-bit security for both privacy and authenticity of nonce-misuse resilience.

Results in the quantum setting. A summary of results in quantum setting is provided below.

- Bijwe et al. [174] presented the quantum circuit for the GIFT family and the precise cost estimate for quantum key search attacks in both the gate count and depth-times-width cost metrics. They implemented the full Grover oracles for the GIFT family in Q# quantum programming language for unit tests and automatic resource estimations.
- Jang et al. [175] presented the first implementation of GIFT in quantum circuits and estimated quantum resources for applying Grover's search algorithm to it.

Security Margin. None of the existing security analyses violates the security claims. The best key-recovery attack is on 27-round (out of 40) GIFT-128 [70]. Note that an attack on GIFT-128 does not immediately apply to GIFT-COFB. We can consider the security margin of GIFT-COFB to be high (i.e., at least 30%).

3.4. Grain-128AEAD

3.4.1. Overview of the Design

Grain-128AEADv2 [24] is a stream cipher based AEAD scheme optimized for hardware implementations. The main component of the Grain family is two bit-oriented feedback shift registers. In 2008, the initial version of the algorithm, namely Grain v1, was selected as a finalist in the hardware profile of the eSTREAM portfolio [176] [3], and Grain-128a is included in ISO/IEC 29167-13:2015 [177], which was developed for RFID systems. The design of Grain-128AEADv2 is similar to Grain-128a but has been updated to allow for a larger tag size and to support AEAD.

Submission updates. In response to the analysis of Chang and Turan [61], the initialization of Grain-128AEADv2 was updated in the final round and resulted in approximately a 33% increase in initialization time.

Variants. The AEAD variant of the Grain-128AEAD family is listed below.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*#Rounds*[1]
Grain-128AEAD	128	96	64	512

[1] The number of rounds is given for the initialization phase.

Security Claims. Submitters claim that the cryptanalytic attacks on Grain-128AEADv2 require at least 2^{112} computations on a classical computer in a single key and nonce-respecting setting and it is computationally difficult to reconstruct the key from the state that is known to the attacker, where Grain-128AEADv2 has a keystream limitation of 2^{80} bits for each key/nonce pair.

3.4.2. Security Analysis

Earlier versions of the Grain family – namely Grain v1, Grain-128, and Grain-128AEADv1 – have been investigated by a large number of third-party analyses (e.g., [96, 178–184]). Although the design of Grain-128AEADv2 is slightly different from earlier versions, some of the third-party analyses are still applicable.

- Hao et al. [185] presented distinguishing attacks up to 189 rounds with 2^{96} time complexity and a key-recovery attack for 190 rounds with 2^{123} time complexity. Note that the attacker is assumed to have access to pre-output bits after 190 rounds.

[3] The eSTREAM, the ECRYPT Stream Cipher Project, was a multi-year effort to promote the design of efficient and compact stream ciphers suitable for widespread adoption. Additional information is available at https://www.ecrypt.eu.org/stream/.

- Chang and Turan [61] analyzed the complexity of key recovery of Grain-128AEAD from the internal state under different scenarios. This study resulted in the final-round tweak.
- Hu et al. [186] proposed a new framework for recovering the exact algebraic normal forms of massive superpolies and applied it to recover the secret key of 191-round Grain-128AEAD with $2^{116.26}$ queries and $2^{118.6}$ memory bits. Note that the attacker is assumed to have access to pre-output bits after 191 rounds.
- He et al. [71] proposed a new framework for recovering superpolies and applied it to recover the secret key of 192-round Grain-128AEAD with 2^{127} queries. Note that the attacker is assumed to have access to pre-output bits after 192 rounds.
- Bendoukha et al. [187] investigated Grain-128AEAD-based transciphering using a fully homomorphic encryption.

Results in the quantum setting. Anand et al. [188] provided the implementation costs of Grover's key search algorithm for Grain-128AEAD.

Security Margin. None of the existing security analyses violates the security claims of the submitters. The best key-recovery attack is on Grain-128AEAD with 192-round (out of 512-round) initialization under the assumption that an attacker has access to the pre-output bits after 192 rounds without reintroducing key in the intialization phase [71], which results in a high security margin.

3.5. ISAP

3.5.1. Overview of the Design

ISAP [28] is a permutation-based AEAD scheme designed to provide algorithm-level security against a wider range of implementation attacks, such as differential fault attacks, statistical fault attacks, statistical ineffective fault attacks, and differential power analysis. The mode of ISAP is a nonce-based encrypt-then-MAC construction, where the encryption is done by XORing a message and a keystream, and the authentication/verification is based on a hash-then-MAC paradigm. The ISAP family uses the 320-bit ASCON and 400-bit KECCAK permutations.

Submission updates. In the final round, the primary variant is changed to ISAP-A-128a.

Variants. The variants of the ISAP family are listed below.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*Permutation*	*Rate* (in bits)	*#Rounds*
ISAP-A-128a	128	128	128	320-bit ASCON	1,64	12/1/6/12
ISAP-K-128a	128	128	128	400-bit KECCAK	1,144	16/1/8/8
ISAP-A-128	128	128	128	320-bit ASCON	1,64	12/12/12/12
ISAP-K-128	128	128	128	400-bit KECCAK	1,144	20/12/12/12

The *Rate* column provides two rate values, one for the nonce processing in the rekeying function IsapRk and the other one for the remaining phases. The *#Rounds* column provides the number of rounds for the authentication phase, nonce processing in the rekeying function, for the encryption and decryption phases, and for generating session keys in the rekeying function, respectively.

Security Claims. Submitters claim that all ISAP variants provide 128-bit security for the confidentiality of plaintexts and the integrity of plaintexts, AD, and nonce in nonce-respecting setting.

3.5.2. Security Analysis

The third-party security analysis of ISAP is summarized below.

- Udvarhelyi et al. [189] showed that the impact of combining masking and re-keying is limited in mitigating single-trace side-channel attacks and that combining shuffling and re-keying is theoretically appealing but can be practically challenging on low-cost embedded devices with low-noise levels.

Results on the ASCON permutation. Security analysis of the ASCON permutation is summarized in Section 3.1.2.

Results on KECCAK permutations. According to the KECCAK Crunchy Crypto Collision and Pre-image Contest [73], some preimage and collision attacks with KECCAK-f[400], where capacity is 160-bit, were reported up to three rounds (by Sun and Li) and four rounds (by Kölbl et al.), respectively. When the complexity is bounded by 2^{400}, zero-sum distinguishing attacks on KECCAK-f[1600] are known up to 15 rounds with 9-round forward and 6-round backward directions [125]. Additional results are listed in Section 3.2.2.

Results on the ISAP mode. Dobraunig and Mennink [190–193] and Dobraunig et al. [194] showed that the ISAP mode is leakage-resilient under both nonce-respecting and nonce-misuse.

Security Margin. None of the existing security analyses violates the security claims. The security of the ISAP mode requires the multi-target second-preimage resistance (2PI+ security) of the underlying hash function used in the authentication module [195], where the output of the hash function is defined by a capacity value. ISAP-A-128a and ISAP-A-128 are based on the 12-round ASCON permutation, and ISAP-K-128a and ISAP-K-128 are based on 16-round and 20-round KECCAK permutations, respectively. In case of ASCON-Xof, the best attack covers 4-round by the preimage attack given by Qin et al. [72]. According to the KECCAK Crunchy Crypto Collision and Pre-image Contest [73], Kölbl et al. described the best collision attack on the 4-round KECCAK permutation. Considering these attacks, ISAP variants have high security margins.

3.6. PHOTON-Beetle

3.6.1. Overview of the Design

PHOTON-Beetle is a permutation-based AEAD and hashing scheme. The underlying permutation of the PHOTON-Beetle family is the 256-bit PHOTON_{256} [196] permutation with 12 rounds. PHOTON-Beetle-AEAD is based on a sponge-like AEAD mode Beetle with a combined feedback (inspired by the COFB mode [148]), and PHOTON-Beetle-Hash is based on a sponge structure.

Submission updates. The PHOTON-Beetle team did not propose a design tweak.

Variants. The variants of the PHOTON-Beetle family are listed below. The *Rate* column provides the absorbing and the squeezing rates, respectively.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*Rate* (in bits)	*#Rounds*
PHOTON-Beetle-AEAD[128]	128	128	128	128/128	12
PHOTON-Beetle-AEAD[32]	128	128	128	32/128	12

Hash variants	*Digest size* (in bits)	*Rate* (in bits)	*#Rounds*
PHOTON-Beetle-Hash[32]	256	32/128*	12

* The first message block is 128-bit, and the following blocks are 32 bits.

Security Claims. Submitters made the following security claims:

- PHOTON-Beetle-AEAD[128] provides 121-bit security for both the confidentiality of plaintexts and the integrity of ciphertexts in nonce-respecting setting.
- PHOTON-Beetle-AEAD[32] provides 128-bit security for both the confidentiality of plaintexts and the integrity of ciphertexts in nonce-respecting setting.
- PHOTON-Beetle-Hash[32] provides 112-bit security for collision resistance (query complexity: $2^{111.5}$) and 128-bit security for preimage resistance.

3.6.2. Security Analysis

The following papers studied the security of PHOTON-Beetle-AEAD.

- Dobraunig and Mennink [197] identified an incorrect security bound in the submission (official comment, March 20, 2020) and described a key recovery attack with an empty message and AD that takes 2^{124} primitive queries. The complexity is too high for it to be

a threat under the NIST security requirements, but they recommended that the submitters update their security claims.

- Security proofs of the Beetle mode are available in [198–201].
- Wang et al. [202] presented a zero-sum distinguisher for full 12-round PHOTON_{256} permutation with a time complexity of 2^{184}.
- Cui et al. [74] presented a statistical integral distinguisher for 10-round PHOTON_{256} permutation with a time complexity of $2^{96.59}$ and memory complexity of $2^{70.46}$.
- Jean et al. [203] presented a rebound-like distingusher for 9-round PHOTON_{256} with a time complexity of 2^{184} and memory complexity of 2^{32}.
- Jean et al. [204] presented a multiple limited-birthday distinguisher for 8-round PHOTON_{256} permutation with a time complexity of $2^{10.8}$ and memory complexity of 2^8.
- Guo et al. [196] presented a rebound-like distinguisher for 8-round PHOTON_{256} permutation with a time complexity of 2^{16} and memory complexity of 2^8.
- Inoue et al. [168] presented a forgery attack on PHOTON-Beetle with success probability $O(q^2/2^b)$, where q is the number of encryption queries and b is the input size of the permutation. It does not invalidate the 121-bit INT-CTXT claim of PHOTON-Beetle since $b = 256$ for PHOTON-Beetle.

Results on PHOTON-Beetle-Hash. Regarding the collision resistance of PHOTON-Beetle-Hash, Mege [205] pointed out that a collision on PHOTON-Beetle-Hash occurs for about $2^{111.5}$ queries with high probability.

Results in the quantum setting. For preimage resistance, Lee et al. [112] provided estimated quantum resources for a quantum preimage attack on PHOTON-Beetle-Hash.

Security Margin. None of the existing security analyses violates the security claims of the submitters. There is no known cryptanalysis on round-reduced PHOTON-Beetle-AEAD, apart from the distinguishing attacks on the underlying permutation.

3.7. Romulus

3.7.1. Overview of the Design

Romulus is an AEAD and hashing scheme based on the tweakable block cipher `Skinny` [206]. Romulus-N uses a rate-1 TBC-based combined feedback mode, and the mode of Romulus-M follows a MAC-then-Encrypt approach.

Submission updates. The following changes were proposed in the final round:

- The number of AEAD modes from round two were reduced such that only one nonce-respecting variant Romulus-N and one nonce-misuse variant Romulus-M moved to the final round. Non-primary variants were removed.
- New variants were also added to the Romulus family. Romulus-T is a new leakage-resilient AEAD mode. The functionality of Romulus expanded with the addition of a new hash function, Romulus-H.
- The underlying tweakable block cipher was changed from `Skinny-128-256` (48 rounds) and `Skinny-128-384` (56 rounds) to `Skinny-128-384+` (40 rounds) in order to increase performance. In addition, `Skinny-128-384+` was designed to focus on 128-bit security.

Variants. Romulus consists of four variants: nonce-based AE (NAE) Romulus-N, nonce misuse-resistant AE (MRAE) Romulus-M, leakage-resilient Romulus-T, and hash function Romulus-H. Each of these variants uses the tweakable block cipher `Skinny-128-384+`, a variant of `Skinny-128-384` with 40 rounds instead of 56. The variants of the Romulus family are listed below.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*#Rounds*
Romulus-N	128	128	128	40
Romulus-M	128	128	128	40
Romulus-T	128	128	128	40

Hash variants	*Digest size* (in bits)	*Block size* (in bits)	*#Rounds*
Romulus-H	256	256	40

Security Claims. Submitters made the following security claims:

- Romulus-N provides 128-bit security for both privacy and authenticity in the nonce-respecting setting.
- Romulus-M provides 128-bit security for both privacy and authenticity in the nonce-respecting setting and 64-bit security for both privacy and authenticity in the nonce-

misuse setting. If the number of nonce repetitions is limited, the actual security bounds of Romulus-M are close to the full 128-bit security.

- Romulus-T provides 121-bit security for both privacy and authenticity in the nonce-respecting setting, 121-bit security for authenticity in the nonce-misuse setting, and 121-bit security for privacy as long as the nonces used for encryption queries are never used (nonce-misuse resilience).
- Romulus-N, Romulus-M, and Romulus-T provide 128-bit security for key recovery attacks in the single-key setting.
- Romulus-H provides 121-bit security for collision, preimage, and second preimage resistances.

3.7.2. Security Analysis

Results on Skinny-128-384. The following papers studied the security of Skinny-128-384.

- Tolba et al. [207] presented the impossible differential attack on 22 rounds of Skinny-128-384 with $2^{92.22}$ data, $2^{373.48}$ time, and $2^{147.22}$ memory complexities.
- Hadipour et al. [208] presented the related-tweakey rectangle attack on 30 rounds of Skinny-128-384 with $2^{125.29}$ data, $2^{361.68}$ time, and $2^{125.8}$ memory.
- Hadipour et al. [209] presented the integral attack on 26 rounds of Skinny-128-384 with 2^{121} data, 2^{344} time, and 2^{340} memory, as well as the related-tweakey integral attack on 27 rounds of Skinny-128-384 with $2^{124.99}$ data, $2^{362.61}$ time, and 2^{344} memory.
- Shi et al. [210] presented the meet-in-the-middle attack on 22 rounds of Skinny-128-384 with 2^{96} data, $2^{382.46}$ time and $2^{330.99}$ memory.
- Chen et al. [211] presented the meet-in-the-middle attack on 22 rounds of Skinny-128-384 with 2^{96} data, $2^{366.28}$ time, and $2^{370.99}$ memory.
- Zhao et al. [212, 213] presented the related-tweakey rectangle attack on 28 rounds of Skinny-128-384 with 2^{122} data, $2^{315.25}$ time, and $2^{122.32}$ memory.
- Qin et al. [214] presented related-key rectangle attacks on up to 30 rounds of Skinny-128-384 with 2^{341} time, and 2^{122} data.
- Dong et al. [75] presented related-key rectangle attacks on up to 32 rounds of Skinny-128-384 with 2^{355} time and 2^{123} data and presented the meet-in-the-middle attack on 23 rounds of Skinny-128-384 with 2^{376} time and 2^{104} data.
- Song et al. [76] presented related-key rectangle attacks on up to 32 rounds of Skinny-128-384 with 2^{345} time and 2^{123} data.
- Bijwe et al. [167] provided the implementation costs of Grover's key search algorithm for all the variants of Skinny.

Results on Romulus modes. The following papers studied the security of modes of Romulus.

- Habu et al. [215] presented privacy and authenticity attacks that show that the provable securities of Romulus-M are tight in both privacy and authenticity up to constant factors.
- Inoue et al. [173] showed that Romulus-N has 128-bit security for privacy and 64-bit security for authenticity in the nonce-misuse resilience setting.
- Proofs for privacy and authenticity of the mode of Romulus-N in the nonce-respecting setting (with beyond-birthday-bound security) and the mode of Romulus-M in the nonce-misuse setting were provided in [216–218].
- Iwata et al. [219] presented security proofs of INT-RUP security and the plaintext-awareness PA1 security for Romulus-M, and proposed Romulus-H hashing mode and Romulus-LR, Romulus-LR-TEDT leakage-resilient AEAD modes based on tweakable block ciphers.
- Guo et al. [220] proposed a rate-1 Leakage-Resilient AEAD based on the Romulus family along with security proofs in terms of CIML2, CCAmL1, and INT-RUP.
- Guo et al. [221] showed that Romulus-H has $(n - \log n)$-bit indifferentiability security from the random oracle, where n=128.

Results on Romulus-H. The preimage and the collision resistance of Romulus-H are studied in the following papers:

- Dong et al. [222] provided preimage and free-start collision attacks on 23-round Romulus-H with time complexity 2^{248} and 2^{124}.
- Nageler et al. [223] showed that collisions of 10-round Romulus-H and semi-free-start collisions of 14-round Romulus-H can be found within practical time.

Security Margin. None of the existing security analyses violates the security claims. There is no known cryptanalysis on round-reduced AEAD variants of Romulus. Instead, the best two key-recovery attacks are on 32-round (out of 40) Skinny-128-384+ in the related-key attack model [75, 76]. These two related-key attacks on Skinny-128-384+ require 2^{355} and 2^{345} time complexity, respectively, where the 384-bit tweakey is a target for recovery. Considering the results, the AEAD variants of Romulus provide sufficient security margin. The best attack on Romulus-H is a preimage attack on 23 rounds (out of 40) of Romulus-H, with a time complexity of 2^{248} beyond the time limit made by the submitters. This attack demonstrates the high security margin of the hash variants.

3.8. SPARKLE

3.8.1. Overview of the Design

SPARKLE includes the SCHWAEMM family of authenticated encryption schemes and the ESCH family of hash functions. Both are based on SPARKLE permutations that apply multiple distinct instances of Alzette – a 4-round 64-bit block cipher – to achieve non-linearity. Alzette is a 64-bit S-box based on an Addition-Rotation-XOR (ARX) design operating on 32-bit words, making it particularly efficient in software. The SCHWAEMM family of AEAD ciphers is based on the duplexed sponge construction with a combined feedback. The ESCH family of hash functions is based on the sponge construction.

Submission updates. In the final round, the primary variant was changed from SCHWAEMM192-192 to SCHWAEMM256-128.

Variants. The variants of the SPARKLE family are listed below. Both primary algorithms rely on the 384-bit SPARKLE permutation. In the table, each variant uses the b-bit SPARKLE permutation with r-bit rate and c-bit capacity, where $b = r + c$. In the *#Steps* column for AEAD variants, x/y indicates that the SPARKLE permutation with y steps is used (1) in the initialization, (2) between the AD processing and the message processing, and (3) in the finalization, and the SPARKLE permutation with x steps is used in (1) the AD processing and (2) the message processing. In the *#Steps* column for Hash variants, x/y indicates that the SPARKLE permutation with y steps is used once to generate the first half of the hash output, and the SPARKLE permutation with x steps is used (1) in the absorption phase and (2) to generate the second half of the hash output.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*Rate* (in bits)	*Capacity* (in bits)	*#Steps*
SCHWAEMM256-128	128	256	128	256	128	7/11
SCHWAEMM128-128	128	128	128	128	128	7/10
SCHWAEMM192-192	192	192	192	192	192	7/11
SCHWAEMM256-256	256	256	256	256	256	8/12

Hash variants	*Digest size* (in bits)	*Rate* (in bits)	*Capacity* (in bits)	*#Steps*
ESCH256	256	128	256	7/11
ESCH384	384	128	384	8/12
XOF variants				
XOESCH256	any	128	256	7/11
XOESCH384	any	128	384	8/12

Security Claims. Submitters made the following security claims:

- AEAD variants of SCHWAEMM have a security level of $k-8$ bits in terms of the number of executions of the underlying permutations in the nonce-respecting setting, where k is the size of the key.
- The claimed security level for ESCH variants is $\frac{c}{2}$ with regard to collision resistance, preimage resistance, second preimage resistance, and security against length-extension attacks, where c denotes the capacity.
- For the XOFs, the security level is $\min\{c/2, t/2\}$ bits for collision resistance and $\min\{c/2, t\}$ for (second) preimage resistance, where the maximal allowed output length t is the same as the data limit.

Functionality	Variant Name	Claimed Security Level (bits)	Data Limit (bytes)
AEAD	**SCHWAEMM256-128**	120	2^{68}
	SCHWAEMM128-128	120	2^{68}
	SCHWAEMM192-192	184	2^{68}
	SCHWAEMM256-256	248	2^{133}
Hash	**ESCH256**	128	2^{132}
	ESCH384	192	2^{196}
XOF	XOESCH256	$\min\{128, t/2\}^{\dagger}$, $\min\{128, t\}^{\ddagger}$	2^{132}
	XOESCH384	$\min\{192, t/2\}^{\dagger}$, $\min\{192, t\}^{\ddagger}$	2^{196}

† Collision resistance. ‡ (Second) preimage resistance.

3.8.2. Security Analysis

The following papers studied the security of SPARKLE.

- Beierle et al. [224] showed that the probability of 7-round differential trails of Alzette is at most 2^{-24}.
- Beierle et al. [225] showed that 4 (5) steps of the SPARKLE256 permutation and 5 (6) steps of the SPARKLE384 permutation and 6 (6) steps of the SPARKLE512 permutation are not enough to guarantee $b/2$-bit security against differential attacks (linear attacks), where b denotes the width of the permutation.
- Beierle et al. [42] reported that the probability of the best 7-round differential trails of Alzette is 2^{-26}.
- Schrottenloher and Stevens [226] provided guess-and-determine distinguishers on four steps of the SPARKLE256 and SPARKLE384 permutations and five steps of SPARKLE512 permutation with practical time complexity.
- Huang et al. [227] provided that the probability of the best 8-, 9-, 10-, and 11-round differential trails of Alzette is 2^{-34}, 2^{-40}, 2^{-46}, and 2^{-51}, respectively.

- Liu et al. [228, 229] presented 4-round differential-linear and rotational differential-linear trails in Alzette with correlation $2^{-0.27}$ ($2^{-0.1}$) and $2^{-11.37}$ ($2^{-7.35}$) theoretically (experimentally), respectively.
- Niu et al. [230] presented 4-, 5-, 6-, and 8-round differential-linear trails in Alzette with correlation 1 (1), $-2^{-0.33}$ ($-2^{-0.13}$), $2^{-4.95}$ ($2^{-1.45}$), and $2^{-8.24}$ ($-2^{-5.50}$) theoretically (experimentally), respectively.
- Xu et al. [231] showed without the Markov assumption that the original probabilities 2^{-23}, 2^{-23}, and 2^{-38} (calculated with the Markov assumption) of some 4-round differential trails should be changed with 0, 2^{-22}, and 0, respectively.
- Speel [232] provided three dominant 5-round linear trails of Alzette with correlations 2^{-5}, 2^{-7}, and 2^{-9}, respectively.

Results in the quantum setting. A summary of results in quantum setting is provided below.

- Jagielski and Kanciak [233] proposed an estimation of the quantum resources necessary for key-recovery attacks on the SCHWAEMM family of authenticated encryption scheme in the known-plaintext attack model.
- For preimage resistance of the hash function variants, Lee et al. [112] provided estimated quantum resources for quantum preimage attacks on ESCH256 and ESCH384.

Security Margin. None of the existing security analyses violates the security claims. The submitters provided guess-and-determine attacks that recover the capacity on 4.5-step SCHWAEMM128-128, 4.5-step SCHWAEMM192-192, and 4.5-step SCHWAEMM256-256 without whitening, as well as 3.5-step SCHWAEMM256-256 with whitening in Section 4.4 of [42]. All of these attacks on SCHWAEMM variants require data beyond the data limit made by the submitters. Note that once the capacity value of SCHWAEMM variants is determined, the key can be recovered from the internal state. Considering these attacks, all of the AEAD variants have high security margins. There is no known cryptanalysis on the hash variants. Instead, distinguishing attacks on the underlying permutations have been provided by the submitters. Their distinguishing attacks on 384-bit and 512-bit SPARKLE permutations are up to 6 steps.

3.9. TinyJAMBU

3.9.1. Overview of the Design

TinyJAMBU is an authenticated encryption scheme inspired by the third-round CAESAR candidate JAMBU [234]. The main component of TinyJAMBU is a keyed permutation (with no key schedule) that is based on a 128-bit nonlinear feedback shift register. The nonlinearity in each round is provided by a single `NAND` operation.

Submission updates. In the final round of evaluation, the number of rounds of the TinyJAMBU permutation that processes the nonce and AD during initialization and generate the last 32 bits of tag increased from 384 to 640. There was no change to the permutation that processes key setup and plaintext blocks. The update was done to provide a larger security margin against differential forgery attacks [235–237]. The new version of the algorithm family is called TinyJAMBU v2.

Variants. TinyJAMBU family has three AEAD variants. The number of rounds is represented by a 2-tuple, where the first number is the number of rounds used in key setup, plaintext processing, and the generation of first 32-bit of tag, and the second number is the number of rounds used in nonce and AD processing and the generation of last 32-bit of tag.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*State size* (in bits)	*#Rounds*
TinyJAMBU-128	128	96	64	128	1024/640
TinyJAMBU-192	192	96	64	128	1152/640
TinyJAMBU-256	256	96	64	128	1280/640

Security Claims. Assuming that each key is used to process at most 2^{50} bytes of messages (AD, plaintext/ciphertext), and each message is at least 8 bytes, the submitters' security goals of TinyJAMBU in nonce-respecting, nonce-misuse, unprotected decryption settings are summarized as follows.

Variants	Unique Nonce		Repeated Nonce			Unprotected decryption[1]		
	Enc.	Auth.	Secret key	Auth.	Forgery adv.	Secret key	Auth.	Forgery adv.
TinyJAMBU-128	112-bit	64-bit	112-bit	64-bit	2^{-15}	112-bit	64-bit	2^{-15}
TinyJAMBU-192	168-bit	64-bit	168-bit	64-bit	2^{-15}	168-bit	64-bit	2^{-15}
TinyJAMBU-256	224-bit	64-bit	224-bit	64-bit	2^{-15}	224-bit	64-bit	2^{-15}

[1] In this model, the decryption device may reveal the decrypted message even when the verification fails.

3.9.2. Security Analysis

The third-party security analyses of TinyJAMBU are summarized below.

- Saha et al. [235–237] showed a 338-round differential with probability $2^{-62.68}$ that leads

to a forgery attack on round-reduced TinyJAMBU v1 breaking 64-bit security and a differential on 384 rounds with probability $2^{-70.64}$.

- Teng et al. [238] presented various distinguishing and key-recovery cube attacks on reduced-round TinyJAMBU, targeting the initialization and the encryption phases. The distinguishing attacks are applied up to 437 (out of 1024) rounds, and the key recovery attacks are applied up to 428 (out of 1024) rounds, where the number of rounds corresponds to the permutation used in the encryption phase.
- Dunkelman et al. [78] presented related-key forgery attacks on (1) TinyJAMBU-256 with 2^{32} time complexity and 2^{10} related keys and (2) TinyJAMBU-192 with 2^{42} time complexity and 2^{12} related-keys.
- Dunkelman et al. [239] presented full-round practical zero-sum distinguishers on TinyJAMBU-128 and TinyJAMBU-192 and a reduced-round zero-sum distinguisher on TinyJAMBU-256 (1152 rounds out of 1280).
- Dutta et al. [77] presented a cube distinguishing attack on 476-round TinyJAMBU permutation in the weak-key setting.
- Datta et al. [240] analyzed the integrity security of TinyJAMBU in the release of unverified plaintext model and showed that it is INT-RUP secure.
- Jana et al. [241] presented a full-round type-4 differential trail of 1024-round TinyJAMBU keyed permutation with probability 2^{-108} (compared to 2^{-128}) that resulted in non-random properties, where type-4 means that no constraint is imposed on the input and output of the permutation.
- Li et al. [242] presented partial key-recovery nonce-respecting attacks on full TinyJAMBU v1 and round-reduced TinyJAMBU v2 by respectively using 384 (out of 384) and 387 (out of 640) rounds in tag generation phase, where $2^{96.8}$ tags are required in the single-key attack model.
- Sibleyras et al. [243] presented key-recovery slide attacks on TinyJAMBU's keyed permutations of key sizes 128, 192, and 256 bits with data/time complexities of about 2^{65}, 2^{66}, and $2^{69.5}$, respectively.

Security Margin. None of the existing security analyses violates the submitters' security goals in a single-key setting (see [49] for the responses of the designers). The best weak-key-recovery attack on TinyJAMBU-128 is on 476 rounds (out of 1024) [77], and there are two related-key-forgery attacks on full-round TinyJAMBU-192 and full-round TinyJAMBU-256 [78]. Although the security margin in single-key setting is high, in related-key setting, TinyJAMBU-192 and TinyJAMBU-256 do not provide sufficient security.

3.10. Xoodyak

3.10.1. Overview of the Design

Xoodyak is a permutation-based AEAD and hashing scheme. Xoodyak is built from a fixed 384-bit permutation (called Xoodoo) operated in Cyclist mode. The design approach of Xoodoo is closely related to that of the KECCAK permutation. The 384-bit state is arranged in a three-dimensional array of $3 \times 4 \times 32$ bits, nonlinearity is provided by simple operations on 3-bit columns, linear mixing is provided by mixing between sheets and moving the bits within the sheets around, and a constant addition ensures that there is some difference between rounds. Cyclist mode takes a fixed permutation and provides the functionality of both hashing (sponge mode) and AEAD (duplex mode) along with some new functionality, including tuple hashing, XOFs, and the generation of rolling subkeys.

Submission updates. In the final round, the key and nonce are processed together in a single call instead of separately in two calls, resulting in 12 fewer rounds needed to compute, which leads to fast processing of short messages.

Variants. The variants of the Xoodyak family are listed below.

AEAD variants	*Key size* (in bits)	*Nonce size* (in bits)	*Tag size* (in bits)	*Rate* (in bits)	*Capacity* (in bits)	*#Rounds*
Xoodyak	128	128	128	192	192	12

Hash variants	*Digest size* (in bits)	*Rate* (in bits)	*Capacity* (in bits)	*#Rounds*
Xoodyak	256	130	254	12
XOF variants				
Xoodyak	any	130	254	12

Security Claims. The submitters claimed that the nonce-based Xoodyak authenticated encryption scheme can resist an adversary with up to 2^{128} computational complexity and up to 2^{160} data complexity. They also claimed that the security strength levels of the Xoodyak hash function and Xoodyak XOF are $\min\{8n/2, 128\}$ bits for collision resistance, $\min\{8n, 128\}$ bits for preimage and second preimage resistances, and $\min(8n - \log\ m, 128)$ bits for m-target preimage resistance, where n is the output size in bytes.

3.10.2. Security Analysis

The following papers studied the security of Xoodyak.

- Song and Guo [244] demonstrated a cube-like key-recovery attack on Xoodoo-AE reduced to six (of 12) rounds in 2^{89} time and 2^{55} memory.

- Zhou et al. [79] also presented a conditional cube attack on Xoodyak reduced to six (of 12) rounds in a nonce-misuse setting, recovering the 128-bit key in $2^{43.8}$ time and negligible memory cost.
- Liu et al. [128] showed a zero-sum distinguisher on the full 12-round Xoodoo with 2^{33} time complexity, and Liu et al. [228] identified a 4-round rotational differential-linear distinguisher with a correlation 1 on Xoodoo with a probability $2^{-117.81}$.
- Zhang et al. [245] suggested using a genetic algorithm to speed up the capacity-recovery of Xoodyak in the nonce-respecting and nonce-misuse settings. They experimented with using 4-round, 5-round, and 6-round Xoodyak with the state reduced to 96 bits, where the rate was 48-bit and the capacity was 48-bit.
- Dunkelman and Weizman [246] presented a key-recovery differential-linear attack on 5-round Xoodyak in the related key attack model with about data complexity $2^{22.05}$ and time $2^{22.04}$.
- Blach [247] analyzed the effect of the shift constants of Xoodoo permutation and presented new shift constants to build a new Xoodoo-like permutation with increased 3-round lower trail bounds.
- Daemen et al. [248] proved that the minimum weights of any 4-round, 6-round, and 12-round trails of Xoodoo are 80, at least 132, and at least 264, respectively, for both differential and linear trails.
- Li et al. [249] developed a SAT-based automatic search toolkit called XoodooSat to search for 2-round, 3-round, and 4-round differential trail cores of Xoodoo.
- Hu and Peyrin [96] gave 4-round deterministic higher-order differential-linear distinguishers for Xoodoo with only four chosen plaintexts.
- Bellini and Makarim [250] proposed a generalized differential-like cryptanalysis using a binary relation in a form of functions, called functional cryptanalysis. As an example, authors presented a functional distinguisher on 3-round Xoodoo with 2^{11} input pairs.
- Gilbert et al. [251] provided a new generic forgery attack against several duplex-based AEAD modes with $O(2^{\frac{3c}{4}})$ data and $O(2^{\frac{3c}{4}})$ time, where c is the capacity. In case of Xoodyak AEAD, their attack also recovered the secret key with a negligible amount of additional computations.

Results on Xoodyak-Hash and Xoodyak-XOF. The following papers studied the preimage resistance of the hash function variants:

- Liu et al. [252] presented a deep learning-based preimage attack on 1-round Xoodyak hash mode.
- Qin et al. [72] introduced bit-level MILP-based automatic tools and gave preimage attacks on 3-round Xoodyak-XOF, whose output size is 128-bit, with time $2^{125.06}$ and

memory 2^{97}.

Results in the quantum setting. Lee et al. [112] provided estimated quantum resources for a quantum preimage attack on Xoodyak-Hash.

Security Margin. None of the existing security analyses violates the submitters' security goals in a single-key setting. The best key-recovery attack is on 6 rounds (out of 12) [244] of Xoodyak, which results in security margin of around 50%. In case of the hash variants, the best attack is a preimage attack on 3-round Xoodyak-XOF, which results in high security margin of 75%.

4. Benchmarking Results

This section summarizes the main software and hardware performance benchmarking initiatives that were considered during evaluation of the finalists. While these efforts provided crucial information on the performance of the finalists, it is important to note their limitations. Results may not present a complete picture of a finalist's potential for optimization in any particular metric. Further, not all implementations are designed with the same assumptions or goals, and there are more diverse implementations for some finalists than others. More efficient implementations are likely possible for all finalists. As such, the results of these efforts were considered as a general guide and not a strict ranking.

4.1. Software Benchmarking

Software performance on microcontrollers is an important criterion for the evaluation of the finalists. Multiple benchmarking initiatives evaluated the performance of the finalists. These initiatives cover a wide range of target platforms from 8-bit microcontrollers with limited memory to 32-bit and 64-bit microcontrollers. The specifications of the microcontrollers used by the benchmarking initiatives are summarized in Table 8.

Table 8. Specifications of microcontrollers used in benchmarking initiatives

Initiative	*Microcontroller*	*Processor*	*Word size*	*Clock speed*	*Flash*	*RAM*
NIST [253]	ATmega328P	AVR	8-bit	16 MHz	32 KB	2 KB
	ATmega4809	AVR	8-bit	16 MHz	48 KB	6 KB
	SAMD21G18A	ARM Cortex-M0+	32-bit	48 MHz	256 KB	32 KB
	nRF52840	ARM Cortex-M4	32-bit	64 MHz	1 MB	256 KB
	PIC32MX320F128H*	MIPS32 M4K	32-bit	80 MHz	128 KB	16 KB
	PIC32MX340F512H	MIPS32 M4K	32-bit	80 MHz	512 KB	32 KB
	ESP8266	Tensilica L106	32-bit	80 MHz	4 MB	80 KB
	AT91SAM3X8E	ARM Cortex-M3	32-bit	84 MHz	512 KB	96 KB
Renner et al. [254]	ATmega328P	AVR	8-bit	16 MHz	32 KB	2 KB
	STM32F103C8T6	ARM Cortex-M3	32-bit	72 MHz	64 KB	20 KB
	STM32F746ZG	ARM Cortex-M7	32-bit	216 MHz	1 MB	320 KB
	ESP32 WROOM	Tensilica Xtensa LX6	32-bit	240 MHz	4 MB	520 KB
	Kendryte K210	RISC-V (Dual Core)	64-bit	400 MHz	16 MB	8 MB
Weatherley [255]	ATmega2560	AVR	8-bit	16 MHz	256 KB	8 KB
	AT91SAM3X8E	ARM Cortex-M3	32-bit	84 MHz	512 KB	96 KB
	ESP32	Tensilica Xtensa LX6	32-bit	240 MHz	4 MB	520 KB

*PIC32MX340F512H microcontroller used with PlatformIO's PIC32MX320F128H board profile

4.1.1. Microcontroller Benchmarking by NIST

The NIST team evaluated the performance of the finalists on microcontrollers and compared them against the NIST standards AES-GCM and SHA-256. The implementations were collected from the submission packages, GitHub repositories of the finalists, and the repositories of other benchmarking initiatives. In total, 275 AEAD, 153 hash and 103 com-

bined implementations were used on two 8-bit MCUs and six 32-bit MCUs [253] (See Table 9).

This benchmarking effort focused on two metrics: code size and execution time. Results were analyzed under each metric independently, as well as jointly. Code size is measured in the amount of flash used, in bytes, upon successful compilation of each implementation under test. Execution time is measured in either microseconds or cycles depending on the platforms.

Table 9. Number of implementations per finalists

Finalists	*AEAD*	*Hash*	*Combined*	*Total*
ASCON	120	110	52	282
Elephant	6	-	-	6
GIFT-COFB	11	-	-	11
Grain-128AEAD	6	-	-	6
ISAP	37	1*	4	42
PHOTON-Beetle	20	10	16	46
Romulus	32	11	27	70
SPARKLE	25	13	3	41
TinyJAMBU	9	-	-	9
Xoodyak	9	8	1	18
Total	275	153	103	531

* Note that ISAP does not include an official hash function variant.

Size experiments
Size information was obtained from the PlatformIO [256] development platform, which displays the flash used, measured in bytes, upon successful compilation of each implementation under test. NIST used this value for code size rather than deriving size as the sum of read-only code and initialized values as was done in the second round. Size measurements were acquired for executables that supported AEAD, hashing, or combined AEAD and hashing functionalities.

Timing experiments
AEAD timing experiments included both encryption and decryption, and therefore both operations were supported in the compiled binaries. Hashing timing experiments were performed on executables that supported hashing only. No binaries supporting a combination of encryption, decryption, and hashing were timed.

Execution time was captured for a variety of input sizes. AEAD operations were performed for all AD length and message length combinations from the range 0 bytes to 128 bytes in increments of 8 bytes. In addition, messages with length 256, 384, and 512 bytes were paired with empty AD. Similarly, AD with length 256, 384, and 512 bytes were paired with empty messages. Hash inputs ranged from 0 bytes to 128 bytes in increments of 8

bytes, with additional input lengths of 256, 384, and 512 bytes. Implementations were further tested up to 2048 bytes, in increments of 128 bytes, on some platforms.

Summary of results
ASCON, GIFT-COFB, and TinyJAMBU were consistently among the top performers in terms of AEAD size. SPARKLE and Xoodyak also demonstrated more compact implementation than AES-GCM on most platforms. Table 10 provides a summary of results when size and time are considered independently.

Size and timing summaries for hashing are presented in Table 11. ASCON and SPARKLE were consistently smaller than SHA-256 on all platforms, while Xoodyak was smaller on six of them. None of the hashing algorithms were faster than the fastest SHA-256 implementation on all boards, though Xoodyak and SPARKLE performed best in this comparison.

Further details and results from the NIST MCU benchmarking effort can be found in Appendix B.

4.1.2. Microcontroller Benchmarking by Renner et al.

Renner et al. [254, 257–259] developed a benchmarking framework to evaluate the performance of AEAD algorithms on microcontrollers. The benchmarking uses execution time (microseconds, average time to generate the test vectors), size of the compiled binary, and RAM usage (only on the STM32F7) as performance metrics. Benchmarks were collected for 295 implementations on five microcontrollers.

Some benchmark results are included in Figure 2. The speed measurements on the Arduino Uno show SPARKLE, GIFT-COFB and Xoodyak in the top three, with ASCON, TinyJAMBU and Romulus being fairly close. ASCON, Xoodyak and TinyJAMBU are leading in the speed benchmark on the ESP32. The code size measurements on the Arduino Uno show ASCON, PHOTON-Beetle reach the smallest code sizes. In the code size measurements on the Maixduino, ASCON, ISAP, TinyJAMBU, SPARKLE, and Xoodyak reach the lowest code sizes.

Considering only the primary variants, the results can be summarized as follows. Overall, ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak stood out as top performers.

- ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak were the fastest on the Arduino Uno, while ASCON, GIFT-COFB, ISAP, PHOTON-Beetle, and Xoodyak had the smallest code sizes.
- On a STM32F103 MCU, ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak had the fastest implementations and ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak used the least ROM.

Table 10. Software performance summary of AEAD primary variants vs. AES-GCM.

		PIC32MX320F128H	PIC32MX340F512H	AT91SAM3X8E	SAMD21G18A	nRF52840	ATmega4809	ESP8266	ATmega328P
ASCON	smaller (encrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (decrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (encrypt and decrypt)	✓	✓	✓	✓	✓	✓	✓	✓
	faster encryption	✓	✓	✓	✓	✓	✓	✓	✓
	faster decryption	✓	✓	✓	✓	✓	✓	✓	✓
Elephant	smaller (encrypt only)	✗	✗	✗	✗	✗	✗	✗	✗
	smaller (decrypt only)	✗	✗	✗	✗	✗	✗	✗	✗
	smaller (encrypt and decrypt)	✗	✗	✗	✗	✗	✗	✗	✗
	faster encryption	✗	✗	✗	✗	✗	✗	✗	✗
	faster decryption	✗	✗	✗	✗	✗	✗	✗	✗
GIFT-COFB	smaller (encrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (decrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (encrypt and decrypt)	✓	✓	✓	✓	✓	✓	✓	✓
	faster encryption	✓	✓	✓	✓	✓	✓	✗	✓
	faster decryption	✓	✓	✓	✓	✓	✓	✗	✓
Grain-128AEAD	smaller (encrypt only)	✗	✗	✗	✗	✗	✗	✗	✗
	smaller (decrypt only)	✗	✗	✗	✗	✗	✗	✓	✗
	smaller (encrypt and decrypt)	✗	✗	✗	✗	✗	✗	✗	✗
	faster encryption	◗	◗	◗	◗	◗	◗	✗	◗
	faster decryption	◗	◗	◗	◗	◗	◗	✗	◗
ISAP	smaller (encrypt only)	✗	✗	✗	✗	✗	✗	✗	✗
	smaller (decrypt only)	✗	✗	✗	✗	✗	✗	✗	✗
	smaller (encrypt and decrypt)	✗	✗	✗	✗	✗	✗	✗	✗
	faster encryption	✗	✗	✗	✗	✗	◗	✗	✗
	faster decryption	✗	✗	✗	✗	✗	✗	✗	✗
PHOTON-Beetle	smaller (encrypt only)	✗	✗	✗	✗	✗	✓	✗	✓
	smaller (decrypt only)	✗	✗	✗	✗	✗	✓	✗	✓
	smaller (encrypt and decrypt)	✗	✗	✗	✗	✗	✓	✗	✓
	faster encryption	✗	✗	✗	✗	✗	✓	✗	✓
	faster decryption	✗	✗	✗	✗	✗	✓	✗	✓
Romulus	smaller (encrypt only)	✗	✗	✗	✗	✗	✗	✗	✗
	smaller (decrypt only)	✗	✗	✗	✗	✗	✗	✗	✗
	smaller (encrypt and decrypt)	✗	✗	✗	✗	✗	✗	✗	✗
	faster encryption	◗	◗	✓	◗	◗	✓	✗	✓
	faster decryption	◗	◗	◗	◗	◗	✓	✗	✓
SPARKLE	smaller (encrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (decrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (encrypt and decrypt)	✓	✓	✗	✗	✓	✗	✓	✗
	faster encryption	✓	✓	✓	✓	✓	✓	◗	✓
	faster decryption	✓	✓	✓	✓	✓	✓	◗	✓
TinyJAMBU	smaller (encrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (decrypt only)	✓	✓	✓	✓	✓	✓	✓	✓
	smaller (encrypt and decrypt)	✓	✓	✓	✓	✓	✓	✓	✓
	faster encryption	✓	✓	✓	✓	✓	✓	◗	✓
	faster decryption	✓	✓	✓	✓	✓	✓	◗	✓
Xoodyak	smaller (encrypt only)	✓	✓	✗	✓	✗	✓	✓	✓
	smaller (decrypt only)	✓	✓	✗	✓	✗	✓	✓	✓
	smaller (encrypt and decrypt)	✓	✓	✗	✓	✗	✗	✓	✗
	faster encryption	✗	✗	✗	✗	✗	✓	✗	✓
	faster decryption	✗	✗	✗	✗	✗	✓	✗	✓

✓and ✗ denote true and false, respectively. For speed measurements, ◗ is also used to denote a candidate was faster for some, but not all, tested inputs.

- The fastest five on the ESP32 MCU were ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak, and the smallest were ASCON, ISAP, SPARKLE, TinyJAMBU, and Xoodyak.
- ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak were fastest on the STM32F7 MCU; ASCON, ISAP, SPARKLE, TinyJAMBU, and Xoodyak used the least ROM; and ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak used the least RAM.
- ASCON, GIFT-COFB, SPARKLE, TinyJAMBU, and Xoodyak were the fastest on the Maixduino and ASCON, ISAP, SPARKLE, TinyJAMBU, and Xoodyak used the least ROM.

4.1.3. Microcontroller Benchmarking by Weatherley

Weatherley [255, 260] developed optimized implementations of the finalists and performed timing measurements on three test platforms: one 8-bit AVR test platform and two with 32-bit architectures (ARM Cortex-M3 and ESP32). Encryption and decryption operations were executed with message sizes of 16 bytes and 128 bytes and no associated data, while hashing measurements were obtained for message lengths of 16, 128, or 1024 bytes. Results were presented in terms of speedup (time to execute operation using base implementation divided by time to execute operation using candidate implementation). ChaChaPoly [261–263] was used as the base for AEAD operations, while BLAKE2s [264] served as the base for hashing. Code in this effort was optimized for size, not speed. However, the sizes of each compiled executable were not presented with the timing results.

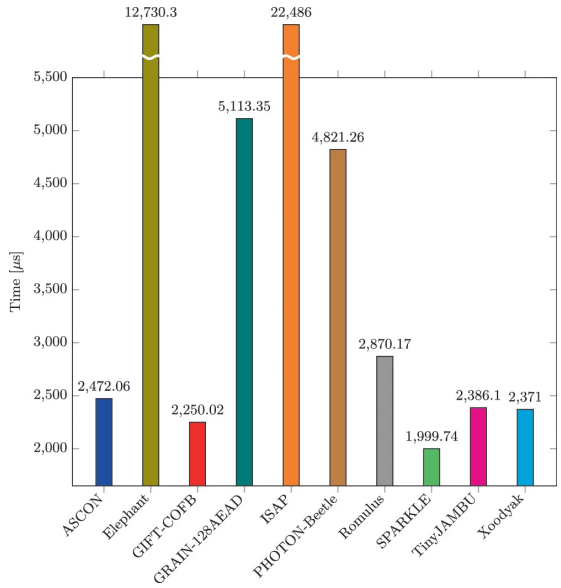
Results can be summarized as follows:

AVR platform. ASCON, GIFT-COFB, PHOTON-Beetle, Romulus, SPARKLE, TinyJAMBU, and Xoodyak were faster than ChaChaPoly for all benchmarks. Grain-128AEAD was faster than ChaChaPoly for the smaller 16-byte messages and slower for the larger 128-byte messages. SPARKLE had the greatest speedups over ChaChaPoly in these benchmarks. In addition, hash results were most favorable for SPARKLE and SHA-256.

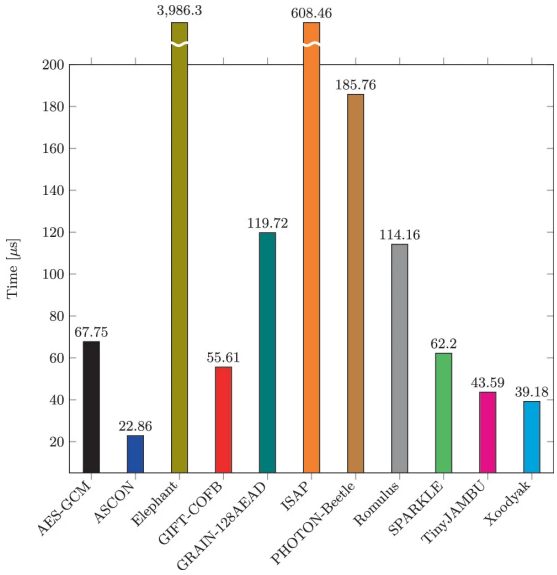
32-bit ARM Cortex-M3 platform. Implementations of ASCON, GIFT-COFB, SPARKLE, and Xoodyak were faster than ChaChaPoly on all AEAD benchmarks, while TinyJAMBU was faster only for the smaller message lengths. SPARKLE and Xoodyak were the only candidates faster than BLAKE2s.

32-bit ESP32 platform. Only SPARKLE implementations were faster than ChaChaPoly on all four AEAD benchmarks. There was no hash algorithm among the finalists that was faster than BLAKE2s.

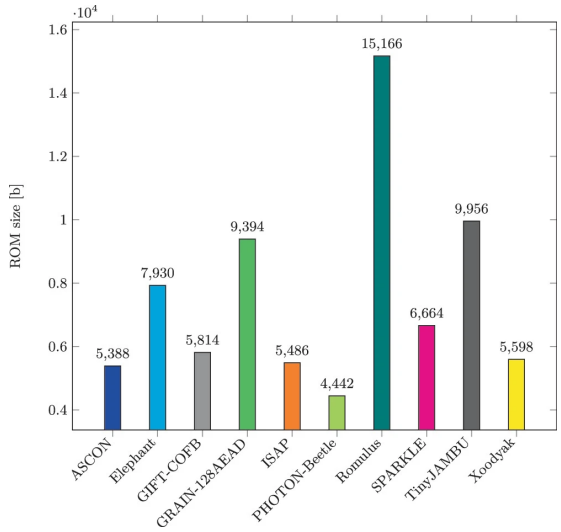
Graphical depictions of the results are presented in Figure 3.



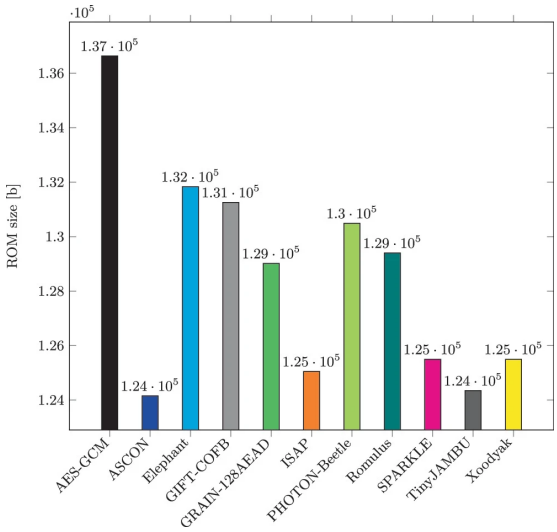
(a) Speed measurements on the Arduino Uno



(b) Speed measurements on the ESP32

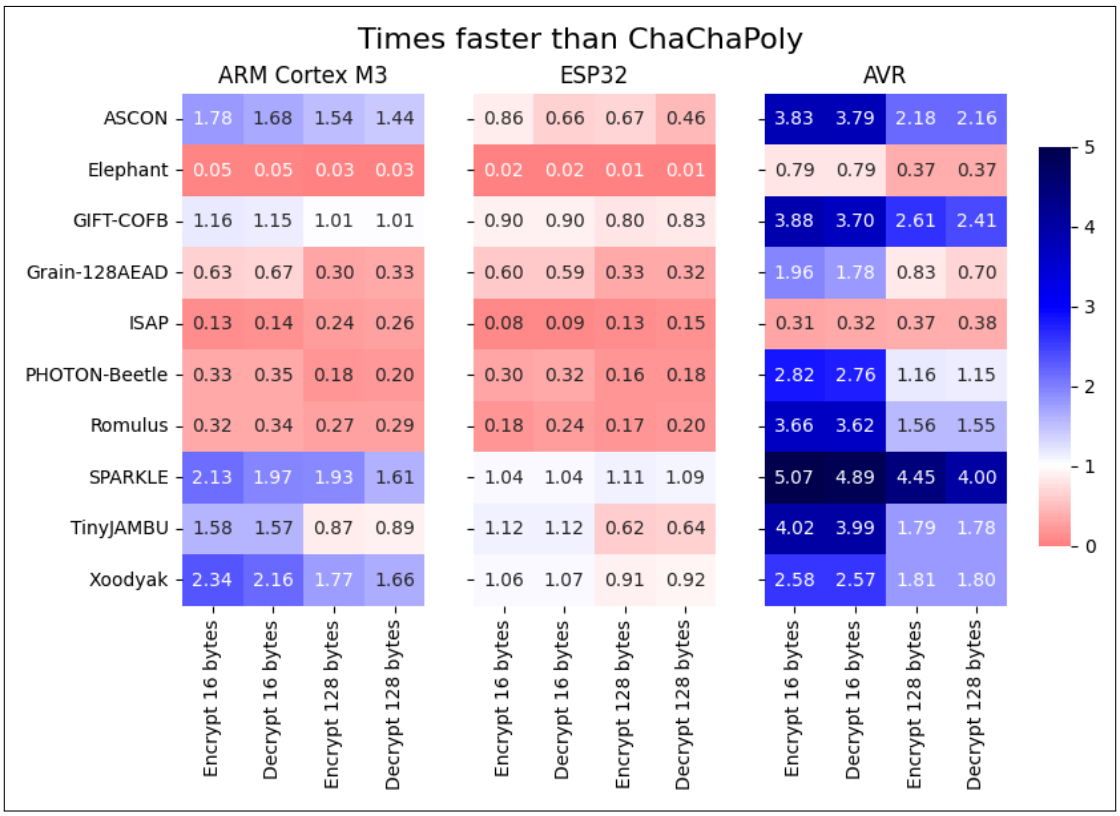


(c) Code size measurements on the Arduino Uno

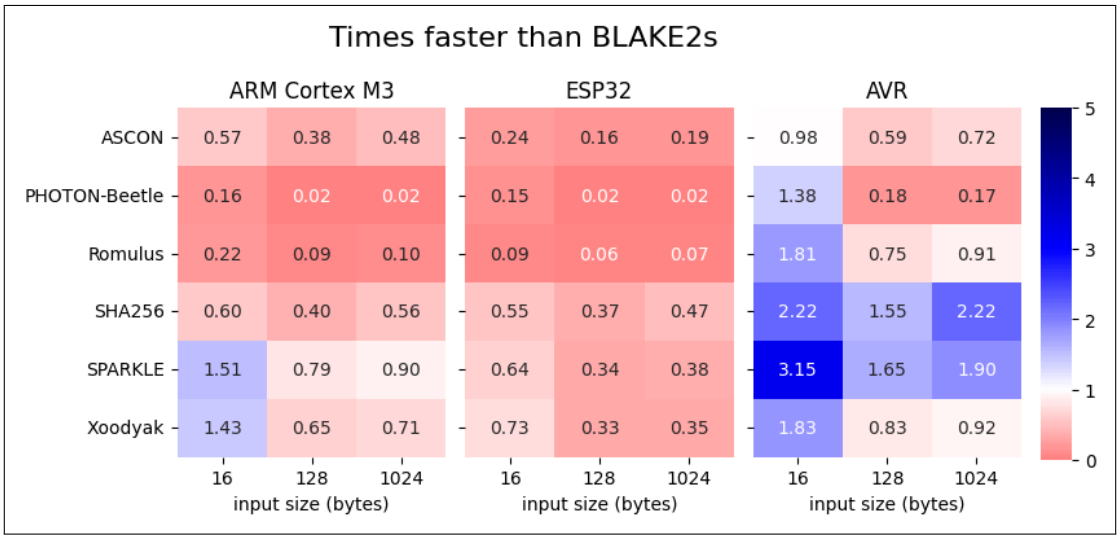


(d) Code size measurements on the Maixduino

Fig. 2. Speed and code size measurements by Renner et al. [259]



(a) Speedup of primary AEAD variants relative to ChaChaPoly with no AD



(b) Speedup of primary hash variants compared to BLAKE2s

Fig. 3. Heatmap representations of speedup results from Weatherly [255]

4.1.4. Benchmarking from eBACS

eBACS (ECRYPT Benchmarking of Cryptographic Systems) [265] is a repository of software performance benchmarking results on a wide variety of platforms that includes servers, desktops and higher-end embedded systems. There are several smaller benchmarking projects, each focusing on a different type of primitive or functionality. Results from eBAEAD (ECRYPT Benchmarking of Authenticated Ciphers) and eBASH (ECRYPT Benchmarking of All Submitted Hashes) results were considered for server or hub devices. A large number of processors are covered, but some processors require specific implementations to take advantage of various features. If these implementations are not available, optimal results for an algorithm on a processor may not be attained.

In platforms where AES-NI instructions are available, AES-GCM outperforms all the finalists (see Table 12). On a 64-bit system without AES-NI instructions the primary variants of ASCON and Xoodyak did better than AES-GCM across all data sizes. Additionally, on long inputs ISAP and GIFT-COFB are competitive followed by Romulus and TinyJAMBU. For short inputs, GIFT-COFB, TinyJAMBU, Romulus, and SPARKLE also beat AES-GCM (or at least were very close). On a 32-bit ARM Cortex-A7 system variants of SPARKLE and ASCON outperformed AES-GCM across most input message sizes. Additionally, as input message sizes decreased, GIFT-COFB and Xoodyak also outperformed AES-GCM. On small sizes TinyJAMBU was also competitive.

For hashing ASCON and Xoodyak are competitive with SHA-256 especially for shorter message sizes. As expected, when intrinsic SHA-256 instructions are available, it outperforms all finalists.

4.1.5. Additional Results

The following studies provide additional software benchmarking results.

- Watanabe et al. [266] implemented four of the finalists, namely ASCON, Grain-128AEAD, TinyJAMBU, and Xoodyak, in addition to AES-128-GCM. Authors optimized the RAM consumption of the implementaions on AVR ATmega 128 and ARM Cortex-M3 microcontrollers. According to their results, TinyJAMBU has the smallest RAM usage, 117 bytes on AVR ATmega 128 and 140 bytes on ARM Cortex-M3.
- Ruigrok [267] provided microcontroller benchmarks for scenarios using a modified Transport Layer Security (TLS) 1.3 implementation that included nine of the finalists, excluding SPARKLE. TLS 1.3 specifies AES used with AEAD modes and ChaCha20-Poly1305, where comparisons to the latter were made in this work. The study found that optimized implementations of ASCON and Xoodyak outperformed the ChaCha20-Poly1305 in terms of speed (seconds), storage (bytes), and memory use (bytes). Cheng et al. [268] presented the design, implementation, and evaluation of RISC-V Instruction Set Extensions (ISEs) for the ten finalists. They demonstrated that ISEs are valuable for lightweight cryptography applications. With little hardware overhead they can reduce

Table 11. Software performance summary of hashing primary variants vs. SHA-256 on microcontrollers.

		PIC32MX320F128H	PIC32MX340F512H	AT91SAM3X8E	SAMD21G18A	nRF52840	ATmega4809	ESP8266	ATmega328P
ASCON	smaller	✓	✓	✓	✓	✓	✓	✓	✓
	faster	✗	✗	◗	✗	◗	✗	✗	✗
PHOTON-Beetle	smaller	✗	✗	✗	✗	✗	✓	✗	✓
	faster	✗	✗	✗	✗	✗	✗	✗	✗
SPARKLE	smaller	✓	✓	✓	✓	✓	✓	✓	✓
	faster	◗	◗	✓	◗	◗	✗	✗	✗
Romulus	smaller	✗	✗	✓	✗	✗	✗	✗	✓
	faster	✗	✗	✗	✗	✗	✗	✗	✗
Xoodyak	smaller	✓	✓	✗	✓	✗	✓	✓	✓
	faster	◗	◗	✓	◗	◗	✗	◗	✗

Table 12. Example results from eBACS for encryption thoughput, in cycles per byte, for two platforms - Hiphop (with AES-NI instructions) and Berry2 (without AES-NI instructions).

Finalist	Hiphop		Berry2	
	(64,64)	(1536,1536)	(64,64)	(1536,1536)
AES-GCM	8.09	1.23	120.61	56.45
ASCON	15.01	10.8	69.55	52.05
Elephant	9341	6975.29	20680	15426
GIFT-COFB	51.82	42.44	99.71	78.29
Grain-128AEAD	35.46	25.92	-	-
ISAP	115	36.11	316.93	128.74
PHOTON-Beetle	75.42	66.32	411.27	364.79
Romulus	39.09	32.56	289.12	236.23
SPARKLE	34.74	21.2	86.81	51.94
TinyJAMBU	58.83	49.77	132.3	110.4
Xoodyak	23.02	15.8	89.16	60.06

(x,x) denotes the size of the AD and message, in bytes.

execution latency and implementation size, as well as allow constant-time execution of software implementations.

- Lee et al. [112] proposed optimization techniques for hash implementations of ASCON, PHOTON-Beetle, SPARKLE, and Xoodyak. The performance of these implementations was evaluated on a GPU platform (RTX 3080) and quantum computer (ProjectQ). In GPU, SPARKLE was able to achieve higher throughput compared to the other three candidates.
- Hira et al. [269] evaluated the software performance of 32 second-round candidates on the mbed LPC1114FN28 microcontroller (32-bit ARM Cortex-M0). The benchmark provides latency (in seconds) and ROM usage (in KB) of reference implementations as performance metrics. They conducted an experiment to investigate the relationship between data size and latency for the finalists. Considering only the primary variants of finalists, ASCON, SPARKLE, TinyJAMBU, and Xoodyak showed low latency and TinyJAMBU used the least ROM.

The following papers provide additional results on the software performance of ASCON.

- Altinay and Örs [270] extended the RISC-V set instructions with basic operations of the ASCON permutation implemented operations of ASCON as an instruction extension for RISC-V, accelerated the ASCON execution, and reduced the instruction memory for constrained devices.
- Steinegger and Primas [271] implemented ASCON permutation as an instruction extension for RISC-V and showed that the instruction extension requires about half of the area of dedicated ASCON co-processor designs and is easy to integrate into low-end embedded devices, like 32-bit ARM Cortex-M or RISC-V microprocessors.
- Dobraunig et al. [272] presented updated results on the performance and code size of ASCON AEAD, hashing, and combined implementations. They also included new performance improvements for ASCON-PRF, MAC, and Short-Input MAC.

The following papers provide additional results on the software performance of GIFT-COFB.

- Adomnicai et al. [273] presented a new representation for the GIFT family called fixslicing, which allows extremely efficient software bitsliced implementations using only a few rotations, making GIFT a very efficient candidate on microcontrollers.
- Charlès and Gravouil [274, 275] developed a white-box encoding solution and applied it to GIFT-128, the permutation of GIFT-COFB.

The following papers provide additional results on the software performance of Grain-128AEAD.

- Maximov and Hell [276] presented software implementations of the cipher that targeted constrained processors. The processors chosen were the 8-bit (AVR) and 16-bit (MSP)

processors used in the FELICS-AEAD framework. They made four different implementations, where both high-speed and small code size were targeted. Using the FELICS framework for benchmarking, they concluded that Grain-128AEAD is competitive with other algorithms currently included in the FELICS framework.

- Watanabe et al. [277] presented implementation techniques for memory-optimized implementations of lightweight hardware-oriented stream ciphers, including Grain-128a specified in ISO/IEC 29167-13 for RFID protocols. Their results were memory-optimized implementations of Grain-128a, one of which required 84 RAM bytes on ARM Cortex-M3.

The following paper provides additional results on the software performance of Romulus.

- Adomnicai et al. [278] introduced optimized Skinny-128 implementations on various SIMD architectures by decomposing the 8-bit S-box into four 4-bit S-boxes and reported improved benchmark results of Romulus on ARM Neon and Intel SSE processors.

The following paper provides additional results on the software performance of TinyJAMBU.

- Duka [279] reported on the software implementation details of TinyJAMBU on a Siemens S7–1200 Programmable Logic Controller. They evaluated the execution speed and memory requirements of each variant of TinyJAMBU on this industrial controller.

The following paper provides additional results on the software performance of Xoodyak.

- Burgt [280] presented implementations based on the Xoodyak specification aimed at low-power devices, like the ARM Cortex-M group of chips used in various microcontrollers.

4.2. Hardware Benchmarking

Performance in hardware is another important criterion for the evaluation of the finalists. Multiple benchmarking initiatives evaluated the performance of unprotected implementations in the second round and many of the results are applicable to the final round. NIST considered results from the hardware benchmarking initiatives described below.

4.2.1. FPGA Benchmarking by GMU

The George Mason University (GMU) Cryptographic Engineering Research Group (CERG) conducted a performance study on FPGA implementations during the second round of evaluation [281]. While tweaks to some of the candidates had an impact on performance in the final round, round two results for many of the finalists remained relevant in this round and are discussed in Section 4.2.2. The second-round FPGA benchmarking effort was not repeated for the final round; instead, GMU CERG shifted their focus to evaluation of protected implementations [282, 283]. However, a limited number of new unprotected implementations were considered for designs that were tweaked for the final round. This was done to evaluate the impacts of protection on implementation size and performance.

The GMU team used the Xilinx Artix-7 platform for benchmarking in this round of evaluation. Both protected and unprotected implementations were benchmarked and compared. Performance comparisons, such as throughput and area of unprotected versus order protected implementations, or the number of random bits needed for masking, provides insight into the cost of applying protection methods to an unprotected implementation. For results on protected implementations, see Section 4.3.1.

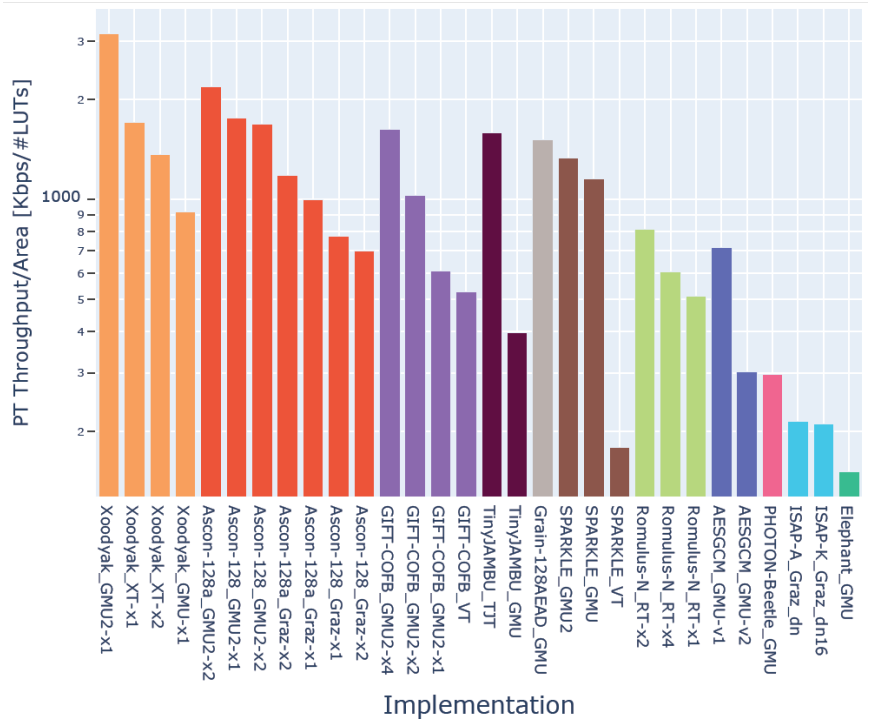
Unprotected AEAD implementations of TinyJAMBU, ASCON, GIFT-COFB, and SPARKLE processed plaintext faster than AES-GCM. TinyJAMBU had the most compact implementation, followed by Romulus. ASCON and Xoodyak had the highest throughput.

For hashing, unprotected implementations of Xoodyak and ASCON had the highest throughput. Xoodyak had the most compact implementation using fewer than 1400 LUTs.

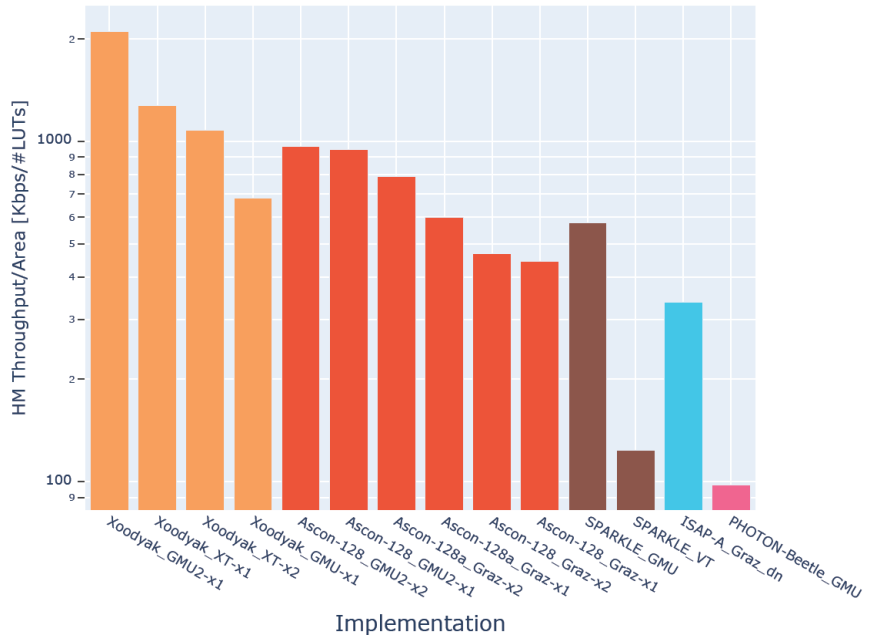
4.2.2. Hardware Benchmarking Results from Round 2

This section briefly summarizes hardware benchmarking efforts during the second round of evaluation and their relevance in the final round. Further details can be found in the second round status report [6].

GMU CERG [281] provided performance results for second-round implementations using their tool suite: ATHENa [284], Minerva [285], and Xeda [286]. This effort evaluated 27 of the second-round candidates on three FPGA platforms. Each of the candidates was ranked in terms of throughput and energy per bit. Round 2 implementations ASCON, Xoodyak, and GIFT-COFB were the top performers for AEAD throughput on the Artix-7, Intel Cyclone 10 LP, and Lattice ECP5. TinyJAMBU had the most compact AEAD



(a) PT Throughput/Area



(b) Hashing Throughput/Area

Fig. 4. Throughput/Area of unprotected hardware implementations [283]

implementations. Of the finalists, Xoodyak and ASCON had the smallest implementations supporting both AEAD and hashing. Authenticated encryption and decryption with Xoodyak, ASCON, and GIFT-COFB used the least energy-per-bit.

Khairallah et al. [287, 288] synthesized second round candidates on 65nm and 28nm technologies, primarily considering two use cases. The first was performance efficiency, where throughput/area ratio is the main concern. The second case was lightweight protocols, where Bluetooth and Bluetooth Low-Energy were selected as representatives of such protocols. Implementations were synthesized according to four different goals: balanced, low-area, high-speed, and low-frequency.

TinyJAMBU and Romulus had the smallest implementations and strong performance when optimized for low area, however, with both finalists containing a tweak that increases the number of rounds, throughput would decrease, area would remain relatively constant, and energy use would increase.

Aagaard and Zidarič [289] synthesized second round candidates using five different ASIC cell libraries and two synthesis tools. The cell libraries target 65nm, 90nm, and 130nm technologies. The study provides ratios of throughput to area, throughput to energy, and throughput to area $\times$ energy. Results were presented as average scaled values taken across the different configurations. Throughput results reflect steady-state plaintext processing and do not include the cost of loading the key, state initialization, or AD processing. Hashing was not evaluated in this study.

TinyJAMBU and Romulus had the smallest footprints, but at a cost of relatively poor performance in terms of throughput. Referring to the tweaks for TinyJAMBU and Romulus, both footprints would remain relatively constant, with performance degrading. Xoodyak and ASCON had the most energy-efficient implementations.

4.2.3. Additional Results

The following studies provide additional hardware benchmarking results.

- Khan et al. [290] implemented the hash functions of ASCON, PHOTON-Beetle, SPARKLE, and Xoodyak in FPGAs with performance goals of high throughput, area-time efficiency, and low hardware area. Authors presented various improvements utilizing the flexibility of the designs.
- Elsadek et al. [291] synthesized all ten finalists using a cell library that targets 22nm CMOS technology. Finalists were compared in terms of throughput, area, and energy efficiency for inputs consisting of plaintext (PT) only, AD only, and PT with AD. Two input sizes, 16 bytes and 1536 bytes, were used. The study considered throughput and energy efficiency of each finalist. Algorithms were grouped into three sets, where TinyJAMBU, Xoodyak, and ASCON are listed in the most energy efficient set. The authors conclude that Xoodyak, TinyJAMBU and ASCON are the most suitable algorithms for

burst short message and continuous long message scenarios.

- Elsadek et al. [292] applied parallelism to Elephant and ISAP, showing that the throughput can be increased up to 96% and 44%, respectively. Authors showed that the increase in throughput improves energy efficiency for parallelized architectures compared to iterative looping architectures for long messages, resulting in energy improvement up to 48.56% in Elephant and up to 27.58% in ISAP.

The following papers provide additional results on the hardware performance of ASCON.

- Gross et al. [293] presented an ASCON hardware optimized implementation with a single unrolled round transformation that requires 7 kGE of chip area and can process up to 0.75 cycles per byte. They also provided a threshold implementation of ASCON that requires about 8 kGE.
- Kaur et al. [294] proposed error detection mechanisms for secure hardware implementations of ASCON.
- Khan et al. [290] presented FPGA optimized implementations of ASCON-Hash by computing multiple permutation rounds in one clock cycle in order to explore the trade-off between computation time and hardware area.

The following papers provide additional results on the hardware performance of GIFT-COFB.

- Caforio et al. [295] proposed a lightweight circuit for GIFT-COFB that occupies less than 1500 GE, demonstrated how the additional operations in the mode can be executed concurrently with GIFT itself to reduce the total latency, and proposed a first-order threshold implementation of GIFT-COFB.
- Zhong and Guin [296] presented a key-recovery attack on the 2-round partial unrolled hardware implementation of GIFT-COFB, where the adversary is granted access to the input and output of 2-round GIFT-128.

The following paper provides additional results on the hardware performance of Grain-128AEAD.

- Sönnerup et al. [297] implemented Grain-128AEAD using a 65nm library. They used various optimization techniques to achieve high-throughput implementation and low-power implementation.

The following paper provides additional results on the hardware performance of PHOTON-Beetle.

- Khan et al. [290] presented a compact FPGA implementation of PHOTON-Beetle-Hash in which the underlying matrix multiplication was executed in a serialized fashion to achieve a small hardware footprint.

The following paper provides additional results on the hardware performance of SPARKLE.

- Khan et al. [290] presented FPGA implementations of the ESCH family of hash functions by implementing the ARX-box in serialized, parallelized, and hybrid approaches.

The following papers provide additional results on the hardware performance of Xoodyak.

- Wakeland [298] provided ASIC benchmark results that showed that Xoodyak is capable of higher throughput than AES-128 while using a lower cell area when they were obtained from builds using an 5nm and 16nm ASIC technology.
- Khan et al. [290] presented FPGA optimized implementations of Xoodyak hash by computing multiple permutation rounds in one clock cycle in order to explore the trade-off between computation time and hardware area.

4.3. Resistance to Side-Channel and Fault Attacks

In many applications, there is a need for protection against side-channel and fault attacks, and it is of particular interest in cases where constraints on implementation size limit the ability to apply countermeasures effectively. To this end, it is important to consider how employing several common countermeasures affects the size and performance of candidate implementations, especially in constrained devices. This section summarizes the results on side-channel and fault attacks that were considered during the evaluation.

4.3.1. Protected Implementations and Side-Channel Security Evaluations

The development, evaluation, and benchmarking of protected implementations requires a tremendous amount of time and expertise, making it difficult for a single group to perform on its own. Researchers at GMU organized a study of protected implementations that pooled the resources and expertise of several groups in order to make such a study possible. This combined effort included protected implementations that were generated manually and using automated tools for several orders of protection.

This section describes the development and evaluation of protected hardware and software implementations. It also summarizes the results of FPGA benchmarks on protected AEAD and hash implementations.

In January 2022, GMU published three calls:

- Call for Protected Hardware Implementations, targeting low-cost modern FPGAs [299],
- Call for Protected Software Implementations, targeting low-cost modern embedded processors [300], and
- Call for Side-Channel Security Evaluation Labs [301].

In response to these calls, GMU received the following protected implementations:

- *Protected software implementations* for ISAP [302] (using mode-level robustness against physical attacks), ASCON [303] (using masking, rotation of shares, and mode-level security), GIFT-COFB [304] (using Boolean masking), Romulus [305] (using Boolean masking), Xoodyak [306] (using ISW scheme).
- *Protected hardware implementations* for ISAP [307] (using mode-level robustness against physical attacks), all finalists except Grain-128AEAD [308] (using hardware private circuits), ASCON, Elephant, TinyJAMBU and Xoodyak [309] (using domain oriented masking), Xoodyak [306] (using threshold implementation).

Six software evaluation labs [310–315] and eight hardware evaluation labs [310, 311, 313–317] applied to evaluate protected implementations.

The number of protected software implementations limited the study to five finalists with implementations. Additionally, the implementations of two candidates, GIFT-COFB and

Table 13. T-Tests for hardware implementations [282]

Finalist	Design team	Implementation team	Protected implementation of order 1
ASCON	Tsinghua [318]	M: Graz	Passed for 10M traces
	Graz [319]	A: Bochum	Passed for 10M traces
	Shanghai Jiao Tong [320]	A: Bochum	Passed for 1M traces
	GMU [321]	A: Bochum	Failed for 1.5M traces with clock synchronization
Elephant	Graz [319]	A: Bochum	Passed for 10M traces
	GMU [322]	A: Bochum	Failed for 21k traces with clock synchronization
Grain-128AEAD	Protected implementation not available		
GIFT-COFB	Graz [319]	A: Bochum	Passed for 10M traces
	Shanghai Jiao Tong [323]	A: Bochum	Passed for 1M traces
ISAP	T-test not applicable to the mode-protected implementation		
PHOTON-Beetle	GMU [324]	A: Bochum	Passed for 10M traces with clock synchronization
Romulus	Graz [319]	A: Bochum	Passed for 10M traces
	Shanghai Jiao Tong [325]	A: Bochum	Failed for 1M traces
SPARKLE	Not tested by any lab		
TinyJAMBU	Tsinghua [326]	M: GMU	Passed for 10M traces
	GMU [327]	A: Bochum	Passed for 10M traces with clock ynchronization
Xoodyak	Secure-IC [328]	M: GMU	Passed for 100K traces
	Graz [319]	A: Bochum	Passed for 10M traces
	Tsinghua [329]	A: Bochum	Passed for 10M traces
	GMU [330]	A: Bochum	Failed for 1.5M traces with clock synchronization

Table 14. T-Tests and CPA for software implementations [282]

Finalist	Design team	Implementation team	Protected implementation of order 1
ASCON	Shanghai Jiao Tong [320]	M: Graz	Passed t-test for 60k traces
	Radboud [331]	M: Graz	Passed CPA for 15M traces
GIFT-COFB	Tsinghua [332]	M: Alexandre Adomnicai	Failed t-test for 100k traces
	Shanghai Jiao Tong [323]	M: Alexandre Adomnicai:	Passed t-test for 20k traces
ISAP	T-test not applicable to the mode-protected implementation		
Romulus	Tsinghua [333]	M: Alexandre Adomnicai	Failed t-test for 100k traces
	Shanghai Jiao Tong [325]	M: Alexandre Adomnicai	Passed t-test for 100k traces

Romulus, failed basic leakage assessment, while the mode-level protection of ISAP could not be verified experimentally. Robust software implementations were developed only for ASCON and Xoodyak. Protected implementations of ASCON were evaluated and passed all leakage assessment and attack attempts. Xoodyak was not evaluated by any lab.

Hardware benchmarking of protected implementations was performed by GMU on the Xilinx Artix-7 platform. Protected Xoodyak and ASCON implementations were the top performers in most tested cases. It is important to note that ISAP provides mode-level leakage protection, but comparing the strength of built-in protection against the first-, second-, and third-order protected implementations of the other candidates was not part of the study. The mode-level protection did not fully protect against simple power analysis.

First-order protection. First-order protected implementations of Xoodyak and ASCON performed well in terms of throughput and throughput over area. ASCON and TinyJAMBU had the best results when considering the size of first-order protected implementations com-

pared to the size of unprotected implementations. ISAP, Xoodyak, and ASCON achieved the most favorable results for random bits required for each byte of message. When considering the number of random bits required for each byte of AD, the top three candidates were ISAP, TinyJAMBU, and Xoodyak. Elephant, SPARKLE, and PHOTON-Beetle had the least favorable results for first-order protected hardware implementations.

Second-order protection. ISAP, ASCON, and TinyJAMBU implementations had the best results for AEAD benchmarks; ASCON and Xoodyak performed best in hashing. Benchmarks for hashing were obtained for only three candidates – ASCON, Xoodyak, and PHOTON-Beetle. Results for ASCON and Xoodyak were comparable, while those for PHOTON-Beetle were far less favorable.

Third-order protection. As with first- and second-order protected implementations, ISAP, ASCON, and TinyJAMBU were the leading AEAD candidates across most metrics. Hashing results were obtained for ASCON, Xoodyak, and ISAP; however, the ISAP submission did not have an official hash variant and recommended pairing ISAP-A variants with the corresponding hash function from ASCON. Hashing benchmarks for ASCON and Xoodyak were similar.

4.3.2. Fault Attacks

While side-channel attacks rely on observation of auxiliary information during normal operation, fault attacks rely on information obtained during abnormal operation. These attacks are carried out by forcing extreme conditions to induce errors in the computation. The manner in which errors propagate or how they are handled may allow an attacker to learn information about the key or internal state that would otherwise be unavailable.

Karl and Gruber [334] provided a survey on the application of fault analysis to the finalists. Madusham et al. [335] provided an overview of the underlying primitives of the finalists and reviewed a number of fault attacks against these finalists, including ciphertext-only fault analysis, fault intensity map analysis, and differential fault attacks.

The following papers are on the fault analysis of ASCON.

- Ramezanpour et al. [336] presented passive and active side-channel attacks on the lightweight implementation of ASCON on Atrix-7 FPGA. The attack recovers two bits of the secret key of ASCON by using 280 output authentication tags under fault injection into a pair of S-boxes. Their power analysis attack based on a deep learning technique recovered the full secret key by using 24K power traces during S-box computations at the beginning of the initialization phase of ASCON.
- Jacob et al. [337] showed how to modify ASCON-128a exploiting the pseudo-random properties of cellula automata to prevent statistical ineffective fault attack (SIFA) and subset fault analysis (SSFA).
- Surya et al. [338] proposed a local clock glitching fault injection attack on ASCON-128.

- Ramezanpour et al. [339] provided a statistical fault attack on ASCON, and they [340] also introduced SCARL (Side-Channel Analysis with Reinforcement Learning) capable of extracting data-dependent features of the measurements in an unsupervised learning approach without requiring a prior knowledge on the leakage model. SCARL can recover the secret key of ASCON-128 using 24K power traces during the key insertion or initialization stage, on a lightweight implementation on the Artix-7 FPGA.
- Joshi and Mazumdar [341] presented a SSFA to recover a 128-bit key of full-round ASCON-128 with the complexity of 2^{64}.

The following papers are on the fault analysis of Elephant.

- Joshi and Mazumdar [342] presented a fault attack on Elephant that recovers the secret key of Dumbo using 85 to 250 ciphertexts.
- Ambili et al. [343] proposed methods to strengthen TinyJAMBU and Elephant against DFA and interpolation attacks using the Cellular Automata.

The following papers are on the fault analysis of GIFT-COFB.

- Luo et al. [344] presented a general differential fault attack on GIFT by injecting a nibble fault before S-box operation in 25th to 28th rounds. Their attack recovered the secret key with 64 nibble fault ciphertexts, the time complexity of $2^{11.91}$, and the data complexity of 2^9.
- Liu et al. [345] presented a fault key-recovery attack on GIFT-COFB with 64 faulty ciphertexts by using the collision fault attack on GIFT-128.

The following papers are on the fault analysis of Grain-128AEAD.

- Salam et al. [346] presented various differential fault attacks on Grain-128AEAD in particular the bit-flipping fault attack that required access to $2^{7.80}$ faulty outputs to recover the initial state, the probabilistic random fault attack that required access to $2^{11.60}$ faulty outputs and $2^{10.45}$ fault injections to recover the initial state, the deterministic random fault attack with a precise control that required an average of $2^{7.64}$ fault injections and a data complexity of $2^{8.80}$, and the deterministic random fault attack with moderate control that requires about $2^{9.39}$ fault injections with a data complexity of about $2^{12.98}$.

The following papers are on the fault analysis of ISAP.

- Dobraunig et al. [347] presented two leakage resilience results relating to ISAP, how the mode affects concrete power analysis and fault attacks, and the performance of ISAP in different use cases.

The following papers are on the fault analysis of PHOTON-Beetle.

- Jana and Paul [348] presented two differential fault attacks on PHOTON-Beetle. The first is a random fault attack that requires around $2^{37.15}$ random queries with time and memory

complexities of 2^{16} and 2^{10} nibbles, respectfully. The second attack uses a known fault that requires around $2^{11.05}$ queries with 2^{11} time and 2^9 memory complexities.

The following papers are on the fault analysis of Romulus.

- Vafaei et al. [349] showed that roughly 10 random nibble/byte fault injections is sufficient to extract the master key of Skinny-n-n, Skinny-n-$2n$ (for $n = 64$ and 128).

The following papers are on the fault analysis of TinyJAMBU.

- Ambili et al. [343] proposed methods to strengthen TinyJAMBU and Elephant against DFA and interpolation attacks using the cellular automata.

The following papers are on the fault analysis of Xoodyak.

- Miteloudi et al. [350] proposed a new countermeasure against fault analysis attacks and implemented an FPGA-oriented protected version of Xoodyak to demonstrate the hardware overhead of the proposed countermeasure.

4.3.3. Additional Results

Some of the additional results on protected implementations are provided below.

- Bellizia et al. [351] considered the nonce misuse-resilient CCA security with a unique challenge nonce (called CCAm security) and the nonce misuse-resistant Ciphertext Integrity (called CIM security) in the two leakage models, L1 (encryption leakage only) and L2 (both encryption and decryption leakages). Authors showed that the achievable securities of leveled implementations of the finalists PHOTON-Beetle, ASCON, and ISAP are CCAL1+CIL1, CCAmL1+CIML2 and CCAmL2+CIML2, respectively.
- Bhasin et al. [352] conducted side-channel analysis on LFSR/NFSR based AEAD schemes, Grain-128AEAD and TinyJAMBU, using Differential Analysis aided Power Attack (DAPA). DAPA could recover the full key from 1-bit implementations but not from 32-bit implementations.
- Verhamme et al. [353] provided the performance result of the leveled implementation of Romulus-T, ASCON and ISAP, and conclude that all of these finalists improve over AES and the results are sensitive to their security margins.
- Diehl et al. [354] presented two types of side-channel resistant FPGA implementations of ASCON: ASCON-small and ASCON-large, where 3-share threshold protected implementations were chosen to make them resistant to first-order DPA. Compared to the protected implementation of AES-GCM, ASCON-large has 83% of the area of AES-GCM and 2.5 times the throughput of AES-GCM. ASCON-small has 74% of the area and slightly greater than throughput of AES-GCM.
- Abdulgadir et al. [355] compared the cost of first-order protection of domain-oriented masking. Their benchmarking showed that the protected design of Elephant occupies

5451 LUTs and has a throughput of 93 Mbps when implemented on Xilinx Artix-7 FPGAs.

- Aljuffri et al. [356] showed that GIFT's s-box (or SubCell function) is vulnerable to profiled and non-profiled attacks when unprotected or protected implementations based on existing balancing or masking techniques are used. They proposed a new countermeasure that smartly combines balancing and masking to offer full protection with negligible overhead.
- Reinbrecht et al. [357] presented a cache attack on GIFT referred to as GRINCH. Their attack recovered the full key within 400 encryptions.
- Dobraunig et al. [358] presented an outline on the applicability of SPA/TA attacks on the cryptographic constructions and introduced a co-processor that implements ASCON's permutation to speed up ASCON/ISAP while increasing protection against SPA and TA.
- Khairallah and Bhasin [359, 360] presented the hardware implementations of Skinny using various masking schemes and provided the implementation results of Romulus with the first-order masked Skinny 8-bit SBoxes.
- Abdulgadir et al. [355, 361] compared the cost of first-order protection of domain-oriented masking. Their benchmarking showed that the protected design of TinyJAMBU occupies 1267 LUTs and has a throughput of 120 Mbps when implemented on Xilinx Artix-7 FPGAs.
- Abdulgadir et al. [355, 361] compared the cost of first-order protection of domain-oriented masking. Their benchmarking showed that the protected designs of Xoodyak occupies 6431 LUTs and has a throughput of 891 Mbps when implemented on Xilinx Artix-7 FPGAs.

5. Next Steps

In June 2023, NIST will host the Sixth Lightweight Cryptography Workshop to further explain the selection process and to discuss various aspects of standardization. Among the topics of interest are additional variants, functionalities, and parameter selection. There has been public interest in possible extensions to the scope of the lightweight cryptography project. In particular, the community has expressed interest in the development of MAC and deterministic random bit generator standards based on the ASCON permutation.

NIST will work with the ASCON designers to draft the new lightweight cryptography standard. There will be a public comment period of at least 45 days during which NIST will solicit public feedback on the draft and publish the comments that were received. NIST will address each of the comments by making minor edits to the document or noting issues raised for future consideration.

The final version of NIST's ASCON standard will be published shortly after public com-

ments have been resolved. At this time, the validation tests and procedures will be developed, followed by inclusion in validation processes under the cryptographic algorithm validation program and cryptographic module validation program.

References

[1] National Institute of Standards and Technology (2001) Advanced Encryption Standard (AES) (U.S. Department of Commerce), Report. https://doi.org/10.6028/NIST.FIPS.197-upd1

[2] Dworkin MJ (2007) Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC (National Institute of Standards and Technology), Report. https://doi.org/10.6028/NIST.SP.800-38D

[3] National Institute of Standards and Technology (2015) Secure Hash Standard (SHS) (U.S. Department of Commerce), Report. https://doi.org/10.6028/NIST.FIPS.180-4

[4] NIST (2018) Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf.

[5] Sönmez Turan M, McKay KA, Çalık Ç, Chang D, Bassham I Lawrence E (2019) Status Report on the First Round of the NIST Lightweight Cryptography Standardization Process (National Institute of Standards and Technology), Report. https://doi.org/10.6028/NIST.IR.8268

[6] Sönmez Turan M, McKay KA, Chang D, Çalık Ç, Bassham I Lawrence E, Kang J, Kelsey J (2021) Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process (National Institute of Standards and Technology), Report. https://doi.org/10.6028/NIST.IR.8369

[7] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2019) Ascon v1.2, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[8] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2019) Ascon v1.2, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[9] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2021) Ascon v1.2, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[10] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2020) Status Update on Ascon v1.2, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/ascon_update.pdf.

[11] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2022) Status Update on Ascon v1.2, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/ascon-update.pdf.

[12] Beyne T, Chen YL, Dobraunig C, Mennink B (2019) Elephant v1, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[13] Beyne T, Chen YL, Dobraunig C, Mennink B (2019) Elephant v1.1, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[14] Beyne T, Chen YL, Dobraunig C, Mennink B (2021) Elephant v2, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projec

ts/lightweight-cryptography/finalists.

[15] Beyne T, Chen YL, Dobraunig C, Mennink B (2020) Status Update on Elephant, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/Elephant_status-update-round-2.pdf.

[16] Beyne T, Chen YL, Dobraunig C, Mennink B (2022) Status Update on Elephant, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/elephant-update.pdf.

[17] Banik S, Chakraborti A, Iwata T, Minematsu K, Nandi M, Peyrin T, Sasaki Y, Sim SM, Todo Y (2019) GIFT-COFB v1.0, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[18] Banik S, Chakraborti A, Iwata T, Minematsu K, Nandi M, Peyrin T, Sasaki Y, Sim SM, Todo Y (2019) GIFT-COFB v1.0, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[19] Banik S, Chakraborti A, Iwata T, Minematsu K, Nandi M, Peyrin T, Sasaki Y, Sim SM, Todo Y (2021) GIFT-COFB v1.1, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[20] Banik S, Chakraborti A, Iwata T, Minematsu K, Nandi M, Peyrin T, Sasaki Y, Sim SM, Todo Y (2020) GIFT-COFB: NIST LWC Second-round Candidate Status Update, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/GIFT-COFB_status_update.pdf.

[21] Banik S, Chakraborti A, Inoue A, Iwata T, Minematsu K, Nandi M, Peyrin T, Sasaki Y, Sim SM, Todo Y (2022) GIFT-COFB Final Round Updates, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/gift-cofb-update.pdf.

[22] Hell M, Johansson T, Meier W, Sönnerup J, Yoshida H (2019) Grain-128AEAD - A lightweight AEAD stream cipher, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[23] Hell M, Johansson T, Meier W, Sönnerup J, Yoshida H (2019) Grain-128AEAD - A lightweight AEAD stream cipher, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[24] Hell M, Johansson T, Maximov A, Meier W, Sönnerup J, Yoshida H (2021) Grain-128AEADv2 - A lightweight AEAD stream cipher, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[25] Hell M, Johansson T, Maximov A, Meier W, Sönnerup J, Yoshida H (2020) Grain-128AEAD - Status Document, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/Grain_128AEAD_status_document.pdf.

[26] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2019) ISAP v2.0, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[27] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2019) ISAP v2.0, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[28] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2021) ISAP v2.0, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[29] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2020) NIST Update: ISAP v2.0, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/ISAP_update_isap.pdf.

[30] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2022) NIST Update: ISAP v2.0, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/isap-update.pdf.

[31] Bao Z, Chakraborti A, Datta N, Guo J, Nandi M, Peyrin T, Yasuda K (2019) PHOTON-Beetle Authenticated Encryption and Hash Family, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[32] Bao Z, Chakraborti A, Datta N, Guo J, Nandi M, Peyrin T, Yasuda K (2019) PHOTON-Beetle Authenticated Encryption and Hash Family, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[33] Bao Z, Chakraborti A, Datta N, Guo J, Nandi M, Peyrin T, Yasuda K (2021) PHOTON-Beetle Authenticated Encryption and Hash Family, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[34] Bao Z, Chakraborti A, Datta N, Guo J, Nandi M, Peyrin T, Yasuda K (2020) PHOTON-Beetle Authenticated Encryption and Hash Family — Updated on Software Implementations, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/PHOTON-Beetle_software_update_18Sep2020.pdf.

[35] Iwata T, Khairallah M, Minematsu K, Peyrin T (2019) Romulus v1.0, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[36] Iwata T, Khairallah M, Minematsu K, Peyrin T (2019) Romulus v1.2, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[37] Guo C, Iwata T, Khairallah M, Minematsu K, Peyrin T (2021) Romulus v1.3, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[38] Iwata T, Khairallah M, Minematsu K, Peyrin T (2020) Romulus for Round 3, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/Romulus-for-Round-3.pdf.

[39] Guo C, Iwata T, Khairallah M, Minematsu K, Peyrin T (2022) Final-round updates on Romulus, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/romulus-update.pdf.

[40] Beierle C, Biryukov A, dos Santos LC, Großschädl J, Perrin L, Udovenko A, Velichkov V, Wang Q (2019) Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family (Version 1.0), Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[41] Beierle C, Biryukov A, dos Santos LC, Großschädl J, Perrin L, Udovenko A, Velichkov V, Wang Q (2019) Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family (Version 1.1), Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[42] Beierle C, Biryukov A, dos Santos LC, Großschädl J, Moradi A, Perrin L, Shahmirzadi AR, Udovenko A, Velichkov V, Wang Q (2021) Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family (Version 1.2), Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[43] Beierle C, Biryukov A, dos Santos LC, Großschädl J, Perrin L, Udovenko A, Velichkov V, Wang Q (2020) An Update on the Sparkle Suite, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/SPARKLE_update.pdf.

[44] Beierle C, Biryukov A, dos Santos LC, Großschädl J, Moradi A, Perrin L, Shahmirzadi AR, Udovenko A, Velichkov V, Wang Q (2022) An Update on the LWC Finalist Sparkle, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/sparkle-update.pdf.

[45] Wu H, Huang T (2019) TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[46] Wu H, Huang T (2019) TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[47] Wu H, Huang T (2021) TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms (Version 2), Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[48] Wu H, Huang T (2020) TinyJAMBU Update, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/TinyJambu_update_20200918.pdf.

[49] Wu H, Huang T (2022) TinyJAMBU Update, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/tinyjambu-update.pdf.

[50] Daemen J, Hoffert S, Peeters M, Assche GV, Keer RV (2019) Xoodyak, a lightweight cryptographic scheme, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-1-candidates.

[51] Daemen J, Hoffert S, Peeters M, Assche GV, Keer RV (2019) Xoodyak, a lightweight cryptographic scheme, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/round-2-candidates.

[52] Daemen J, Hoffert S, Mella S, Peeters M, Assche GV, Keer RV (2021) Xoodyak, a lightweight cryptographic scheme, Submission to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/Projects/lightweight-cryptography/finalists.

[53] Daemen J, Hoffert S, Mella S, Peeters M, Assche GV, Keer RV (2020) Xoodyak, an update, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/status-update-sep2020/Xoodyak-update.pdf.

[54] Daemen J, Hoffert S, Mella S, Peeters M, Assche GV, Keer RV (2022) Xoodyak, a final update, Update to the NIST Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/csrc/media/Projects/lightweight-cryptography/documents/finalist-round/status-updates/xoodyak-update.pdf.

[55] McKay KA, Bassham I Lawrence E, Sönmez Turan M, Mouha N (2017) Report on Lightweight Cryptography (National Institute of Standards and Technology), Report. https://doi.org/10.6028/NIST.IR.8114

[56] Bassham L, Çalık Ç, McKay K, Mouha N, Sönmez Turan M (2017) Profiles for the Lightweight Cryptography Standardization Process. Available at https://csrc.nist.gov/CSRC/media/Publications/white-paper/2017/04/26/profiles-for-lightweight-cryptography-standardization-process/draft/documents/profiles-lwc-std-proc-draft.pdf.

[57] NIST (2018) Federal Register Notice. Available at https://www.govinfo.gov/content/pkg/FR-2018-08-27/pdf/2018-18433.pdf.

[58] Chang D, Hong D, Kang J (2022) Conditional Cube Attacks on Ascon-128 and Ascon-80pq in a Nonce-misuse Setting, Cryptology ePrint Archive, Paper 2022/544. Available at https://eprint.iacr.org/2022/544.

[59] Chang D, Datta N, Dutta A, Mennink B, Nandi M, Sanadhya S, Sibleyras F (2019) Release of Unverified Plaintext: Tight Unified Model and Application to ANYDAE. *IACR Trans Symmetric Cryptol* 2019(4):119–146. https://doi.org/10.13154/tosc.v2019.i4.119-146

[60] Bernstein DJ, Gilbert H, Sönmez Turan M (2020) Observations on COMET, Cryptology ePrint Archive, Paper 2020/1445. Available at https://eprint.iacr.org/2020/1445.

[61] Chang D, Sönmez Turan M (2021) Recovering the Key from the Internal State of Grain-128AEAD, Cryptology ePrint Archive, Paper 2021/439. Available at https://eprint.iacr.org/2021/439.

[62] Chang D, Kang J, Turan MS (2022) A New Conditional Cube Attack on Reduced-Round Ascon-128a in a Nonce-misuse Setting, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/Presentations/2022/a-new-conditional-cube-attack-on-reduced-round-asc.

[63] Chang D, Hong D, Kang J, Turan MS (2023) Resistance of Ascon Family Against Conditional Cube Attacks in Nonce-Misuse Setting. *IEEE Access* 11:4501–4516. https://doi.org/10.1109/ACCESS.2022.3223991

[64] Naito Y (2019) Optimally Indifferentiable Double-Block-Length Hashing Without Post-processing and with Support for Longer Key Than Single Block. *Progress in Cryptology - LATINCRYPT 2019 - 6th International Conference on Cryptology and Information Security in Latin America, Santiago de Chile, Chile, October 2-4, 2019, Proceedings*, eds Schwabe P, Thériault N (Springer), *Lecture Notes in Computer Science*, Vol. 11774, pp 65–85. https://doi.org/10.1007/978-3-030-30530-7_4

[65] Hirose S (2006) Some Plausible Constructions of Double-Block-Length Hash Functions. *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, ed Robshaw MJB (Springer), *Lecture Notes in Computer Science*, Vol. 4047, pp 210–225. https://doi.org/10.1007/11799313_14

[66] Rohit R, Hu K, Sarkar S, Sun S (2021) Misuse-Free Key-Recovery and Distinguishing Attacks on 7-Round Ascon. *IACR Trans Symmetric Cryptol* 2021(1):130–155. https://doi.org/10.46586/tosc.v2021.i1.130-155

[67] Bogdanov A, Knezevic M, Leander G, Toz D, Varıcı K, Verbauwhede I (2013) SPONGENT: The Design Space of Lightweight Cryptographic Hashing. *IEEE Trans Computers* 62(10):2041–2053. https://doi.org/10.1109/TC.2012.196

[68] Zhang G, Liu M (2016) A Distinguisher on PRESENT-Like Permutations with Application to SPONGENT, Cryptology ePrint Archive, Paper 2016/236. Available at https://eprint.iacr.org/2016/236.

[69] Zhou H, Zong R, Dong X, Jia K, Meier W (2020) Interpolation Attacks on Round-Reduced Elephant, Kravatte and Xoofff, Cryptology ePrint Archive, Paper 2020/781. Available at https://eprint.iacr.org/2020/781.

[70] Zong R, Dong X, Chen H, Luo Y, Wang S, Li Z (2021) Towards Key-recovery-attack Friendly Distinguishers: Application to GIFT-128. *IACR Trans Symmetric Cryptol* 2021(1):156–184. https://doi.org/10.46586/tosc.v2021.i1.156-184

[71] He J, Hu K, Preneel B, Wang M (2022) Stretching Cube Attacks: Improved Methods to Recover Massive Superpolies, Cryptology ePrint Archive, Paper 2022/1218. Available at https://eprint.iacr.org/2022/1218.

[72] Qin L, Hua J, Dong X, Yan H, Wang X (2022) Meet-in-the-Middle Preimage Attacks on Sponge-based Hashing, Cryptology ePrint Archive, Paper 2022/1714. Available at https://eprint.iacr.org/2022/1714.

[73] The Keccak Crunchy Crypto Collision and Pre-image Contest. Available at https://keccak.team/crunchy_contest.html.

[74] Cui T, Sun L, Chen H, Wang M (2017) Statistical Integral Distinguisher with Multi-structure and Its Application on AES. *Information Security and Privacy - 22nd Australasian Conference, ACISP 2017, Auckland, New Zealand, July 3-5, 2017, Proceedings, Part I*, eds Pieprzyk J, Suriadi S (Springer), *Lecture Notes in Computer Science*, Vol. 10342, pp 402–420. https://doi.org/10.1007/978-3-319-60055-0_21

[75] Dong X, Qin L, Sun S, Wang X (2022) Key Guessing Strategies for Linear Key-Schedule Algorithms in Rectangle Attacks. *Advances in Cryptology - EUROCRYPT 2022 - 41st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Trondheim, Norway, May 30 - June 3, 2022, Proceedings, Part III*, eds Dunkelman

O, Dziembowski S (Springer), *Lecture Notes in Computer Science*, Vol. 13277, pp 3–33. https://doi.org/10.1007/978-3-031-07082-2_1

[76] Song L, Zhang N, Yang Q, Shi D, Zhao J, Hu L, Weng J (2022) Optimizing Rectangle Attacks: A Unified and Generic Framework for Key Recovery, Cryptology ePrint Archive, Paper 2022/723. Available at https://eprint.iacr.org/2022/723.

[77] Dutta P, Rajasree MS, Sarkar S (2022) Weak-keys and Key-recovery Attack for TinyJAMBU. *Scientific Reports* https://doi.org/10.1038/s41598-022-19046-2

[78] Dunkelman O, Lambooij E, Ghosh S (2022) Practical Related-Key Forgery Attacks on the Full TinyJAMBU-192/256, Cryptology ePrint Archive, Paper 2022/1122. Available at https://eprint.iacr.org/2022/1122.

[79] Zhou H, Li Z, Dong X, Jia K, Meier W (2020) Practical Key-Recovery Attacks On Round-Reduced Ketje Jr, Xoodoo-AE And Xoodyak. *Comput J* 63(8):1231–1246. https://doi.org/10.1093/comjnl/bxz152

[80] Grover LK (1996) A Fast Quantum Mechanical Algorithm for Database Search. *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing, Philadelphia, Pennsylvania, USA, May 22-24, 1996*, ed Miller GL (ACM), pp 212–219. https://doi.org/10.1145/237814.237866

[81] Kuwakado H, Morii M (2012) Security on the Quantum-type Even-Mansour Cipher. *Proceedings of the International Symposium on Information Theory and its Applications, ISITA 2012, Honolulu, HI, USA, October 28-31, 2012* (IEEE), pp 312–316. Available at http://ieeexplore.ieee.org/document/6400943/.

[82] Bonnetain X, Jaques S (2020) Quantum Period Finding against Symmetric Primitives in Practice, Cryptology ePrint Archive, Report 2020/1418. Available at https://eprint.iacr.org/2020/1418.

[83] Gookyi DAN, Kanda G, Ryoo K (2021) NIST Lightweight Cryptography Standardization Process: Classification of Second Round Candidates, Open Challenges, and Recommendations. *J Inf Process Syst* 17(2):253–270. https://doi.org/10.3745/JIPS.03.0156

[84] Naseer M, Tariq S, Riaz N (2022) Substitution Layer Analysis of NIST Lightweight Cryptography Competition Finalists. *2022 19th International Bhurban Conference on Applied Sciences and Technology (IBCAST)*, pp 659–664. https://doi.org/10.1109/IBCAST54850.2022.9990069

[85] Bertoni G, Daemen J, Peeters M, Assche GV (2012) Permutation-based Encryption, Authentication and Authenticated Encryption, DIAC – Directions in Authenticated Ciphers. Available at https://keccak.team/files/KeccakDIAC2012.pdf.

[86] Bertoni G, Daemen J, Peeters M, Assche GV (2011) Duplexing the Sponge: Single-Pass Authenticated Encryption and Other Applications. *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, eds Miri A, Vaudenay S (Springer), *Lecture Notes in Computer Science*, Vol. 7118, pp 320–337. https://doi.org/10.1007/978-3-642-28496-0_19

[87] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2016) Ascon v1.2, Submission to the CAESAR Competition. Available at https://ascon.iaik.tugraz.at/files/asconv12.pdf.

[88] Rohit R, Sarkar S (2021) Diving Deep into the Weak Keys of Round Reduced Ascon. *IACR Transactions on Symmetric Cryptology* 2021(4):74–99. https://doi.org/10.46586/tosc.v2021.i4.74-99

[89] Tezcan C (2019) Distinguishers for Reduced Round Ascon, DryGASCON, and Shamash

Permutations, NIST Lightweight Cryptography Workshop 2019. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2019/documents/papers/distinguishers-for-reduced-round-ascon-drygascon-shamash-lwc2019.pdf.

[90] Tezcan C (2020) Analysis of Ascon, DryGASCON, and Shamash Permutations, Cryptology ePrint Archive, Paper 2020/1458. Available at https://eprint.iacr.org/2020/1458.

[91] Li Z, Dong X, Wang X (2017) Conditional Cube Attack on Round-Reduced ASCON. *IACR Transactions on Symmetric Cryptology* 2017(1):175–202. https://doi.org/10.13154/tosc.v2017.i1.175-202

[92] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2015) Cryptanalysis of Ascon. *Topics in Cryptology - CT-RSA 2015, The Cryptographer's Track at the RSA Conference 2015, San Francisco, CA, USA, April 20-24, 2015. Proceedings*, ed Nyberg K (Springer), *Lecture Notes in Computer Science*, Vol. 9048, pp 371–387. https://doi.org/10.1007/978-3-319-16715-2_20

[93] Tezcan C (2016) Truncated, Impossible, and Improbable Differential Analysis of ASCON. *Proceedings of the 2nd International Conference on Information Systems Security and Privacy, ICISSP 2016, Rome, Italy, February 19-21, 2016*, eds Camp O, Furnell S, Mori P (SciTePress), pp 325–332. https://doi.org/10.5220/0005689903250332

[94] Dwivedi AD, Kloucek M, Morawiecki P, Nikolic I, Pieprzyk J, Wójtowicz S (2017) SAT-based Cryptanalysis of Authenticated Ciphers from the CAESAR Competition. *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications (ICETE 2017) - Volume 4: SECRYPT, Madrid, Spain, July 24-26, 2017*, eds Samarati P, Obaidat MS, Cabello E (SciTePress), pp 237–246. https://doi.org/10.5220/0006387302370246

[95] Gérault D, Peyrin T, Tan QQ (2021) Exploring Differential-Based Distinguishers and Forgeries for ASCON. *IACR Trans Symmetric Cryptol* 2021(3):102–136. https://doi.org/10.46586/tosc.v2021.i3.102-136

[96] Hu K, Peyrin T (2022) Revisiting Higher-Order Differential(-Linear) Attacks from an Algebraic Perspective – Applications to Ascon, Grain v1, Xoodoo, and ChaCha, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/revisiting-higher-order-differential-linear)-atacks-from-algebraic-perspective.pdf.

[97] Liu M, Lu X, Lin D (2021) Differential-Linear Cryptanalysis from an Algebraic Perspective. *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, eds Malkin T, Peikert C (Springer), *Lecture Notes in Computer Science*, Vol. 12827, pp 247–277. https://doi.org/10.1007/978-3-030-84252-9_9

[98] Halak B, Duarte-Sanchez J (2020) Cube Attack on a Trojan-Compromised Hardware Implementation of Ascon. *33rd IEEE International System-on-Chip Conference, SoCC 2020, Las Vegas, NV, USA, September 8-11, 2020* (IEEE), pp 43–47. https://doi.org/10.1109/SOCC49529.2020.9524771

[99] Li Y, Zhang G, Wang W, Wang M (2017) Cryptanalysis of Round-reduced ASCON. *Sci China Inf Sci* 60(3):38102. https://doi.org/10.1007/s11432-016-0283-3

[100] Baudrin J, Canteaut A, Perrin L (2022) Practical Cube-attack against Nonce-misused Ascon, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/practical-cube-attack-against-nonce-isused-ascon.pdf.

[101] Todo Y (2015) Structural Evaluation by Generalized Integral Property. *Advances in Cryptol-*

ogy - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I, eds Oswald E, Fischlin M (Springer), *Lecture Notes in Computer Science*, Vol. 9056, pp 287–314. https://doi.org/10.1007/978-3-662-46800-5_12

[102] Baksi A, Breier J, Chen Y, Dong X (2021) Machine Learning Assisted Differential Distinguishers For Lightweight Ciphers. *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021* (IEEE), pp 176–181. https://doi.org/10.23919/DATE51398.2021.9474092

[103] Dobraunig C, Eichlseder M, Mendel F (2015) Heuristic Tool for Linear Cryptanalysis with Applications to CAESAR Candidates. *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, eds Iwata T, Cheon JH (Springer), *Lecture Notes in Computer Science*, Vol. 9453, pp 490–509. https://doi.org/10.1007/978-3-662-48800-3_20

[104] Leander G, Tezcan C, Wiemer F (2018) Searching for Subspace Trails and Truncated Differentials. *IACR Trans Symmetric Cryptol* 2018(1):74–100. https://doi.org/10.13154/tosc.v2018.i1.74-100

[105] Erlacher J, Mendel F, Eichlseder M (2022) Bounds for the Security of Ascon against Differential and Linear Cryptanalysis. *IACR Transactions on Symmetric Cryptology* 2022(1):64–87. https://doi.org/10.46586/tosc.v2022.i1.64-87

[106] El Hirch S, Mella S, Mehrdad A, Daemen J (2022) Improved Differential and Linear Trail Bounds for ASCON. *IACR Transactions on Symmetric Cryptology* 2022(4):145–178. https://doi.org/10.46586/tosc.v2022.i4.145-178

[107] Sommervoll ÅÅ (2021) Dreaming of Keys: Introducing the Phantom Gradient Attack. *Proceedings of the 7th International Conference on Information Systems Security and Privacy, ICISSP 2021, Online Streaming, February 11-13, 2021*, eds Mori P, Lenzini G, Furnell S (SCITEPRESS), pp 619–627. https://doi.org/10.5220/0010317806190627

[108] Dobraunig C, Eichlseder M, Mendel F, Schläffer M (2019) Preliminary Analysis of Ascon-Xof and Ascon-Hash. *Technical Report* Available at https://ascon.iaik.tugraz.at/files/Preliminary_Analysis_of_Ascon-Xof_and_Ascon-Hash_v01.pdf.

[109] Zong R, Dong X, Wang X (2019) Collision Attacks on Round-Reduced Gimli-Hash/Ascon-Xof/Ascon-Hash, Cryptology ePrint Archive, Paper 2019/1115. Available at https://eprint.iacr.org/2019/1115.

[110] Liu F, Isobe T, Meier W (2020) Automatic Verification of Differential Characteristics: Application to Reduced Gimli. *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, eds Micciancio D, Ristenpart T (Springer), *Lecture Notes in Computer Science*, Vol. 12172, pp 219–248. https://doi.org/10.1007/978-3-030-56877-1_8

[111] Lefevre C, Mennink B (2022) Tight Preimage Resistance of the Sponge Construction, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/tight-preimage-resistance-of-the-sponge.pdf.

[112] Lee W, Jang K, Song G, Kim H, Hwang SO, Seo H (2022) Efficient Implementation of Lightweight Hash Functions on GPU and Quantum Computers for IoT Applications. *IEEE Access* 10:59661–59674. https://doi.org/10.1109/ACCESS.2022.3179970

[113] Bogdanov A, Knezevic M, Leander G, Toz D, Varıcı K, Verbauwhede I (2011) Spongent: A Lightweight Hash Function. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, eds Preneel B, Takagi T (Springer), *Lecture Notes in Computer Science*, Vol. 6917, pp 312–325. https://doi.org/10.1007/978-3-642-23951-9_21

[114] National Institute of Standards and Technology (2015) SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (U.S. Department of Commerce), Report. https://doi.org/10.6028/NIST.FIPS.202

[115] Vialar L (2022) Trumping the Elephant: Fast Side-Channel Key-Recovery Attack against Dumbo. *SSTIC2022* Available at https://www.sstic.org/media/SSTIC2022/SSTIC-actes/fast_side-channel_key_recovery_attack_against_elep/SSTIC2022-Article-fast_side-channel_key_recovery_attack_against_elephant-dumbo-vialar.pdf.

[116] Beyne T, Chen YL, Dobraunig C, Mennink B (2021) Multi-user Security of the Elephant v2 Authenticated Encryption Mode. *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, eds AlTawy R, Hülsing A (Springer), *Lecture Notes in Computer Science*, Vol. 13203, pp 155–178. https://doi.org/10.1007/978-3-030-99277-4_8

[117] Sun L, Wang W, Wang M (2020) MILP-aided Bit-based Division Property for Primitives with Non-bit-permutation Linear Layers. *IET Inf Secur* 14(1):12–20. https://doi.org/10.1049/iet-ifs.2018.5283

[118] Huang S, Ben-Yehuda OA, Dunkelman O, Maximov A (2022) Finding Collisions against 4-Round SHA-3-384 in Practical Time. *IACR Trans Symmetric Cryptol* 2022(3):239–270. https://doi.org/10.46586/tosc.v2022.i3.239-270

[119] He L, Lin X, Yu H (2022) Improved Preimage Attacks on Round-Reduced Keccak-384/512 via Restricted Linear Structures, Cryptology ePrint Archive, Paper 2022/788. Available at https://eprint.iacr.org/2022/788.

[120] Wei C, Wu C, Fu X, Dong X, He K, Hong J, Wang X (2021) Preimage Attacks on 4-round Keccak by Solving Multivariate Quadratic Systems, Cryptology ePrint Archive, Paper 2021/732. Available at https://eprint.iacr.org/2021/732.

[121] Lin X, He L, Yu H (2021) Improved Preimage Attacks on 3-Round Keccak-224/256. *IACR Trans Symmetric Cryptol* 2021(3):84–101. https://doi.org/10.46586/tosc.v2021.i3.84-101

[122] Boissier RH, Noûs C, Rotella Y (2021) Algebraic Collision Attacks on Keccak. *IACR Trans Symmetric Cryptol* 2021(1):239–268. https://doi.org/10.46586/tosc.v2021.i1.239-268

[123] Suryawanshi S, Saha D, Sachan S (2020) New Results on the SymSum Distinguisher on Round-Reduced SHA3. *Progress in Cryptology - AFRICACRYPT 2020 - 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20-22, 2020, Proceedings*, eds Nitaj A, Youssef AM (Springer), *Lecture Notes in Computer Science*, Vol. 12174, pp 132–151. https://doi.org/10.1007/978-3-030-51938-4_7

[124] Morawiecki P, Pieprzyk J, Srebrny M, Straus M (2013) Preimage Attacks on the Round-reduced Keccak with the Aid of Differential Cryptanalysis, Cryptology ePrint Archive, Paper 2013/561. Available at https://eprint.iacr.org/2013/561.

[125] Guo J, Liu M, Song L (2016) Linear Structures: Applications to Cryptanalysis of Round-Reduced Keccak. *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part I*, eds Cheon JH, Takagi T, *Lecture Notes in Com-*

puter Science, Vol. 10031, pp 249–274. https://doi.org/10.1007/978-3-662-53887-6_9

[126] Rajasree MS (2019) Cryptanalysis of Round-Reduced KECCAK Using Non-linear Structures. *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings*, eds Hao F, Ruj S, Gupta SS (Springer), *Lecture Notes in Computer Science*, Vol. 11898, pp 175–192. https://doi.org/10.1007/978-3-030-35423-7_9

[127] Li T, Sun Y (2019) Preimage Attacks on Round-Reduced Keccak-224/256 via an Allocating Approach. *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part III*, eds Ishai Y, Rijmen V (Springer), *Lecture Notes in Computer Science*, Vol. 11478, pp 556–584. https://doi.org/10.1007/978-3-030-17659-4_19

[128] Liu F, Isobe T, Meier W, Yang Z (2020) Algebraic Attacks on Round-Reduced Keccak/Xoodoo, Cryptology ePrint Archive, Paper 2020/346. Available at https://eprint.iacr.org/2020/346.

[129] He L, Lin X, Yu H (2021) Improved Preimage Attacks on 4-Round Keccak-224/256. *IACR Trans Symmetric Cryptol* 2021(1):217–238. https://doi.org/10.46586/tosc.v2021.i1.217-238

[130] Wang R, Li X, Gao J, Li H, Wang B (2022) Quantum Rotational Cryptanalysis for Preimage Recovery of Round-Reduced Keccak, Cryptology ePrint Archive, Paper 2022/013. Available at https://eprint.iacr.org/2022/013.

[131] Wang R, Li X, Gao J, Li H, Wang B (2022) Allocating Rotational Cryptanalysis based Preimage Attack on 4-round Keccak-224 for Quantum Setting, Cryptology ePrint Archive, Paper 2022/977. Available at https://eprint.iacr.org/2022/977.

[132] Qiao K, Song L, Liu M, Guo J (2017) New Collision Attacks on Round-Reduced Keccak. *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Paris, France, April 30 - May 4, 2017, Proceedings, Part III*, eds Coron J, Nielsen JB, *Lecture Notes in Computer Science*, Vol. 10212, pp 216–243. https://doi.org/10.1007/978-3-319-56617-7_8

[133] Song L, Liao G, Guo J (2017) Non-full Sbox Linearization: Applications to Collision Attacks on Round-Reduced Keccak. *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, eds Katz J, Shacham H (Springer), *Lecture Notes in Computer Science*, Vol. 10402, pp 428–451. https://doi.org/10.1007/978-3-319-63715-0_15

[134] Guo J, Liao G, Liu G, Liu M, Qiao K, Song L (2020) Practical Collision Attacks against Round-Reduced SHA-3. *J Cryptol* 33(1):228–270. https://doi.org/10.1007/s00145-019-09313-3

[135] Guo J, Liu G, Song L, Tu Y (2022) Exploring SAT for Cryptanalysis: (Quantum) Collision Attacks against 6-Round SHA-3 (Full Version), Cryptology ePrint Archive, Paper 2022/184. Available at https://eprint.iacr.org/2022/184.

[136] Bernstein DJ (2010) Second preimages for 6 (7? (8??)) rounds of Keccak?, NIST Hash Forum. Available at http://cr.yp.to/hash/keccak-20101127.txt.

[137] Chang D, Kumar A, Morawiecki P, Sanadhya SK (2014) 1st and 2nd Preimage Attacks on 7, 8 and 9 Rounds of Keccak-224,256,384,512, SHA-3 2014 Workshop. Available at https://csrc.nist.gov/events/2014/sha-3-2014-workshop.

[138] Boura C, Canteaut A, Cannière CD (2011) Higher-Order Differential Properties of Keccak and *Luffa*. *Fast Software Encryption - 18th International Workshop, FSE 2011, Lyngby, Den-*

mark, February 13-16, 2011, Revised Selected Papers, ed Joux A (Springer), *Lecture Notes in Computer Science*, Vol. 6733, pp 252–269. https://doi.org/10.1007/978-3-642-21702-9_15

[139] Duan M, Lai X (2011) Improved Zero-sum Distinguisher for Full Round Keccak-f Permutation, Cryptology ePrint Archive, Paper 2011/023. Available at https://eprint.iacr.org/2011/023.

[140] Li M, Cheng L (2017) Distinguishing Property for Full Round KECCAK-f Permutation. *Complex, Intelligent, and Software Intensive Systems - Proceedings of the 11th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS-2017), Torino, Italy, July 10-12, 2017*, eds Barolli L, Terzo O (Springer), *Advances in Intelligent Systems and Computing*, Vol. 611, pp 639–646. https://doi.org/10.1007/978-3-319-61566-0_59

[141] Yan H, Lai X, Wang L, Yu Y, Xing Y (2019) New Zero-sum Distinguishers on Full 24-round Keccak-f using the Division Property. *IET Inf Secur* 13(5):469–478. https://doi.org/10.1049/iet-ifs.2018.5263

[142] Bonnetain X, Jaques S (2020) Quantum Period Finding against Symmetric Primitives in Practice. *CoRR* abs/2011.07022. Available at https://arxiv.org/abs/2011.07022.

[143] Bonnetain X, Hosoyamada A, Naya-Plasencia M, Sasaki Y, Schrottenloher A (2020) Quantum Attacks without Superposition Queries: the Offline Simon's Algorithm. *CoRR* abs/2002.12439. Available at https://arxiv.org/abs/2002.12439.

[144] Alagic G, Bai C, Katz J, Majenz C, Struck P (2022) Post-Quantum Security of the (Tweakable) FX Construction, and Applications, Cryptology ePrint Archive, Paper 2022/1097. Available at https://eprint.iacr.org/2022/1097.

[145] Shi T, Wu W, Hu B, Guan J, Wang S (2021) Breaking LWC candidates: sESTATE and Elephant in Quantum Setting. *Des Codes Cryptogr* https://doi.org/10.1007/s10623-021-00875-7

[146] Banik S, Pandey SK, Peyrin T, Sasaki Y, Sim SM, Todo Y (2017) GIFT: A Small Present, Cryptology ePrint Archive, Paper 2017/622. Available at https://eprint.iacr.org/2017/622.

[147] Banik S, Pandey SK, Peyrin T, Sasaki Y, Sim SM, Todo Y (2017) GIFT: A Small Present - Towards Reaching the Limit of Lightweight Encryption. *Cryptographic Hardware and Embedded Systems - CHES 2017 - 19th International Conference, Taipei, Taiwan, September 25-28, 2017, Proceedings*, eds Fischer W, Homma N (Springer), *Lecture Notes in Computer Science*, Vol. 10529, pp 321–345. https://doi.org/10.1007/978-3-319-66787-4_16

[148] Chakraborti A, Iwata T, Minematsu K, Nandi M (2017) Blockcipher-Based Authenticated Encryption: How Small Can We Go? *CHES* (Springer), *Lecture Notes in Computer Science*, Vol. 10529, pp 277–298. https://doi.org/10.1007/978-3-319-66787-4_14

[149] Chakraborti A, Iwata T, Minematsu K, Nandi M (2020) Blockcipher-Based Authenticated Encryption: How Small Can We Go? *J Cryptol* 33(3):703–741. https://doi.org/10.1007/s00145-019-09325-z

[150] Mennink B (2012) Optimal Collision Security in Double Block Length Hashing with Single Length Key. *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, eds Wang X, Sako K (Springer), *Lecture Notes in Computer Science*, Vol. 7658, pp 526–543. https://doi.org/10.1007/978-3-642-34961-4_32

[151] Cao M, Zhang W (2019) Related-Key Differential Cryptanalysis of the Reduced-Round Block Cipher GIFT. *IEEE Access* 7:175769–175778. https://doi.org/10.1109/ACCESS.2019.2957581

[152] CUI Y, XU H, QI W (2022) MILP-Based Linear Attacks on Round-Reduced GIFT. *Chinese Journal of Electronics* 31(1):89–98. Available at https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/cje.2020.00.113.

[153] Eskandari Z, Kidmose AB, Kölbl S, Tiessen T (2018) Finding Integral Distinguishers with Ease. *Selected Areas in Cryptography - SAC 2018 - 25th International Conference, Calgary, AB, Canada, August 15-17, 2018, Revised Selected Papers*, eds Cid C, Jr MJJ (Springer), *Lecture Notes in Computer Science*, Vol. 11349, pp 115–138. https://doi.org/10.1007/978-3-030-10970-7_6

[154] Ji F, Zhang W, Ding T (2021) Improving Matsui's Search Algorithm For The Best Differential/Linear Trails And Its Applications For DES, DESL And GIFT. *Comput J* 64(4):610–627. https://doi.org/10.1093/comjnl/bxaa090

[155] Ji F, Zhang W, Zhou C, Ding T (2020) Improved (Related-key) Differential Cryptanalysis on GIFT, Cryptology ePrint Archive, Paper 2020/1242. Available at https://eprint.iacr.org/2020/1242.

[156] Khalesi A, Ahmadian Z (2022) Provably Minimum Data Complexity Integral Distinguisher Based on Conventional Division Property, Cryptology ePrint Archive, Paper 2022/752. Available at https://eprint.iacr.org/2022/752.

[157] Li L, Wu W, Zheng Y, Zhang L (2019) The Relationship between the Construction and Solution of the MILP Models and Applications, Cryptology ePrint Archive, Paper 2019/049. Available at https://eprint.iacr.org/2019/049.

[158] Liu Y, Liang H, Li M, Huang L, Hu K, Yang C, Wang M (2021) STP Models of Optimal Differential and Linear Trail for S-box based Ciphers. *Sci China Inf Sci* 64(5). https://doi.org/10.1007/s11432-018-9772-0

[159] Liu Y, Sasaki Y (2019) Related-Key Boomerang Attacks on GIFT with Automated Trail Search Including BCT Effect. *Information Security and Privacy - 24th Australasian Conference, ACISP 2019, Christchurch, New Zealand, July 3-5, 2019, Proceedings*, eds Jang-Jaccard J, Guo F (Springer), *Lecture Notes in Computer Science*, Vol. 11547, pp 555–572. https://doi.org/10.1007/978-3-030-21548-4_30

[160] Sun L, Wang W, Wang M (2021) Linear Cryptanalyses of Three AEADs with GIFT-128 as Underlying Primitives. *IACR Trans Symmetric Cryptol* 2021(2):199–221. https://doi.org/10.46586/tosc.v2021.i2.199-221

[161] Sun L, Wang W, Wang M (2022) Addendum to Linear Cryptanalyses of Three AEADs with GIFT-128 as Underlying Primitives. *IACR Transactions on Symmetric Cryptology* 2022(1):212–219. https://doi.org/10.46586/tosc.v2022.i1.212-219

[162] Zhu B, Dong X, Yu H (2018) MILP-based Differential Attack on Round-reduced GIFT, Cryptology ePrint Archive, Paper 2018/390. Available at https://eprint.iacr.org/2018/390.

[163] Hu K, Peyrin T, Wang M (2022) Finding All Impossible Differentials When Considering the DDT, Cryptology ePrint Archive, Paper 2022/1034. Available at https://eprint.iacr.org/2022/1034.

[164] Baksi A (2020) New Insights on Differential and Linear Bounds Using Mixed Integer Linear Programming. *Innovative Security Solutions for Information Technology and Communications - 13th International Conference, SecITC 2020, Bucharest, Romania, November 19-20, 2020, Revised Selected Papers*, eds Maimut D, Oprina A, Sauveron D (Springer), *Lecture Notes in Computer Science*, Vol. 12596, pp 41–54. https://doi.org/10.1007/978-3-030-69255-1_4

[165] Sun L, Wang W, Wang M (2021) Accelerating the Search of Differential and Linear Characteristics with the SAT Method. *IACR Transactions on Symmetric Cryptology* 2021(1):269–315. https://doi.org/10.46586/tosc.v2021.i1.269-315

[166] Kim S, Hong D, Sung J, Hong S (2022) Accelerating the Best Trail Search on AES-Like Ciphers. *IACR Trans Symmetric Cryptol* 2022(2):201–252. https://doi.org/10.46586/tosc.v2022.i2.201-252

[167] Bijwe S, Chauhan AK, Sanadhya SK (2022) Implementing Grover Oracle for Lightweight Block Ciphers Under Depth Constraints. *Information Security and Privacy - 27th Australasian Conference, ACISP 2022, Wollongong, NSW, Australia, November 28-30, 2022, Proceedings*, eds Nguyen K, Yang G, Guo F, Susilo W (Springer), *Lecture Notes in Computer Science*, Vol. 13494, pp 85–105. https://doi.org/10.1007/978-3-031-22301-3_5

[168] Inoue A, Iwata T, Minematsu K (2022) Analyzing the Provable Security Bounds of GIFT-COFB and Photon-Beetle, Cryptology ePrint Archive, Paper 2022/001. Available at https://eprint.iacr.org/2022/001.

[169] Inoue A, Minematsu K (2021) GIFT-COFB is Tightly Birthday Secure with Encryption Queries, Cryptology ePrint Archive, Paper 2021/737. Available at https://eprint.iacr.org/2021/737.

[170] Khairallah M (2021) Security of COFB against Chosen Ciphertext Attacks, Cryptology ePrint Archive, Paper 2021/648. Available at https://eprint.iacr.org/2021/648.

[171] Khairallah M (2020) Observations on the Tightness of the Security Bounds of GIFT-COFB and HyENA, Cryptology ePrint Archive, Paper 2020/1463. Available at https://eprint.iacr.org/2020/1463.

[172] Rajan R, Roy RK, Sen D, Mishra G (2022) Deep Learning-Based Differential Distinguisher for Lightweight Cipher GIFT-COFB. *Machine Intelligence and Smart Systems* (Springer), pp 397–406. https://doi.org/10.1007/978-981-16-9650-3_31

[173] Inoue A, Guo C, Minematsu K (2022) Nonce-Misuse Resilience of Romulus-N and GIFT-COFB, Cryptology ePrint Archive, Paper 2022/1012. Available at https://eprint.iacr.org/2022/1012.

[174] Bijwe S, Chauhan AK, Sanadhya SK (2020) Quantum Search for Lightweight Block Ciphers: GIFT, SKINNY, SATURNIN, Cryptology ePrint Archive, Paper 2020/1485. Available at https://eprint.iacr.org/2020/1485.

[175] Jang K, Kim H, Eum S, Seo H (2020) Grover on GIFT, Cryptology ePrint Archive, Paper 2020/1405. Available at https://eprint.iacr.org/2020/1405.

[176] Cid C, Robshaw M, Babbage S, Borghoff J, Velichkov V (2012) The eSTREAM Portfolio in 2012, eSTREAM: the ECRYPT Stream Cipher Project. Available at https://www.ecrypt.eu.org/stream/.

[177] ISO/IEC 29167-13:2015 (2015) Information technology — Automatic identification and data capture techniques — Part 13: Crypto suite Grain-128A security services for air interface communications. Available at https://www.iso.org/standard/60682.html.

[178] Berbain C, Gilbert H, Maximov A (2006) Cryptanalysis of Grain. *Fast Software Encryption, 13th International Workshop, FSE 2006, Graz, Austria, March 15-17, 2006, Revised Selected Papers*, ed Robshaw MJB (Springer), *Lecture Notes in Computer Science*, Vol. 4047, pp 15–29. https://doi.org/10.1007/11799313_2

[179] Banik S, Maitra S, Sarkar S, Sönmez Turan M (2013) A Chosen IV Related Key Attack on Grain-128a. *Information Security and Privacy - 18th Australasian Conference, ACISP 2013,*

Brisbane, Australia, July 1-3, 2013. Proceedings, eds Boyd C, Simpson L (Springer), *Lecture Notes in Computer Science*, Vol. 7959, pp 13–26. https://doi.org/10.1007/978-3-642-39059-3_2

[180] Castagnos G, Berzati A, Canovas C, Debraize B, Goubin L, Gouget A, Paillier P, Salgado S (2009) Fault Analysis of Grain-128. *IEEE International Workshop on Hardware-Oriented Security and Trust, HOST 2009, San Francisco, CA, USA, July 27, 2009. Proceedings*, eds Tehranipoor M, Plusquellic J (IEEE Computer Society), pp 7–14. https://doi.org/10.1109/HST.2009.5225030

[181] Zhang H, Wang X (2009) Cryptanalysis of Stream Cipher Grain Family, Cryptology ePrint Archive, Paper 2009/109. Available at https://eprint.iacr.org/2009/109.

[182] Todo Y, Isobe T, Meier W, Aoki K, Zhang B (2018) Fast Correlation Attack Revisited - Cryptanalysis on Full Grain-128a, Grain-128, and Grain-v1. *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, eds Shacham H, Boldyreva A (Springer), *Lecture Notes in Computer Science*, Vol. 10992, pp 129–159. https://doi.org/10.1007/978-3-319-96881-0_5

[183] Wang Q, Hao Y, Todo Y, Li C, Isobe T, Meier W (2018) Improved Division Property Based Cube Attacks Exploiting Algebraic Properties of Superpoly. *Advances in Cryptology – CRYPTO 2018* (Springer), *Lecture Notes in Computer Science*, Vol. 10991, pp 275–305. https://doi.org/10.1007/978-3-319-96884-1_10

[184] Ghafari VA, Hu H (2019) A New Chosen IV Statistical Distinguishing Framework to Attack Symmetric Ciphers, and Its Application to ACORN-v3 and Grain-128a. *J Ambient Intell Humaniz Comput* 10(6):2393–2400. https://doi.org/10.1007/s12652-018-0897-x

[185] Hao Y, Leander G, Meier W, Todo Y, Wang Q (2020) Modeling for Three-Subset Division Property Without Unknown Subset - Improved Cube Attacks Against Trivium and Grain-128AEAD. *Advances in Cryptology - EUROCRYPT 2020 - 39th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, May 10-14, 2020, Proceedings, Part I*, eds Canteaut A, Ishai Y (Springer), *Lecture Notes in Computer Science*, Vol. 12105, pp 466–495. https://doi.org/10.1007/978-3-030-45721-1_17

[186] Hu K, Sun S, Todo Y, Wang M, Wang Q (2021) Massive Superpoly Recovery with Nested Monomial Predictions. *Advances in Cryptology - ASIACRYPT 2021 - 27th International Conference on the Theory and Application of Cryptology and Information Security, Singapore, December 6-10, 2021, Proceedings, Part I*, eds Tibouchi M, Wang H (Springer), *Lecture Notes in Computer Science*, Vol. 13090, pp 392–421. https://doi.org/10.1007/978-3-030-92062-3_14

[187] Bendoukha A, Boudguiga A, Sirdey R (2021) Revisiting Stream-Cipher-Based Homomorphic Transciphering in the TFHE Era. *Foundations and Practice of Security - 14th International Symposium, FPS 2021, Paris, France, December 7-10, 2021, Revised Selected Papers*, eds Aïmeur E, Laurent M, Yaich R, Dupont B, García-Alfaro J (Springer), *Lecture Notes in Computer Science*, Vol. 13291, pp 19–33. https://doi.org/10.1007/978-3-031-08147-7_2

[188] Anand R, Maitra S, Maitra A, Mukherjee CS, Mukhopadhyay S (2020) Resource Estimation of Grovers-kind Quantum Cryptanalysis against FSR based Symmetric Ciphers, Cryptology ePrint Archive, Paper 2020/1438. Available at https://eprint.iacr.org/2020/1438.

[189] Udvarhelyi B, Bronchain O, Standaert F (2021) Security Analysis of Deterministic Re-keying with Masking and Shuffling: Application to ISAP. *Constructive Side-Channel Analysis and Secure Design - 12th International Workshop, COSADE 2021, Lugano, Switzerland, October*

25-27, 2021, Proceedings, eds Bhasin S, Santis FD (Springer), *Lecture Notes in Computer Science*, Vol. 12910, pp 168–183. https://doi.org/10.1007/978-3-030-89915-8_8

[190] Dobraunig C, Mennink B (2019) Leakage Resilience of the Duplex Construction. *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part III*, eds Galbraith SD, Moriai S (Springer), *Lecture Notes in Computer Science*, Vol. 11923, pp 225–255. https://doi.org/10.1007/978-3-030-34618-8_8

[191] Dobraunig C, Mennink B (2019) Leakage Resilience of the ISAP Mode: a Vulgarized Summary, NIST Lightweight Cryptography Workshop 2019. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2019/documents/papers/leakage-resilience-isap-mode-lwc2019.pdf.

[192] Dobraunig C, Mennink B (2019) Security of the Suffix Keyed Sponge. *IACR Trans Symmetric Cryptol* 2019(4):223–248. https://doi.org/10.13154/tosc.v2019.i4.223-248

[193] Dobraunig C, Mennink B (2020) Tightness of the Suffix Keyed Sponge Bound. *IACR Trans Symmetric Cryptol* 2020(4):195–212. https://doi.org/10.46586/tosc.v2020.i4.195-212

[194] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2020) Isap v2.0. *IACR Trans Symmetric Cryptol* 2020(S1):390–416. https://doi.org/10.13154/tosc.v2020.iS1.390-416

[195] Bhattacharjee A, Chakraborti A, Datta N, Mancillas-López C, Nandi M (2022) ISAP+: ISAP with Fast Authentication. *Progress in Cryptology - INDOCRYPT 2022 - 23rd International Conference on Cryptology in India, Kolkata, India, December 11-14, 2022, Proceedings*, eds Isobe T, Sarkar S (Springer), *Lecture Notes in Computer Science*, Vol. 13774, pp 195–219. https://doi.org/10.1007/978-3-031-22912-1_9

[196] Guo J, Peyrin T, Poschmann A (2011) The PHOTON Family of Lightweight Hash Functions. *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, ed Rogaway P (Springer), *Lecture Notes in Computer Science*, Vol. 6841, pp 222–239. https://doi.org/10.1007/978-3-642-22792-9_13

[197] Dobraunig C, Mennink B (2020) [lwc-forum] Official Comment: PHOTON-Beetle, Email to lwc-forum. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/official-comments/photon-beetle-round2-official-comment.pdf.

[198] Chakraborti A, Datta N, Nandi M, Yasuda K (2018) Beetle Family of Lightweight and Secure Authenticated Encryption Ciphers. *IACR Trans Cryptogr Hardw Embed Syst* 2018(2):218–241. https://doi.org/10.13154/tches.v2018.i2.218-241

[199] Chakraborty B, Jha A, Nandi M (2019) Security Proof of Beetle and SpoC, NIST Lightweight Cryptography Workshop 2019. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2019/documents/papers/security-proof-of-beetle-spoc-proof-lwc2019.pdf.

[200] Chakraborty B, Jha A, Nandi M (2019) On the Security of Sponge-type Authenticated Encryption Modes, Cryptology ePrint Archive, Paper 2019/1475. Available at https://eprint.iacr.org/2019/1475.

[201] Chakraborty B, Jha A, Nandi M (2020) On the Security of Sponge-type Authenticated Encryption Modes. *IACR Trans Symmetric Cryptol* 2020(2):93–119. https://doi.org/10.13154/tosc.v2020.i2.93-119

[202] Wang Q, Grassi L, Rechberger C (2018) Zero-Sum Partitions of PHOTON Permutations. *Topics in Cryptology - CT-RSA 2018 - The Cryptographers' Track at the RSA Conference*

2018, San Francisco, CA, USA, April 16-20, 2018, Proceedings, ed Smart NP (Springer), *Lecture Notes in Computer Science*, Vol. 10808, pp 279–299. https://doi.org/10.1007/978-3-319-76953-0_15

[203] Jean J, Naya-Plasencia M, Peyrin T (2012) Improved Rebound Attack on the Finalist Grøstl. *Fast Software Encryption - 19th International Workshop, FSE 2012, Washington, DC, USA, March 19-21, 2012. Revised Selected Papers*, ed Canteaut A (Springer), *Lecture Notes in Computer Science*, Vol. 7549, pp 110–126. https://doi.org/10.1007/978-3-642-34047-5_7

[204] Jean J, Naya-Plasencia M, Peyrin T (2013) Multiple Limited-Birthday Distinguishers and Applications. *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, eds Lange T, Lauter KE, Lisonek P (Springer), *Lecture Notes in Computer Science*, Vol. 8282, pp 533–550. https://doi.org/10.1007/978-3-662-43414-7_27

[205] Mege A (2021) [lwc-forum] Official Comment: PHOTON-Beetle, Email to lwc-forum. Available at https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/official-comments/photon-beetle-round2-official-comment.pdf.

[206] Beierle C, Jean J, Kölbl S, Leander G, Moradi A, Peyrin T, Sasaki Y, Sasdrich P, Sim SM (2016) The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. *Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part II*, eds Robshaw M, Katz J (Springer), *Lecture Notes in Computer Science*, Vol. 9815, pp 123–153. https://doi.org/10.1007/978-3-662-53008-5_5

[207] Tolba M, Abdelkhalek A, Youssef AM (2017) Impossible Differential Cryptanalysis of Reduced-Round SKINNY. *Progress in Cryptology - AFRICACRYPT 2017 - 9th International Conference on Cryptology in Africa, Dakar, Senegal, May 24-26, 2017, Proceedings*, eds Joye M, Nitaj A, *Lecture Notes in Computer Science*, Vol. 10239, pp 117–134. https://doi.org/10.1007/978-3-319-57339-7_7

[208] Hadipour H, Bagheri N, Song L (2021) Improved Rectangle Attacks on SKINNY and CRAFT. *IACR Trans Symmetric Cryptol* 2021(2):140–198. https://doi.org/10.46586/tosc.v2021.i2.140-198

[209] Hadipour H, Sadeghi S, Eichlseder M (2022) Finding the Impossible: Automated Search for Full Impossible Differential, Zero-Correlation, and Integral Attacks, Cryptology ePrint Archive, Paper 2022/1147. Available at https://eprint.iacr.org/2022/1147.

[210] Shi D, Sun S, Derbez P, Todo Y, Sun B, Hu L (2018) Programming the Demirci-Selçuk Meet-in-the-Middle Attack with Constraints. *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part II*, eds Peyrin T, Galbraith SD (Springer), *Lecture Notes in Computer Science*, Vol. 11273, pp 3–34. https://doi.org/10.1007/978-3-030-03329-3_1

[211] Chen Q, Shi D, Sun S, Hu L (2019) Automatic Demirci-Selçuk Meet-in-the-Middle Attack on SKINNY with Key-Bridging. *Information and Communications Security - 21st International Conference, ICICS 2019, Beijing, China, December 15-17, 2019, Revised Selected Papers*, eds Zhou J, Luo X, Shen Q, Xu Z (Springer), *Lecture Notes in Computer Science*, Vol. 11999, pp 233–247. https://doi.org/10.1007/978-3-030-41579-2_14

[212] Zhao B, Dong X, Meier W, Jia K, Wang G (2019) Generalized Related-Key Rectangle Attacks on Block Ciphers with Linear Key Schedule: Applications to SKINNY and GIFT,

Cryptology ePrint Archive, Paper 2019/714. Available at https://eprint.iacr.org/2019/714.

[213] Zhao B, Dong X, Meier W, Jia K, Wang G (2020) Generalized Related-key Rectangle Attacks on Block Ciphers with Linear Key Schedule: Applications to SKINNY and GIFT. *Des Codes Cryptogr* 88(6):1103–1126. https://doi.org/10.1007/s10623-020-00730-1

[214] Qin L, Dong X, Wang X, Jia K, Liu Y (2021) Automated Search Oriented to Key Recovery on Ciphers with Linear Key Schedule Applications to Boomerangs in SKINNY and ForkSkinny. *IACR Trans Symmetric Cryptol* 2021(2):249–291. https://doi.org/10.46586/tosc.v2021.i2.249-291

[215] Habu M, Minematsu K, Iwata T (2022) Matching Attacks on Romulus-M, Cryptology ePrint Archive, Paper 2022/369. Available at https://eprint.iacr.org/2022/369.

[216] Iwata T, Khairallah M, Minematsu K, Peyrin T (2019) Updates on Romulus, Remus and TGIF, NIST Lightweight Cryptography Workshop 2019. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2019/documents/papers/updates-on-romulus-remus-tgif-lwc2019.pdf.

[217] Iwata T, Khairallah M, Minematsu K, Peyrin T (2019) Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms, Cryptology ePrint Archive, Paper 2019/992. Available at https://eprint.iacr.org/2019/992.

[218] Iwata T, Khairallah M, Minematsu K, Peyrin T (2020) Duel of the Titans: The Romulus and Remus Families of Lightweight AEAD Algorithms. *IACR Trans Symmetric Cryptol* 2020(1):43–120. https://doi.org/10.13154/tosc.v2020.i1.43-120

[219] Iwata T, Khairallah M, Minematsu K, Peyrin T (2020) New Results on Romulus, NIST Lightweight Cryptography Workshop 2020. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/new-results-romulus-lwc2020.pdf.

[220] Guo C, Khairallah M, Peyrin T (2020) AET-LR: Rate-1 Leakage-Resilient AEAD based on the Romulus Family, NIST Lightweight Cryptography Workshop 2020. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/AET-LR-lwc2020.pdf.

[221] Guo C, Iwata T, Minematsu K (2022) New Indifferentiability Security Proof of MDPH Hash Function. *IET Inf Secur* 16(4):262–281. https://doi.org/10.1049/ise2.12058

[222] Dong X, Hua J, Sun S, Li Z, Wang X, Hu L (2021) Meet-in-the-Middle Attacks Revisited: Key-Recovery, Collision, and Preimage Attacks. *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, eds Malkin T, Peikert C (Springer), *Lecture Notes in Computer Science*, Vol. 12827, pp 278–308. https://doi.org/10.1007/978-3-030-84252-9_10

[223] Nageler M, Pallua F, Eichlseder M (2022) Finding Collisions for Round-Reduced Romulus-H, Cryptology ePrint Archive, Paper 2022/1630. Available at https://eprint.iacr.org/2022/1630.

[224] Beierle C, Biryukov A, dos Santos LC, Großschädl J, Perrin L, Udovenko A, Velichkov V, Wang Q (2020) Alzette: A 64-Bit ARX-box - (Feat. CRAX and TRAX. *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, eds Micciancio D, Ristenpart T (Springer), *Lecture Notes in Computer Science*, Vol. 12172, pp 419–448. https://doi.org/10.1007/978-3-030-56877-1_15

[225] Beierle C, Biryukov A, dos Santos LC, Großschädl J, Perrin L, Udovenko A, Velichkov

V, Wang Q (2020) Lightweight AEAD and Hashing using the Sparkle Permutation Family. *IACR Trans Symmetric Cryptol* 2020(S1):208–261. https://doi.org/10.13154/tosc.v2020.iS1.208-261

[226] Schrottenloher A, Stevens M (2022) Simplified MITM Modeling for Permutations: New (Quantum) Attacks. *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part III*, eds Dodis Y, Shrimpton T (Springer), *Lecture Notes in Computer Science*, Vol. 13509, pp 717–747. https://doi.org/10.1007/978-3-031-15982-4_24

[227] Huang M, Xu Z, Wang L (2022) On the Probability and Automatic Search of Rotational-XOR Cryptanalysis on ARX Ciphers. *Comput J* 65(12):3062–3080. https://doi.org/10.1093/comjnl/bxab126

[228] Liu Y, Sun S, Li C (2021) Rotational Cryptanalysis From a Differential-linear Perspective: Practical Distinguishers for Round-reduced FRIET, Xoodoo, and Alzette, Cryptology ePrint Archive, Paper 2021/189. Available at https://eprint.iacr.org/2021/189.

[229] Liu Y, Sun S, Li C (2021) Rotational Cryptanalysis from a Differential-Linear Perspective - Practical Distinguishers for Round-Reduced FRIET, Xoodoo, and Alzette. *Advances in Cryptology - EUROCRYPT 2021 - 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17-21, 2021, Proceedings, Part I*, eds Canteaut A, Standaert F (Springer), *Lecture Notes in Computer Science*, Vol. 12696, pp 741–770. https://doi.org/10.1007/978-3-030-77870-5_26

[230] Niu Z, Sun S, Liu Y, Li C (2022) Rotational Differential-Linear Distinguishers of ARX Ciphers with Arbitrary Output Linear Masks, Cryptology ePrint Archive, Paper 2022/765. Available at https://eprint.iacr.org/2022/765.

[231] Xu Z, Li Y, Jiao L, Wang M, Meier W (2022) Do NOT Misuse the Markov Cipher Assumption - Automatic Search for Differential and Impossible Differential Characteristics in ARX Ciphers. *IACR Cryptol ePrint Arch* :135Available at https://eprint.iacr.org/2022/135.

[232] Speel T (2022) Cryptanalysis of SPARKLE's ARX-box Alzette. *Bachelor Thesis, Radboud University* Available at http://www.cs.ru.nl/bachelors-theses/2022/Ties_Speel___1020150___Cryptanalysis_of_SPARKLE_ARX-Box_Alzette.pdf.

[233] Jagielski A, Kanciak K (2022) Grover on Sparkle Quantum Resource Estimation for a NIST LWC Call Finalist. *Quantum Information and Computation* 22(13&14):1132–1143. https://doi.org/10.26421/QIC22.13-14-3

[234] Wu H, Huang T (2014) JAMBU Lightweight Authenticated Encryption Mode and AES-JAMBU, CAESAR competition proposal. Available at https://competitions.cr.yp.to/round1/aesjambuv1.pdf.

[235] Saha D, Sasaki Y, Shi D, Sibleyras F, Sun S, Zhang Y (2020) On the Security Margin of TinyJAMBU with Refined Differential and Linear Cryptanalysis, NIST Lightweight Cryptography Workshop 2020. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/security-margin-tinyJAMBU-lwc2020.pdf.

[236] Saha D, Sasaki Y, Shi D, Sibleyras F, Sun S, Zhang Y (2020) On the Security Margin of TinyJAMBU with Refined Differential and Linear Cryptanalysis, Cryptology ePrint Archive, Paper 2020/1045. Available at https://eprint.iacr.org/2020/1045.

[237] Saha D, Sasaki Y, Shi D, Sibleyras F, Sun S, Zhang Y (2020) On the Security Margin of TinyJAMBU with Refined Differential and Linear Cryptanalysis. *IACR Trans Symmetric Cryptol* 2020(3):152–174. https://doi.org/10.13154/tosc.v2020.i3.152-174

[238] Teng WL, Salam I, Yau WC, Pieprzyk J, Phan RCW (2021) Cube Attacks on Round-Reduced TinyJAMBU, Cryptology ePrint Archive, Paper 2021/1164. Available at https://eprint.iacr.org/2021/1164.

[239] Dunkelman O, Ghosh S, Lambooij E (2022) Full Round Zero-sum Distinguishers on TinyJAMBU-128 and TinyJAMBU-192 Keyed-permutation in the Known-key Setting, Cryptology ePrint Archive, Paper 2022/1567. Available at https://eprint.iacr.org/2022/1567.

[240] Datta N, Dutta A, Ghosh S (2022) INT-RUP Security of SAEB and TinyJAMBU, Cryptology ePrint Archive, Paper 2022/1414. Available at https://eprint.iacr.org/2022/1414.

[241] Jana A, Rahman M, Saha D (2022) DEEPAND: In-Depth Modeling of Correlated AND Gates for NLFSR-based Lightweight Block Ciphers, Cryptology ePrint Archive, Paper 2022/1123. Available at https://eprint.iacr.org/2022/1123.

[242] Li M, Mouha N, Sun L, Wang M (2022) Revisiting the Extension of Matsui's Algorithm 1 to Linear Hulls: Application to TinyJAMBU. *IACR Transactions on Symmetric Cryptology* 2022(2):161–200. https://doi.org/10.46586/tosc.v2022.i2.161-200

[243] Sibleyras F, Sasaki Y, Todo Y, Hosoyamada A, Yasuda K (2022) Birthday-Bound Slide Attacks on TinyJAMBU's Keyed-Permutations for All Key Sizes, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/birthday-bound-slide-attacks-on-tinyjambus-keyed-permutations-for-all-key-sizes.pdf.

[244] Song L, Guo J (2018) Cube-Attack-Like Cryptanalysis of Round-Reduced Keccak Using MILP. *IACR Trans Symmetric Cryptol* 2018(3):182–214. https://doi.org/10.13154/tosc.v2018.i3.182-214

[245] Zhang Z, Zhang W, Shi H (2021) Genetic Algorithm Assisted State-Recovery Attack on Round-Reduced Xoodyak. *Computer Security - ESORICS 2021 - 26th European Symposium on Research in Computer Security, Darmstadt, Germany, October 4-8, 2021, Proceedings, Part II*, eds Bertino E, Shulman H, Waidner M (Springer), *Lecture Notes in Computer Science*, Vol. 12973, pp 257–274. https://doi.org/10.1007/978-3-030-88428-4_13

[246] Dunkelman O, Weizman A (2022) Differential-Linear Cryptanalysis on Xoodyak, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/differential-linear-cryptanalysis-on-xoodyak.pdf.

[247] Blach C (2019) Xoodoo Trail Analysis. *Bachelor Thesis, Radboud University* Available at https://www.cs.ru.nl/bachelors-theses/2019/Constantin_Blach___4329872___Xoodoo_Trail_Analysis.pdf.

[248] Daemen J, Mella S, Assche GV (2022) Tighter Trail Bounds for Xoodoo, Cryptology ePrint Archive, Paper 2022/1088. Available at https://eprint.iacr.org/2022/1088.

[249] Li H, Liu G, Zhang H, Hu K, Guo J, Qiu W (2022) AlgSAT – a SAT Method for Search and Verification of Differential Characteristics from Algebraic Perspective, Cryptology ePrint Archive, Paper 2022/1641. Available at https://eprint.iacr.org/2022/1641.

[250] Bellini E, Makarim RH (2022) Functional Cryptanalysis: Application to Reduced-round Xoodoo, Cryptology ePrint Archive, Paper 2022/134. Available at https://eprint.iacr.org/2022/134.

[251] Gilbert H, Boissier RH, Khati L, Rotella Y (2023) Generic Attack on Duplex-Based AEAD Modes using Random Function Statistics. *IACR Cryptol ePrint Arch* :262Available at https://eprint.iacr.org/2023/262.

[252] Liu G, Lu J, Li H, Tang P, Qiu W (2021) Preimage Attacks Against Lightweight Scheme Xoodyak Based on Deep Learning. *Advances in Information and Communication*, ed Arai K (Springer International Publishing, Cham), pp 637–648. https://doi.org/10.1007/978-3-030-73103-8_45

[253] NIST (2020) Benchmarking of Lightweight Cryptographic Algorithms on Microcontrollers, GitHub repository. Available at https://doi.org/10.18434/mds2-2958.

[254] Renner S, Pozzobon E, Mottok J LWC Benchmark, GitHub repository. Available at https://lab.las3.de/gitlab/lwc/compare.

[255] Weatherley R LWC Finalists, GitHub repository. Available at https://github.com/rweather/lwc-finalists.

[256] PlatformIO A Professional Collaborative Platform for Embedded Development. Available at https://platformio.org/.

[257] Renner S, Pozzobon E, Mottok J NIST LWC Software Performance Benchmarks on Microcontrollers, Project Webpage. Available at https://lwc.las3.de/.

[258] Renner S (2022) 3rd Round Ciphers Evaluation on Microcontrollers, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/Presentations/2022/3rd-round-ciphers-evaluation-on-microcontrollers.

[259] Renner S, Pozzobon E, Mottok J (2022) The Final Round: Benchmarking NIST LWC Ciphers on Microcontrollers. *Attacks and Defenses for the Internet-of-Things - 5th International Workshop, ADIoT 2022, Copenhagen, Denmark, September 30, 2022, Revised Selected Papers*, eds Li W, Furnell S, Meng W (Springer), *Lecture Notes in Computer Science*, Vol. 13745, pp 1–20. https://doi.org/10.1007/978-3-031-21311-3_1

[260] Weatherley R Lightweight Cryptography Primitives, Project Webpage. Available at https://rweather.github.io/lwc-finalists/index.html.

[261] Bernstein DJ (2008) ChaCha, a variant of Salsa20, State of the Art of Stream Ciphers (SASC) 2008. Available at https://www.ecrypt.eu.org/stvl/sasc2008/.

[262] Bernstein DJ (2005) The Poly1305-AES Message-Authentication Code. *FSE*, eds Gilbert H, Handschuh H (Springer), *Lecture Notes in Computer Science*, Vol. 3557, pp 32–49. Available at https://doi.org/10.1007/11502760_3.

[263] Nir Y, Langley A (2018) ChaCha20 and Poly1305 for IETF Protocols, Internet Research Task Force (IRTF), Request for Comments (RFC) 8439. Available at https://doi.org/10.17487/RFC8439.

[264] Aumasson J, Neves S, Wilcox-O'Hearn Z, Winnerlein C (2013) BLAKE2: Simpler, Smaller, Fast as MD5. *Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings*, eds Jr MJJ, Locasto ME, Mohassel P, Safavi-Naini R (Springer), *Lecture Notes in Computer Science*, Vol. 7954, pp 119–135. https://doi.org/10.1007/978-3-642-38980-1_8

[265] Bernstein D, Lange T eBACS: ECRYPT Benchmarking of Cryptographic Systems. Available at https://bench.cr.yp.to/.

[266] Watanabe Y, Yamamoto H, Yoshida H (2022) Performance Evaluation of NIST LWC Finalists on AVR ATmega and ARM Cortex-M3 Microcontrollers, Cryptology ePrint Archive, Paper 2022/1071. Available at https://eprint.iacr.org/2022/1071.

[267] Ruigrok L (2022) Benchmarking Lightweight Cryptography for TLS. *Bachelor's thesis, Universiteit Leiden* Available at https://theses.liacs.nl/pdf/2022-2023-RuigrokL.pdf.

[268] Cheng H, Großschädl J, Marshall B, Page D, Pham TH (2023) RISC-V Instruction Set Ex-

tensions for Lightweight Symmetric Cryptography. *IACR Trans Cryptogr Hardw Embed Syst* 2023(1):193–237. https://doi.org/10.46586/tches.v2023.i1.193-237

[269] Hira R, Kitahara T, Miyahara D, Hara-Azumi Y, Li Y, Sakiyama K (2023) Software Evaluation for Second Round Candidates in NIST Lightweight Cryptography. *J Inf Process* 31:205–219. https://doi.org/10.2197/ipsjjip.31.205

[270] Altinay Ö, Örs B (2021) Instruction Extension of RV32I and GCC Back End for Ascon Lightweight Cryptography Algorithm. *2021 IEEE International Conference on Omni-Layer Intelligent Systems, COINS 2021, Barcelona, Spain, August 23-25, 2021* (IEEE), pp 1–6. https://doi.org/10.1109/COINS51742.2021.9524190

[271] Steinegger S, Primas R (2020) A Fast and Compact RISC-V Accelerator for Ascon and Friends. *Smart Card Research and Advanced Applications - 19th International Conference, CARDIS 2020, Virtual Event, November 18-19, 2020, Revised Selected Papers*, eds Liardet P, Mentens N (Springer), *Lecture Notes in Computer Science*, Vol. 12609, pp 53–67. https://doi.org/10.1007/978-3-030-68487-7_4

[272] Dobraunig C, Eichlseder M, Mendel F, Primas R, Schläffer M (2022) New Ascon Implementations, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/new-ascon-implementations.pdf.

[273] Adomnicai A, Najm Z, Peyrin T (2020) Fixslicing: A New GIFT Representation Fast Constant-Time Implementations of GIFT and GIFT-COFB on ARM Cortex-M. *IACR Trans Cryptogr Hardw Embed Syst* 2020(3):402–427. https://doi.org/10.13154/tches.v2020.i3.402-427

[274] Charlès A, Gravouil C (2022) Review of the White-Box Encodability of NIST Lightweight Finalists, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/review-of-the-white-box-encodability-of-nist-lighweight-finalists.pdf.

[275] Charlès A, Gravouil C (2022) Review of the White-Box Encodability of NIST Lightweight Finalists, Cryptology ePrint Archive, Paper 2022/804. Available at https://eprint.iacr.org/2022/804.

[276] Maximov A, Hell M (2020) Software Evaluation of Grain-128AEAD for Embedded Platforms, Cryptology ePrint Archive, Paper 2020/659. Available at https://eprint.iacr.org/2020/659.

[277] Watanabe Y, Yamamoto H, Yoshida H (2020) Towards Minimizing RAM Requirement for Implementation of Grain-128a on ARM Cortex-M3. *IEICE Trans Fundam Electron Commun Comput Sci* 103-A(1):2–10. https://doi.org/10.1587/transfun.2019CIP0025

[278] Adomnicai A, Minematsu K, Shigeri M (2022) Fast Skinny-128 SIMD Implementations for Sequential Modes of Operation. *Information Security and Privacy - 27th Australasian Conference, ACISP 2022, Wollongong, NSW, Australia, November 28-30, 2022, Proceedings*, eds Nguyen K, Yang G, Guo F, Susilo W (Springer), *Lecture Notes in Computer Science*, Vol. 13494, pp 125–144. https://doi.org/10.1007/978-3-031-22301-3_7

[279] Duka AV (2023) Software Implementation and Benchmarking of TinyJAMBU on Programmable Logic Controllers. *The 16th International Conference Interdisciplinarity in Engineering*, eds Moldovan L, Gligor A (Springer International Publishing, Cham), pp 889–899. https://doi.org/10.1007/978-3-031-22375-4_73

[280] van der Burgt T (2019) *Implementing the NIST Lightweight Cryptography Candidates*

Xoodyak & Subterranean 2.0 Bachelor thesis Radboud University. Available at https://www.cs.ru.nl/bachelors-theses/2019/Thomas_van_der_Burgt___Implementing_the_NIST_lightweight_cryptography_candidates_Xooday_and_Subterranean_2.0.pdf.

[281] Mohajerani K, Haeussler R, Nagpal R, Farahmand F, Abdulgadir A, Kaps JP, Gaj K (2021) FPGA Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process: Methodology, Metrics, Tools, and Results, Cryptology ePrint Archive, Report 2020/1207 (updated February 24, 2021). Available at https://eprint.iacr.org/2020/1207/20210224:202629.

[282] Kaps JP, Gaj K (2022) SCA Evaluation & Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process, Email to lwc-forum. Available at https://cryptography.gmu.edu/athena/LWC/SCA_Evaluation_and_Benchmarking_of_LWC_Finalists.pdf.

[283] Mohajerani K, Beckwith L, Abdulgadir A, Ferrufino E, Kaps JP, Gaj K (2023) SCA Evaluation and Benchmarking of Finalists in the NIST Lightweight Cryptography Standardization Process, Cryptology ePrint Archive, Paper 2023/484. Available at https://eprint.iacr.org/2023/484.

[284] ATHENa, Project Webpage. Available at https://cryptography.gmu.edu/athena/index.php?id=LWC.

[285] Farahmand F, Ferozpuri A, Diehl W, Gaj K (2017) Minerva: Automated Hardware Optimization Tool. *International Conference on ReConFigurable Computing and FPGAs, ReConFig 2017, Cancun, Mexico, December 4-6, 2017* (IEEE), pp 1–8. https://doi.org/10.1109/RECONFIG.2017.8279804

[286] Mohajerani K, Nagpal R (2020) Xeda: Cross-EDA Abstraction and Automation. Available at https://github.com/XedaHQ/xeda.

[287] Khairallah M, Peyrin T, Chattopadhyay A (2020) Preliminary Hardware Benchmarking of a Group of Round 2 NIST Lightweight AEAD Candidates, Cryptology ePrint Archive, Paper 2020/1459. Available at https://eprint.iacr.org/2020/1459.

[288] Khairallah M Lightweight Cryptography ASIC Benchmarking, GitHub repository. Available at https://github.com/mustafam001/lwc-aead-rtl.

[289] Aagaard MD, Zidaric N (2021) ASIC Benchmarking of Round 2 Candidates in the NIST Lightweight Cryptography Standardization Process, Cryptology ePrint Archive, Paper 2021/049. Available at https://eprint.iacr.org/2021/049.

[290] Khan S, Lee WK, Karmakar A, Mera JMB, Majeed A, Hwang SO (2022) Area-time Efficient Implementation of NIST Lightweight Hash Functions Targeting IoT Applications. *IEEE Internet of Things Journal* :1–1https://doi.org/10.1109/JIOT.2022.3229516

[291] Elsadek I, Aftabjahani S, Gardner D, MacLean E, Wallrabenstein JR, Tawfik EY (2022) Hardware and Energy Efficiency Evaluation of NIST Lightweight Cryptography Standardization Finalists. *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp 133–137. https://doi.org/10.1109/ISCAS48785.2022.9937643

[292] Elsadek I, Aftabjahani S, Gardner D, MacLean E, Wallrabenstein JR, Tawfik EY (2022) Energy Efficiency Enhancement of Parallelized Implementation of NIST Lightweight Cryptography Standardization Finalists. *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp 138–141. https://doi.org/10.1109/ISCAS48785.2022.9937755

[293] Gross H, Wenger E, Dobraunig C, Ehrenhöfer C (2017) Ascon hardware implementations and side-channel evaluation. *Microprocessors and Microsystems* 52:470–479. https://doi.org/10.1016/j.micpro.2016.10.006

[294] Kaur J, Mozaffari Kermani M, Azarderakhsh R (2022) Hardware Constructions for Error Detection in Lightweight Authenticated Cipher ASCON Benchmarked on FPGA. *IEEE Transactions on Circuits and Systems II: Express Briefs* 69(4):2276–2280. https://doi.org/10.1109/TCSII.2021.3136463

[295] Caforio A, Collins D, Banik S, Regazzoni F (2022) A Small GIFT-COFB: Lightweight Bit-Serial Architectures, Cryptology ePrint Archive, Paper 2022/955. Available at https://eprint.iacr.org/2022/955.

[296] Zhong Y, Guin U (2022) Chosen-Plaintext Attack on Energy-Efficient Hardware Implementation of GIFT-COFB. *2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST)*, pp 73–76. https://doi.org/10.1109/HOST54066.2022.9840293

[297] Sönnerup J, Hell M, Sönnerup M, Khattar R (2019) Efficient Hardware Implementations of Grain-128AEAD. *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings*, eds Hao F, Ruj S, Gupta SS (Springer), *Lecture Notes in Computer Science*, Vol. 11898, pp 495–513. https://doi.org/10.1007/978-3-030-35423-7_25

[298] Wakeland MC (2022) *ASIC Benchmarking for Proposed Lightweight Cryptography Standard Xoodyak* Ph.D. thesis Monterey, CA; Naval Postgraduate School. Available at https://apps.dtic.mil/sti/trecms/pdf/AD1173488.pdf.

[299] Cryptographic Engineering Research Group (2022) Call for Protected Hardware Implementations of Finalists in the NIST Lightweight Cryptography Standardization Process, Email to lwc-forum. Available at https://cryptography.gmu.edu/athena/LWC/Call_for_Protected_Hardware_Implementations.pdf.

[300] Cryptographic Engineering Research Group (2022) Call for Protected Software Implementations of Finalists in the NIST Lightweight Cryptography Standardization Process, Email to lwc-forum. Available at https://cryptography.gmu.edu/athena/LWC/Call_for_Protected_Software_Implementations.pdf.

[301] Cryptographic Engineering Research Group (2022) Call for Side-Channel Security Evaluation Labs, Email to lwc-forum. Available at https://cryptography.gmu.edu/athena/LWC/Call_for_Security_Evaluation_Labs.pdf.

[302] Primas R, Mangard S ISAP Code Package, GitHub repository. Available at https://github.com/isap-lwc/isap-code-package.

[303] Schlaffer M Submission to Call for Protected Software Implementations, GitHub repository. Available at https://github.com/ascon/simpleserial-ascon/releases/tag/v1.2.6.

[304] Adomnicai A GIFT-COFB Software Implementation Protected against 1st-order Side-Channel Attacks, GitHub repository. Available at https://github.com/aadomn/giftcofb_adomnicai.

[305] Adomnicai A Romulus Software Implementations Protected against 1st-order Side-Channel Attacks, GitHub repository. Available at https://github.com/aadomn/romulus_adomnicai.

[306] Peng S, Yin S, Zhao C, Liu L, Yang B, Zhu W NIST LWC SCA Protected Hardware and Software Implementations, GitHub repository. Available at https://github.com/ybhphoenix/THU_HWSec_LWC.

[307] Primas R, Mangard S NIST LWC Hardware Reference Implementation of ISAP v2.0, GitHub repository. Available at https://github.com/isap-lwc/isap-hardware-package.

[308] Nicolai Mueller and Amir Moradi LWC-Masking, GitHub repository. Available at https://github.com/Chair-for-Security-Engineering/LWC-Masking.

[309] Lin S, Abdulgadir A, Kaps JP, Gaj K Masked Implementation of TinyJAMBU, GitHub repository. Available at https://github.com/GMUCERG/TinyJAMBU-SCA.
[310] Nagpal R, Primas R, Mangard S (2022) Application for SCA Evaluation Lab, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_SW_HW_IAIK_Graz_Austria.pdf.
[311] Sauvage L (2022) Trusted Analysis Platform, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_SW_HW_Telecom_Paris_France.pdf.
[312] Laboratory for Safe & Secure Systems (2022) NIST LWC Side-Channel Security Evaluation Lab, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_SW_OTH_Regensburg_Germany.pdf.
[313] Gu D (2022) NIST LWC Side-Channel Security Evaluation Lab, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_SW_HW_SJTU_China.pdf.
[314] Tsinghua University (2022) Hardware Security and Cryptographic Processor Lab, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_SW_HW_Tsinghua_China.pdf.
[315] Batina L (2022) Side-Channel Security Evaluation Lab Description, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_SW_HW_CESCA_Radboud_the_Netherlands.pdf.
[316] Guilley S, Karray K (2022) Secure-IC's Answer to Side-Channel Security Evaluation Labs Call, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_HW_Secure-IC_France.pdf.
[317] Cryptographic Engineering Research Group (2022) GMU Side-Channel Security Evaluation Lab Proposal (Hardware Implementations), Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/LWC_lab_specs/Lab_HW_CERG_GMU_USA.pdf.
[318] Zhao C, Zhao H, Yang B, Zhu W, Liu L (2022) Leakage Assessment Report For Ascon_DOM_1st_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/Tsinghua/Tsinghua_Report_HW_Ascon_TUGraz.pdf.
[319] Steinbauer T, Nagpal R, Primas R, Manguard S (2022) TVLA On Selected NIST LWC Finalists, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/TUGraz/TUGraz_Report_HW_5_candidates_RUB.pdf.
[320] Zheng H, Cao P (2022) On the Side-Channel Leakage Assessment of Ascon with Boolean Masking, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/SJTU/SJTU_Report_HW_4_candidates_RUB.pdf.
[321] George Mason University (2022) Leakage Assessment on Ascon_first_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/GMU/GMU_Report_HW_Ascon_first_order_RUB.pdf.
[322] Abdulgadir A, Kaps JP, Gaj K (2022) Leakage Assessment on Elephant_first_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/GMU/GMU_Report_HW_Elephant_first_order_RUB.pdf.
[323] Lu X, Qu S, Wang T, Cao P (2022) On the Side Channel Leakage Assessment of First-Order Masked GIFT-COFB, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/SJTU/SJTU_Report_HW_4_candidates_RUB.pdf.
[324] George Mason University (2022) Leakage Assessment on Photon-Beetle_first_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/GMU/

GMU_Report_HW_PHOTON-Beetle_first_order_RUB.pdf.

[325] Zhang X, Wang T, Cao P (2022) On the Side Channel Leakage Assessment of First-Order Masked Romulus, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/SJTU/SJTU_Report_HW_4_candidates_RUB.pdf.

[326] Zhao C, Zhao H, Yang B, Zhu W, Liu L (2022) Leakage Assessment Report for TinyJAMBU_DOM_1st_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/Tsinghua/Tsinghua_Report_HW_TinyJAMBU_GMU.pdf.

[327] Abdulgadir A, Kaps JP, Gaj K (2022) Leakage Assessment on TinyJAMBU_first_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/GMU/GMU_Report_HW_TinyJAMBU_first_order_RUB.pdf.

[328] Forlot F, Riadi N, Karray K, Guilley S (2022) Side Channel Analysis of Xoodyak LWC NIST candidate, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/Secure-IC/Secure-IC_Report_HW_Xoodyak_GMU.pdf.

[329] Zhao C, Zhao H, Yang B, Zhu W, Liu L (2022) Leakage Assessment Report for Xoodyak_R3_first_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/Tsinghua/Tsinghua_Report_HW_Xoodyak_RUB_v2.pdf.

[330] Abdulgadir A, Kaps JP, Gaj K (2022) Leakage Assessment on Xoodyak_R3_first_order, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/GMU/GMU_Report_HW_Xoodyak_first_order_RUB_v2.pdf.

[331] Batina L, Buhan I, Chmielewski L, Gunnarsdóttir E, Jahandideh V, Stock T, Weissbart L (2022) Side-Channel Evaluation Report on Implementations of Several NIST LWC Finalists, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/Radboud/Radboud_Report_SW_3_candidates.pdf.

[332] Peng S, Liu J, Yang B, Zhu W, Liu L (2022) Leakage Assessment Report for First-order Masked GIFT-COFB, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/Tsinghua/Tsinghua_Report_SW_GIFT-COFB_AA_v2.pdf.

[333] Peng S, Liu J, Yang B, Zhu W, Liu L (2022) Leakage Assessment Report for First-order Masked Romulus, Submission to GMU. Available at https://cryptography.gmu.edu/athena/LWC/Reports/Tsinghua/Tsinghua_Report_SW_Romulus_AA.pdf.

[334] Karl P, Gruber M (2021) A Survey on the Application of Fault Analysis on Lightweight Cryptography. *11th IFIP International Conference on New Technologies, Mobility and Security, NTMS 2021, Paris, France, April 19-21, 2021* (IEEE), pp 1–3. https://doi.org/10.1109/NTMS49979.2021.9432667

[335] Madushan H, Salam I, Alawatugoda J (2022) A Review of the NIST Lightweight Cryptography Finalists and Their Fault Analyses. *Electronics* 11(24). https://doi.org/10.3390/electronics11244199

[336] Ramezanpour K, Abdulgadir A, Diehl W, Kaps JP, Ampadu P (2020) Active and Passive Side-Channel Key Recovery Attacks on Ascon, NIST Lightweight Cryptography Workshop 2020. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/active-passive-recovery-attacks-ascon-lwc2020.pdf.

[337] Jacob J, Joseph J, Abinshad MK, Ambili KN, Jose J (2020) Prevention of Fault Attacks in ASCON Authenticated Cipher Using Cellular Automata. *Cellular Automata - 14th International Conference on Cellular Automata for Research and Industry, ACRI 2020, Lodz, Poland, December 2-4, 2020, Proceedings*, eds Gwizdalla TM, Manzoni L, Sirakoulis GC, Bandini S, Podlaski K (Springer), *Lecture Notes in Computer Science*, Vol. 12599, pp 18–25.

https://doi.org/10.1007/978-3-030-69480-7_3

[338] Surya G, Maistri P, Sankaran S (2020) Local Clock Glitching Fault Injection with Application to the ASCON Cipher. *IEEE International Symposium on Smart Electronic Systems, iSES 2020 (Formerly iNiS), Chennai, India, December 14-16, 2020* (IEEE), pp 271–276. https://doi.org/10.1109/iSES50453.2020.00067

[339] Ramezanpour K, Ampadu P, Diehl W (2019) A Statistical Fault Analysis Methodology for the Ascon Authenticated Cipher. *IEEE International Symposium on Hardware Oriented Security and Trust, HOST 2019, McLean, VA, USA, May 5-10, 2019* (IEEE), pp 41–50. https://doi.org/10.1109/HST.2019.8741029

[340] Ramezanpour K, Ampadu P, Diehl W (2020) SCARL: Side-Channel Analysis with Reinforcement Learning on the Ascon Authenticated Cipher. *CoRR* abs/2006.03995. Available at https://arxiv.org/abs/2006.03995.

[341] Joshi P, Mazumdar B (2019) A Subset Fault Analysis of ASCON, Cryptology ePrint Archive, Paper 2019/1370. Available at https://eprint.iacr.org/2019/1370.

[342] Joshi P, Mazumdar B (2021) Single Event Transient Fault Analysis of ELEPHANT Cipher. *CoRR* abs/2106.09536. Available at https://arxiv.org/abs/2106.09536.

[343] Ambili KN, Jose J (2022) Inapplicability of Differential Fault Attacks against Cellular Automata based Lightweight Authenticated Cipher, Cryptology ePrint Archive, Paper 2022/042. Available at https://eprint.iacr.org/2022/042.

[344] Luo H, Chen W, Ming X, Wu Y (2021) General Differential Fault Attack on PRESENT and GIFT Cipher With Nibble. *IEEE Access* 9:37697–37706. https://doi.org/10.1109/ACCESS.2021.3062665

[345] Liu S, Guan J, Hu B (2022) Fault Attacks on Authenticated Encryption Modes for GIFT. *IET Inf Secur* 16(1):51–63. https://doi.org/10.1049/ise2.12041

[346] Salam MI, Ooi TH, Xue L, Yau W, Pieprzyk J, Phan RC (2021) Random Differential Fault Attacks on the Lightweight Authenticated Encryption Stream Cipher Grain-128AEAD. *IEEE Access* 9:72568–72586. https://doi.org/10.1109/ACCESS.2021.3078845

[347] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2022) Update on the Performance and Mode-level Properties of ISAP, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/update-on-the-performance-and-mode-level-properties-of-isap.pdf.

[348] Jana A, Paul G (2022) Differential Fault Attack on PHOTON-Beetle. *Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security* ASHES'22 (Association for Computing Machinery, New York, NY, USA), p 25–34. https://doi.org/10.1145/3560834.3563824

[349] Vafaei N, Bagheri N, Saha S, Mukhopadhyay D (2018) Differential Fault Attack on SKINNY Block Cipher. *Security, Privacy, and Applied Cryptography Engineering - 8th International Conference, SPACE 2018, Kanpur, India, December 15-19, 2018, Proceedings*, eds Chattopadhyay A, Rebeiro C, Yarom Y (Springer), *Lecture Notes in Computer Science*, Vol. 11348, pp 177–197. https://doi.org/10.1007/978-3-030-05072-6_11

[350] Miteloudi K, Batina L, Daemen J, Mentens N (2021) ROCKY: Rotation Countermeasure for the Protection of Keys and Other Sensitive Data. *Embedded Computer Systems: Architectures, Modeling, and Simulation - 21st International Conference, SAMOS 2021, Virtual Event, July 4-8, 2021, Proceedings*, eds Orailoglu A, Jung M, Reichenbach M

(Springer), *Lecture Notes in Computer Science*, Vol. 13227, pp 288–299. https://doi.org/10.1007/978-3-031-04580-6_19

[351] Bellizia D, Bronchain O, Cassiers G, Grosso V, Guo C, Momin C, Pereira O, Peters T, Standaert FX (2020) Mode-Level vs. Implementation-Level Physical Security in Symmetric Cryptography: A Practical Guide Through the Leakage-Resistance Jungle, Cryptology ePrint Archive, Report 2020/211. Available at https://eprint.iacr.org/2020/211.

[352] Bhasin S, Jap D, Ng WC, Sim SM (2022) Survey on the Effectiveness of DAPA-Related Attacks against Shift Register Based AEAD Schemes, Cryptology ePrint Archive, Paper 2022/561. Available at https://eprint.iacr.org/2022/561.

[353] Verhamme C, Cassiers G, Standaert FX (2022) Analyzing the Leakage Resistance of the NIST's Lightweight Crypto Competition's Finalists, Cryptology ePrint Archive, Paper 2022/1628. Available at https://eprint.iacr.org/2022/1628.

[354] Diehl W, Farahmand F, Abdulgadir A, Kaps JP, Gaj K (2019) Face-off between the CAESAR Lightweight Finalists: ACORN vs. Ascon, Cryptology ePrint Archive, Paper 2019/184. Available at https://eprint.iacr.org/2019/184.

[355] Abdulgadir A, Haeussler R, Lin S, Kaps JP, Gaj K (2022) Side-Channel Resistant Implementations of Three Finalists of the NIST Lightweight Cryptography Standardization Process: Elephant, TinyJAMBU, and Xoodyak, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/side-channel-resistant-implementations-of-three-finalists-of-the-nist-lwc-standardization-process.pdf.

[356] Aljuffri A, Reinbrecht C, Hamdioui S, Taouil M, Sepulveda J (2021) Balanced Dual-Mask Protection Scheme for GIFT Cipher Against Power Attacks. *2022 IEEE 40th VLSI Test Symposium (VTS)* (IEEE Computer Society, Los Alamitos, CA, USA), pp 1–6. https://doi.org/10.1109/VTS52500.2021.9794230

[357] Reinbrecht C, Aljuffri A, Hamdioui S, Taouil M, Sepúlveda J (2021) GRINCH: A Cache Attack against GIFT Lightweight Cipher. *Design, Automation & Test in Europe Conference & Exhibition, DATE 2021, Grenoble, France, February 1-5, 2021* (IEEE), pp 549–554. https://doi.org/10.23919/DATE51398.2021.9474201

[358] Dobraunig C, Eichlseder M, Mangard S, Mendel F, Mennink B, Primas R, Unterluggauer T (2020) Updates on the Implementation Security of ISAP, NIST Lightweight Cryptography Workshop 2020. Available at https://csrc.nist.gov/CSRC/media/Events/lightweight-cryptography-workshop-2020/documents/papers/updates-on-ISAP-lwc2020.pdf.

[359] Khairallah M, Bhasin S (2022) Hardware Implementations of Romulus: Exploring Nonce Misuse Resistance and Boolean Masking, Fifth NIST Lightweight Cryptography Workshop 2022. Available at https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/hardware-implementations-of-romulus.pdf.

[360] Khairallah M, Bhasin S (2022) Hardware Implementation of Masked SKINNY SBox with Application to AEAD. *Security, Privacy, and Applied Cryptography Engineering - 12th International Conference, SPACE 2022, Jaipur, India, December 9-12, 2022, Proceedings*, eds Batina L, Picek S, Mondal M (Springer), *Lecture Notes in Computer Science*, Vol. 13783, pp 50–69. https://doi.org/10.1007/978-3-031-22829-2_3

[361] Abdulgadir A (2021) Secure Hardware Implementations of Lightweight and Post-Quantum Cryptography; Trade-Offs, Applicability, and Tools. *PhD Thesis, George Mason University* Available at http://mars.gmu.edu/handle/1920/12675.

A. List of Acronyms

Acronym	Definition
AD	**A**ssociated **D**ata
AE	**A**uthenticated **E**ncryption
AEAD	**A**uthenticated **E**ncryption with **A**ssociated **D**ata
AES	**A**dvanced **E**ncryption **S**tandard
AES-NI	**AES N**ew **I**nstructions
ARX	**A**ddition-**R**otation-**X**OR
ASIC	**A**pplication-**S**pecific **I**ntegrated **C**ircuit
CAESAR	**C**ompetition for **A**uthenticated **E**ncryption: **S**ecurity, **A**pplicability, and **R**obustness
CERG	**C**ryptographic **E**ngineering **R**esearch **G**roup
CMOS	**C**omplementary **M**etal-**O**xide-**S**emiconductor
COFB	**CO**mbined **F**eed**B**ack
CPA	**C**orrelation **P**ower **A**nalysis
DAPA	**D**ifferential **A**nalysis aided **P**ower **A**ttack
DBL	**D**ouble-**B**lock-**L**ength
DFA	**D**ifferential **F**ault **A**ttack
DPA	**D**ifferential **P**ower **A**nalysis
eBACS	ECRYPT **B**enchm**A**rking of **C**ryptographic **S**ystems
eBAEAD	ECRYPT **B**enchmarking of **AEAD** algorithms
eBASH	ECRYPT **B**enchmarking of **A**ll **S**ubmitted **H**ashes
FELICS	**F**air **E**valuation of **LI**ghtweight **C**ryptographic **S**ystems
FPGA	**F**ield-**P**rogrammable **G**ate **A**rray
GCM	**G**alois/**C**ounter **M**ode
GE	**G**ate **E**quivalent
GMU	**G**eorge **M**ason **U**niversity
IND-CCA	**IND**istinguishability security under the **C**hosen **C**iphertext **A**ttack
IND-CPA	**IND**istinguishability security under the **C**hosen **P**laintext **A**ttack
INT-CTXT	**INT**egrity of **C**ipher**T**e**XT**s
INT-RUP	**INT**egrity security under the **RUP**
ISE	**I**nstruction **S**et **E**xtension
ISW	**I**shai-**S**ahai-**W**agner
KB	**K**ilo**B**yte
LFSR	**L**inear **F**eedback **S**hift **R**egister
LUT	**L**ook **U**p **T**able
MAC	**M**essage **A**uthentication **C**ode
MB	**M**ega**B**yte
MCU	**M**icro**C**ontroller **U**nit
MDPH	**M**erkle-**D**amgård with **P**ermutation using **H**irose's DBL compression function

Acronym	Definition
MILP	**M**ix-**I**nteger **L**inear **P**rogramming
MRAE	Nonce **M**isuse-**R**esistant **AE**
NAE	**N**once-based **AE**
NFSR	**N**on-linear **F**eedback **S**hift **R**egister
NIST	**N**ational **I**nstitute of **S**tandards and **T**echnology
NISTIR	**NIST I**nternal **R**eport
PRF	**P**seudo-**R**andom **F**unction
RAM	**R**andom-**A**ccess **M**emory
RFID	**R**adio-**F**requency **ID**entification
RISC	**R**educed **I**nstruction **S**et **C**omputer
ROM	**R**ead-**O**nly **M**emory
RTC	**R**eal **T**ime **C**lock
RUP	**R**eleasing **U**nverified **P**laintext
SCA	**S**ide-**C**hannel **A**ttack
SCARL	**S**ide **C**hannel **A**nalysis with **R**einforcement **L**earning
SHA	**S**ecure **H**ash **A**lgorithms
SIFA	**S**tatistical **I**neffective **F**ault **A**ttack
SIMD	**S**ingle **I**nstruction **M**ultiple **D**ata
SPA	**S**imple **P**ower **A**nalysis
SPN	**S**ubstitution–**P**ermutation **N**etwork
SSFA	**S**ub**S**et **F**ault **A**nalysis
STP	**S**imple **T**heorem **P**rover constraint solver
TA	**T**emplate **A**ttack
TBC	**T**weakable **B**lock **C**ipher
TLS	**T**ransport **L**ayer **S**ecurity
XOF	e**X**tendable-**O**utput **F**unction
XOR	e**X**clusive **OR**

B. NIST Software Benchmarking Results

This section contains selected results from the NIST software benchmarking effort on microcontollers. Additional plots can be found in the NIST Benchmarking of Lightweight Cryptographic Algorithms on Microcontrollers GitHub repository [253].

B.1. AEAD Size Comparison

AEAD size experiments compiled each AEAD implementation that supported three cases: encryption only (`enc`), decryption only (`dec`), and both encryption and decryption (`enc+dec`). Figure 5 shows the minimum flash sizes used by the primary AEAD variants of the finalists on various platforms.

The primary variants of ASCON, GIFT-COFB, and TinyJAMBU consistently demonstrated smaller code sizes than AES-GCM for binaries supporting `enc`, `dec`, or `enc+dec`. TinyJAMBU had the smallest implementations for `enc` and `dec` on all platforms and achieved the smallest `enc+dec` code size on six platforms. PHOTON-Beetle had the smallest `enc+dec` binary on the other two platforms. SPARKLE implementations that supported `enc` or `dec` were smaller than AES-GCM on all tested platforms, while implementations that supported `enc+dec` were smaller than AES-GCM on four platforms.

The primary variants of Elephant, ISAP, and Romulus did not exhibit a performance advantage over AES-GCM in terms of size measurements.

B.2. Hash Size Comparison

The minimum flash sizes for compiled hashing implementations are presented in Figure 6. The primary hashing variants of ASCON and SPARKLE had smaller implementations than SHA-256 on all test platforms. Xoodyak was smaller than SHA-256 on six of the platforms. As with the AEAD implementations, PHOTON-Beetle had a compact implementation on AVR-based MCUs and was relatively large on all others. Romulus was also more compact than AES-GCM on only two platforms.

B.3. Combined AEAD and Hashing Size

NIST requested that AEAD and hashing functionality be paired in submissions in the hopes that size could be reduced by sharing logic between AEAD and hashing functionalities. All candidates with hashing functionality contained common elements between AEAD and hash algorithms, such as a permutation or block cipher, that enabled both functionalities to be implemented with lower flash requirements than those required by implementing both functionalities separately.

Figure 7 provides a summary of flash use for the smallest implementation of each recommended AEAD and hash pairing on each MCU. Each of these implementations includes AEAD (both encryption and decryption) and hashing functionalities.

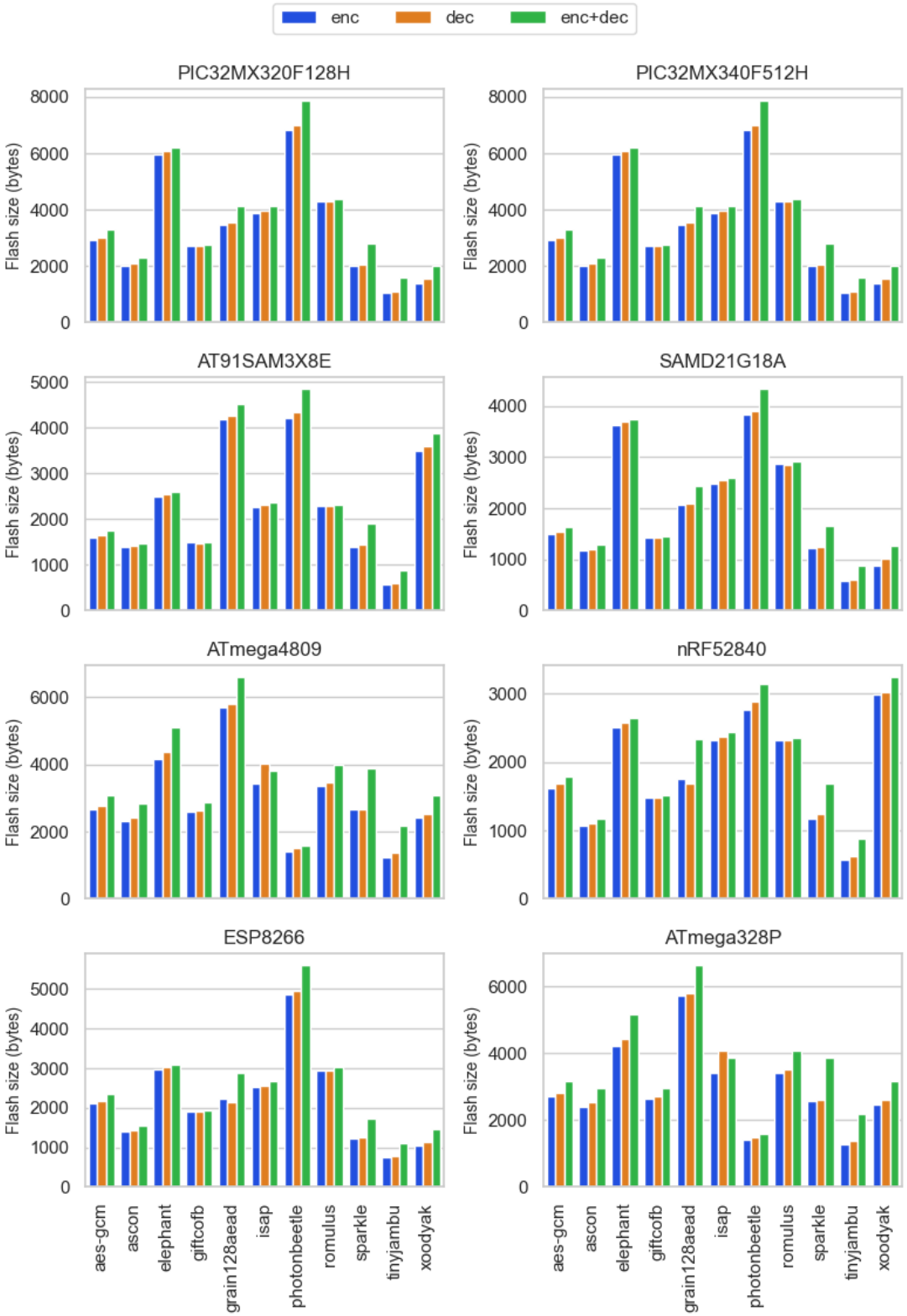


Fig. 5. Minimum flash size used by primary AEAD variants

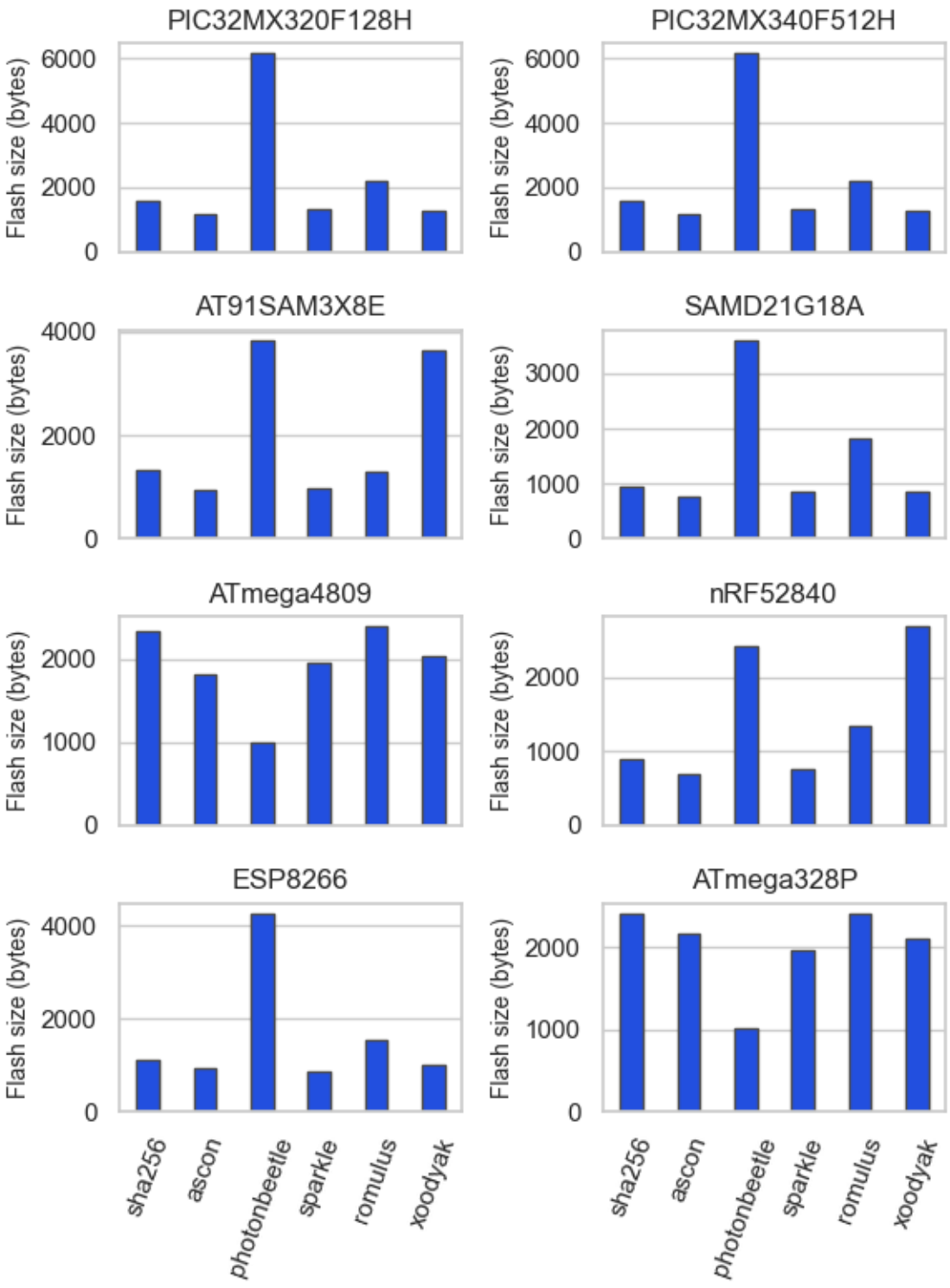


Fig. 6. Minimum flash size used by primary hashing variants

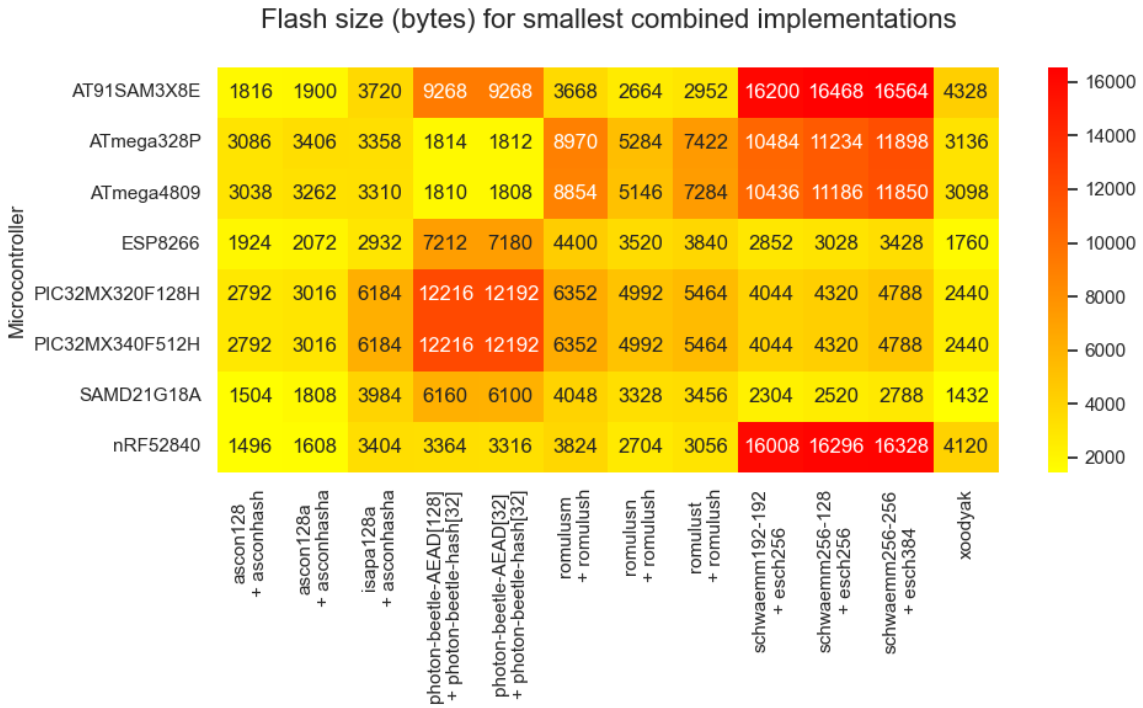


Fig. 7. Flash size used by smallest combined AEAD and hashing implementations on each MCU.

B.4. Time vs. Size Explorations

To gain insight into implementation trade-offs, the AEAD implementation's encryption time and its flash size requirements were compared. An implementation's hashing time and flash size requirements were also compared, where applicable.

The timing metrics varied by platform. A summary of the timer function and metric used for each microcontroller[4] is provided in Table 15. The SysTick timer used a 24-bit register and experienced underflows while timing cryptographic operations. To account for this, the cycle counts returned by the platform were adjusted. For example, consider two messages with lengths m and $m+8$ that are both encrypted with the same AD. If the cycle count for the larger message is less than the count for the shorter message, an underflow may have occurred. There were instances where this was true but an underflow did not appear to be present (e.g., at a boundary where less padding was needed), so a range of cycle counts was considered where no adjustment was made if the lower cycle count was within the given range. Results presented here use a 500-cycle range for AEAD and 5000-cycle range for hashing.

Table 15. Timers and timing metrics used in NIST benchmarks

Microcontroller	*Timer*	*Timing metric*
ATmega328P	micros	microseconds
ATmega4809	micros	microseconds
SAMD21G18A	SysTick	cycles
nRF52840	RTC	cycles
PIC32MX340F512H	micros	microseconds
ESP8266	RTC	cycles
AT91SAM3X8E	micros	microseconds

ASCON. ASCON demonstrated favorable performance compared to current NIST standards, AES-GCM and SHA-2. ASCON implementations were able to encrypt and decrypt significantly faster than similarly sized AES-GCM implementations, as well as achieve smaller times than the fastest tested AES-GCM implementations (Figure 8a). The variants of ASCON that provide hash and XOF functionalities also performed well when compared with SHA-256. In particular, hashing was performed faster when ASCON and SHA-256 implementations had similar flash requirements (Figure 8b).

Elephant. The smallest implementations of Elephant's primary variant were larger than those of AES-GCM, and Elephant generally did not demonstrate performance advantages when its size was very close to the smallest AES-GCM implementations. However, the implementations of the KECCAK-based Delirium variant outperformed the primary variant, Dumbo, in terms of flash size and encryption speed. Delirium was the only variant of

[4]Note that if comparisons across platforms are desired, cycles can be converted to microseconds using a microcontroller's clock speed.

Elephant that demonstrated lower flash usage for similar speeds and a significant speedup over AES-GCM with a moderate increase of flash memory (see Figure 9).

GIFT-COFB. There were implementations of GIFT-COFB that required less time than AES-GCM implementations of similar size on all boards. Further, encryption with GIFT-COFB took less time than the fastest AES-GCM implementations. However, GIFT-COFB implementations that had similar size requirements to the smallest AES-GCM implementations were slower on the two AVR-based MCUs (see Figure 10).

Grain-128AEAD. None of the benchmarked implementations of Grain-128AEAD were more compact than the smallest AES-GCM implementation on the tested platforms. When considering implementations most similar in size, some Grain-128AEAD implementations demonstrated faster encryption times (see Figure 11).

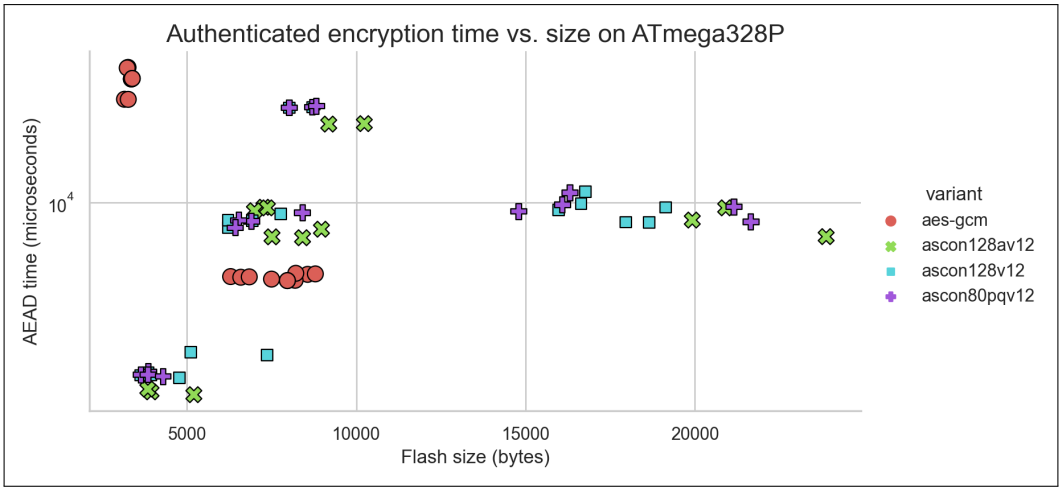
ISAP. Figure 12 shows that ISAP implementations had a large range of flash sizes, with most executables using far more flash than AES-GCM. The only variant that was competitive with AES-GCM was the primary variant, where performance advantages could only be seen when the smallest implementations of AES-GCM and ISAP were compared.

PHOTON-Beetle. PHOTON-Beetle implementations performed best on the two AVR platforms, ATmega4809 and ATmega328P, but did not perform well compared to AES-GCM on all other tested platforms. Figure 13 provides an example of the tradeoff space disparities between AVR and non-AVR platforms. There are several instances in Figure 13a showing that PHOTON-Beetle had smaller implementations, as well as faster encryption than AES-GCM for implementations of similar size. However, there are no implementations smaller than AES-GCM in Figure 13b, and none of the implementations are faster than all AES-GCM implementations of similar size. PHOTON-Beetle-Hash[32] did not show performance advantages over SHA-256.

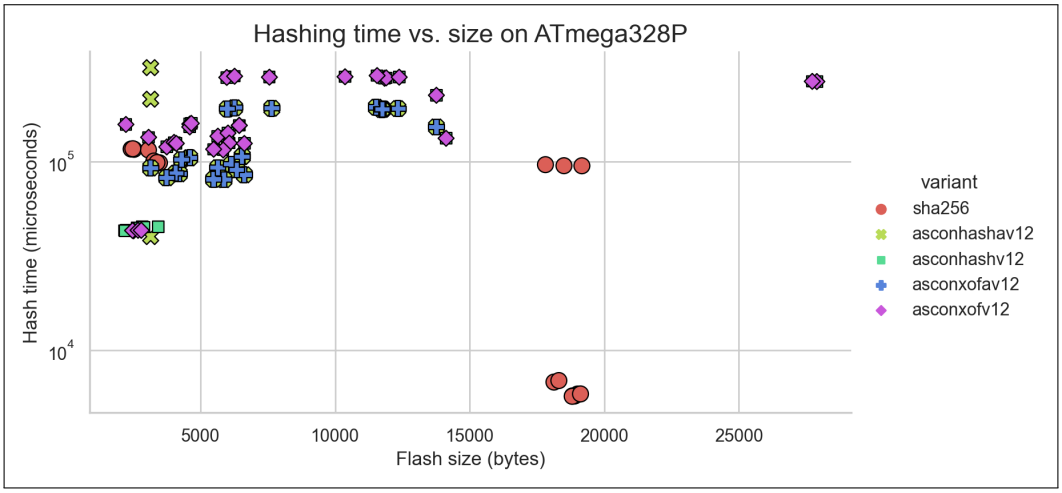
Romulus. Several implementations of Romulus-N demonstrated faster encryption times compared to AES-GCM implementations with similar size. This can be seen in Figure 14a when the flash use exceeded 7000 bytes. Romulus also demonstrated more compact implementations compared to AES-GCM implementations that had similar speed. Romulus-M implementations were larger and/or slower than those of Romulus-N, but were still competitive. Romulus-H implementations did not demonstrate performance advantages over SHA-256 implementations.

SPARKLE. Implementations for all variants of SCHWAEMM showed performance advantages over AES-GCM. Implementations of ESCH256 were generally competitive with those of SHA-2. The speed of compact hash implementations was particularly favorable on AVR platforms.

TinyJAMBU. All variants of TinyJAMBU demonstrated implementations that were significantly faster than AES-GCM. Further, there were implementations for all variants that were simultaneously smaller and faster than AES-GCM (see Figure 16a) on all but the AVR platforms, where timing information for the smallest implementations was not available.



(a) Authenticated encryption time vs. size with 16-byte AD and 16-byte message



(b) Hashing time vs. size for 512-byte message

Fig. 8. Execution time vs. size explorations for ASCON

In many cases on AVR platforms the decrypted ciphertext did not match the plaintext and AD and the results were omitted from further processing. This effect was limited to the AVR platforms and the same source code was successfully compiled and executed on the other platforms. The implementations that did run correctly were all significantly faster than AES-GCM implementations with similar size, as shown in Figure 16b.

Xoodyak. Xoodyak implementations were generally able to achieve faster encryption speeds than AES-GCM, but were not able to demonstrate more compact implementations on all test platforms (see Figure 17). On AVR platforms, this was reversed – implementations were smaller, but not faster. Faster hashing than SHA-2 was demonstrated for most platforms.

B.5. Execution Time Comparison

The NIST benchmarking effort compared the speed of encryption and decryption with various AD and message lengths. The fastest and smallest implementations for each candidate were compared to the fastest and smallest tested implementations of AES-GCM, respectively.

Determining the smallest implementation for each variant is straightforward – it is simply the executable using the least flash that successfully completed the timing experiment. Results for the smallest implementations are summarized as heatmaps in Figures 18 - 25. The relative size of the implementation compared to the smallest AES-GCM implementation appears beside the submission name in each heatmap title.

Results for the fastest implementations are summarized as heatmaps in Figures 26 - 33. Note that results presented in these Figures may contain data from multiple implementations, as the fastest implementation varied depending on the input sizes.

The heatmaps compare the encryption times of the finalists to that of AES-GCM for various message and AD sizes. The value in each cell represents an execution time ratio, calculated by dividing the authenticated encryption time of the finalists by that of AES-GCM for the given input sizes. Unlike the results presented in Figure 3, smaller values are better. Each cell is also colored relative to this value with Matplotlib's seismic palette, where blue indicates that the candidate is faster than AES-GCM.

ASCON. ASCON demonstrated clear speed advantages in these experiments. The fastest implementation of ASCON on the ESP8266 platform was the only case where AES-GCM is faster (see Figure 33), but it should be noted that the fastest AES-GCM implementations were much larger than the fastest ASCON implementations. The smallest implementation heatmap from the same platform showed a far different picture, where ASCON demonstrated 25x and 33x speedups over AES-GCM, depending on the input sizes, while using less flash memory (see Figure 25).

Elephant. Elephant was slower than AES-GCM in most cases. The relative execution time

of the smallest implementation was favorable on three platforms, but with increased flash usage. The most favorable relative execution times were obtained on the ATmega328P using an implementation that was about 54.6% larger than the smallest AES-GCM implementation (see Figure 19).

GIFT-COFB. Timing results for GIFT-COFB were generally favorable. The smallest implementations of GIFT-COFB were faster than AES-GCM on five of the eight platforms (Figures 21 – 25), while the fastest implementations showed faster encryption on seven platforms (Figures 26 – 32).

Grain-128AEAD. Grain-128AEAD did well for the smallest implementations, where its relative execution time was favorable on seven platforms. It should be noted, however, that these implementations were also larger than AES-GCM. The fastest implementations did not clearly demonstrate speed advantages.

ISAP. ISAP generally had speed advantages for smaller implementations, with better performance as AD and message sizes increased. All of these ISAP implementations were larger than the AES-GCM implementations.

PHOTON-Beetle. The smallest implementations of PHOTON-Beetle demonstrated favorable relative execution times on all platforms, sometimes at the cost of much larger implementations. The fastest implementations only demonstrated performance advantages on the tested AVR platforms (Figures 26 and 27).

Romulus. Encryption with the smallest implementations of Romulus was generally faster than AES-GCM in Figures 23-25 and slower in Figures 18-20. Relative execution times were noticeably worse for empty AD and best when the lengths of AD and message were either 8 bytes or 16 bytes. In addition, the smallest implementations of Romulus were larger than the smallest implementations of AES-GCM. The fastest implementations generally outperformed AES-GCM on three of the test platforms.

SPARKLE. The smallest SPARKLE implementations outperformed AES-GCM in both speed and size. The fastest implementations showed favorable relative execution time on all platforms except the ESP8266, where SPARKLE was slower for some of the inputs. The difference in performance, visible in Figure 33, arises due to different SPARKLE implementations being fastest for particular input lengths.

TinyJAMBU. TinyJAMBU demonstrated clear speed advantages over AES-GCM in all tested cases.

Xoodyak. The smallest Xoodyak implementations demonstrated speed advantages over AES-GCM. However, Xoodyak did not outperform AES-GCM for all input sizes when its code was smaller than that of AES-GCM (shown in Figures 18 and 19). The fastest implementations generally had favorable relative execution times, with the exception of the ATmega4809 (see Figure 26).

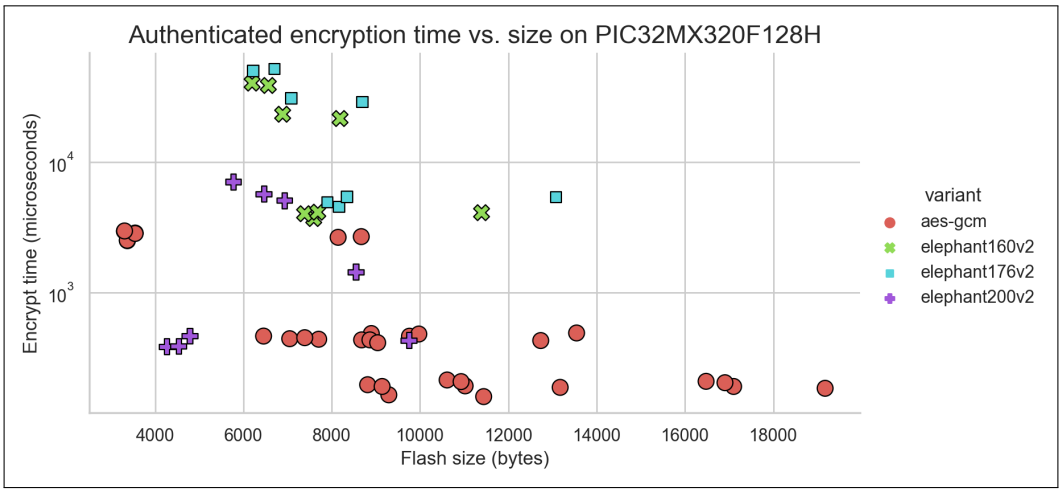


Fig. 9. Authenticated encryption time vs. size exploration for Elephant with 16-byte AD and 16-byte message

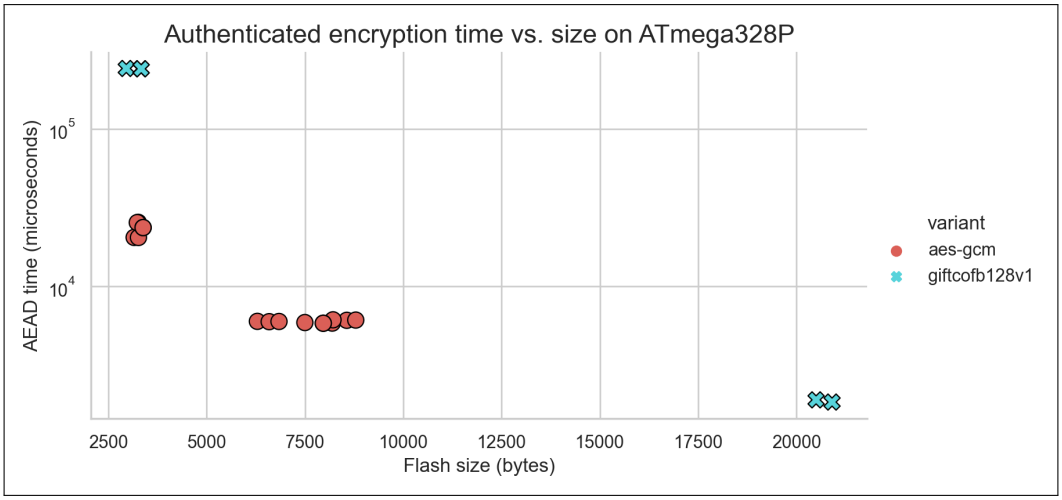


Fig. 10. Authenticated encryption time vs. size exploration for GIFT-COFB with 16-byte AD and 16-byte message

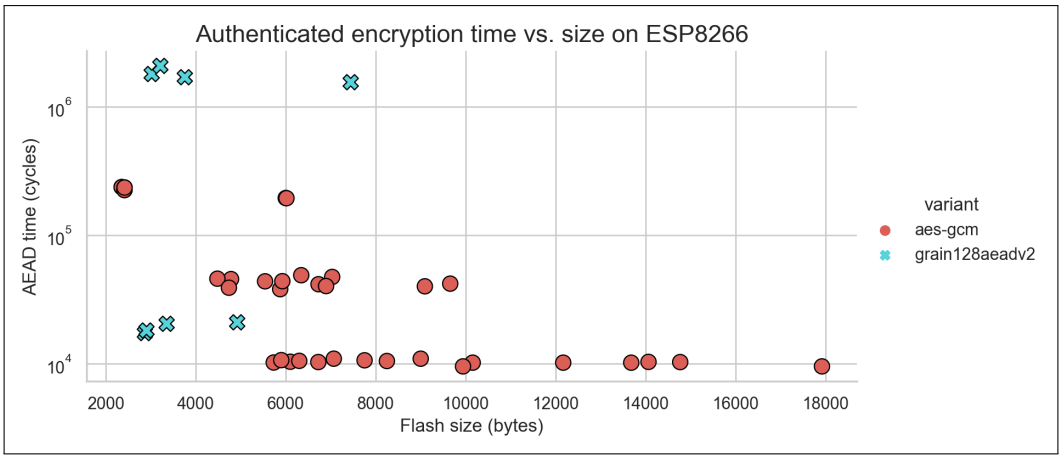


Fig. 11. Authenticated encryption time vs. size exploration for Grain-128AEAD with 16-byte AD and 16-byte message

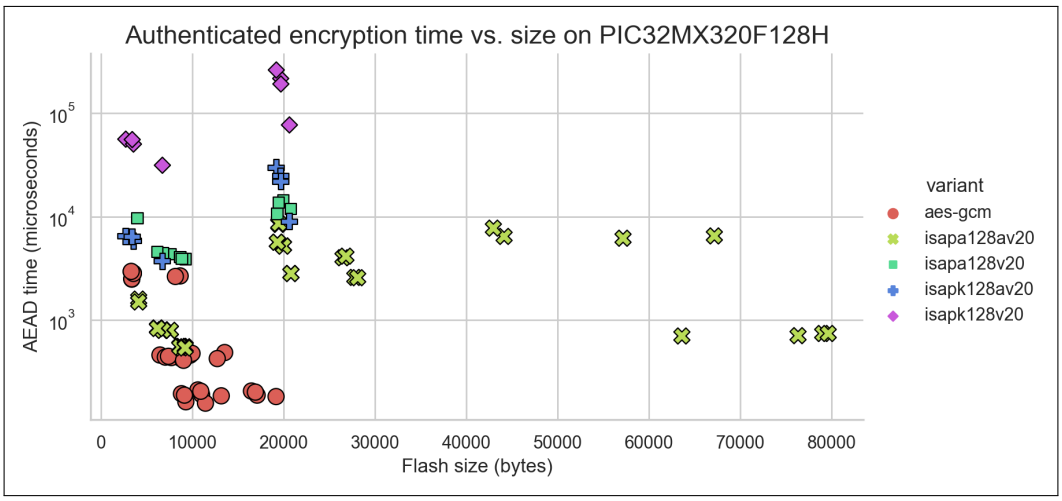
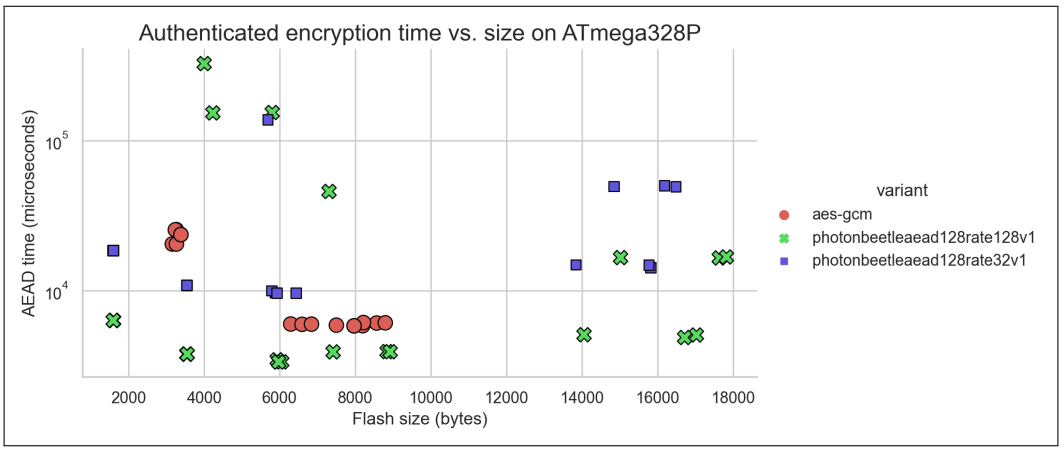
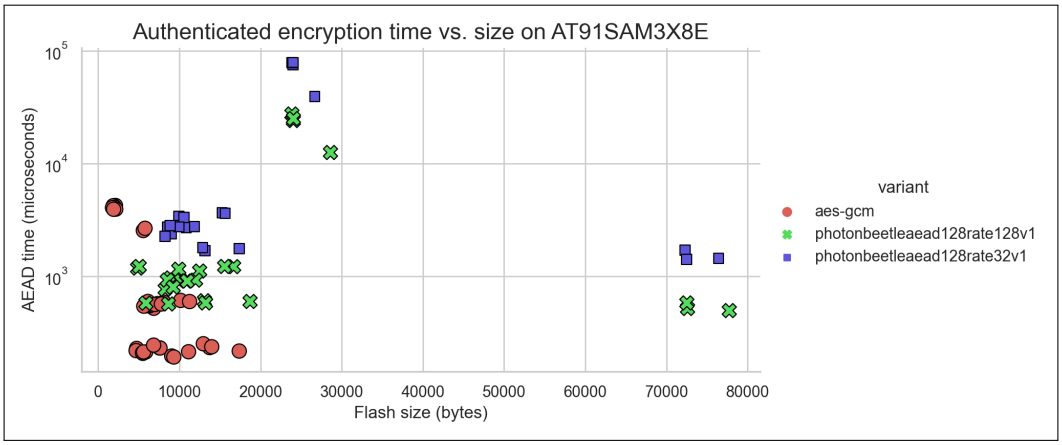


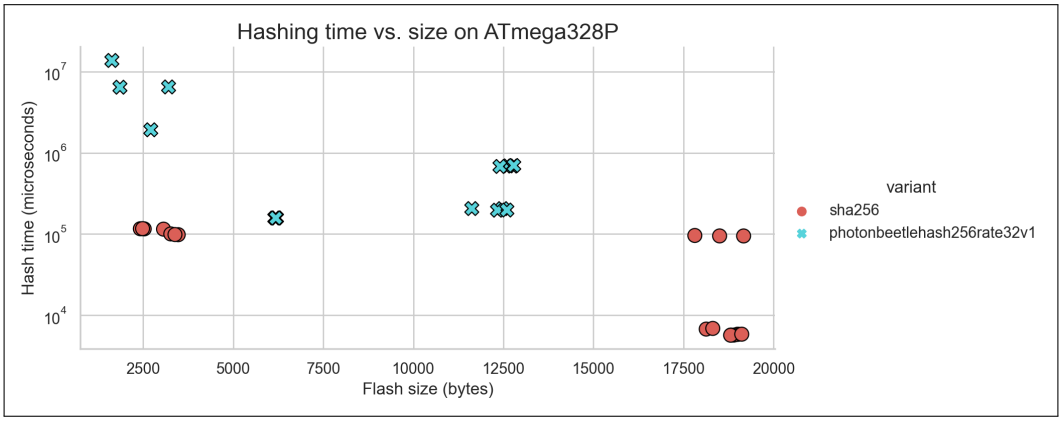
Fig. 12. Authenticated encryption time vs. size exploration for ISAP with 16-byte AD and 16-byte message



(a) Authenticated encryption time vs. size with with 16-byte AD and 16-byte message on AVR platform

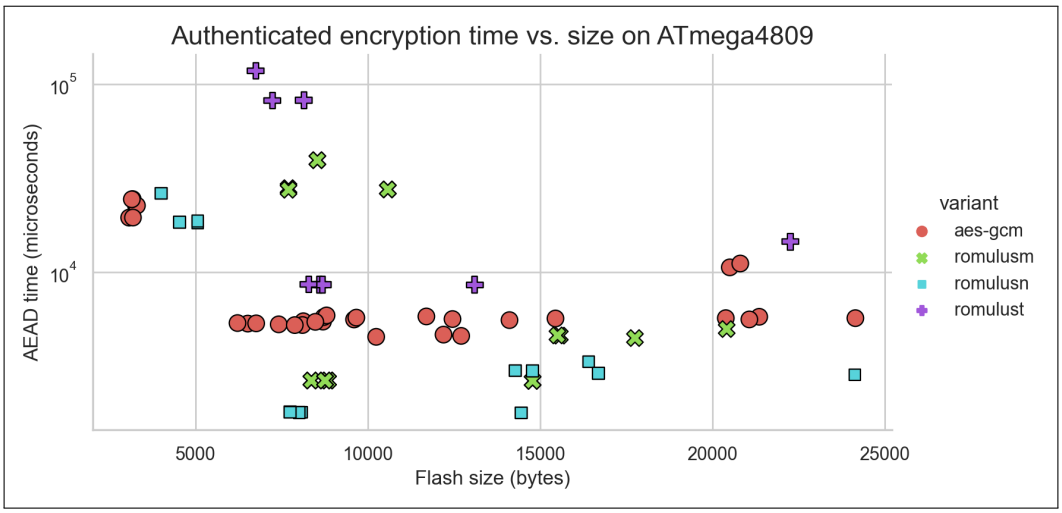


(b) Authenticated encryption time vs. size with with 16-byte AD and 16-byte message on ARM platform

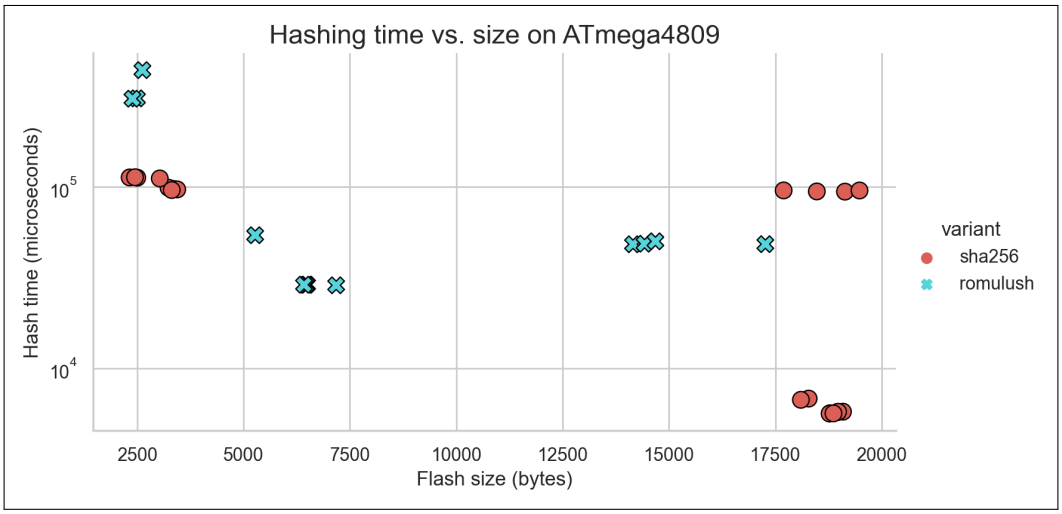


(c) Hash time vs. size with with 512-byte message on AVR platform

Fig. 13. Execution time vs. size exploration for PHOTON-Beetle

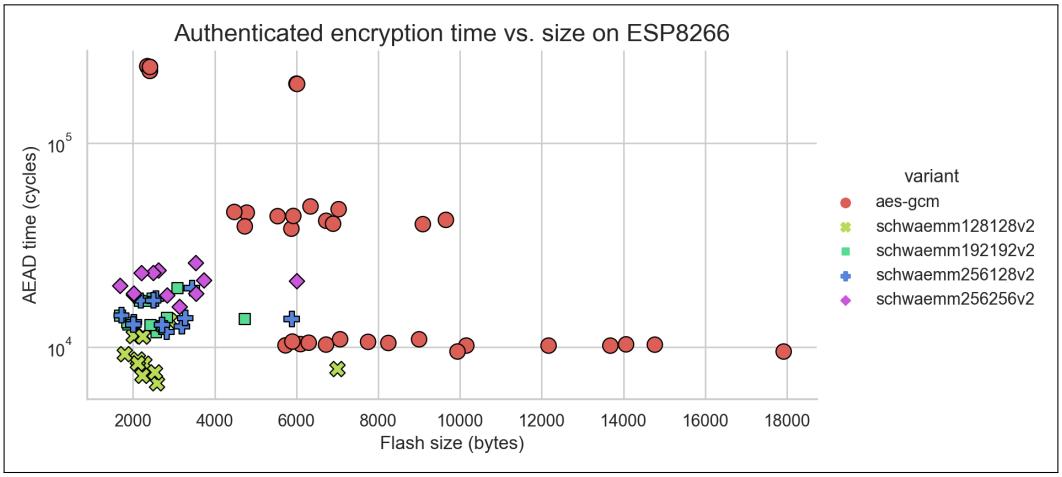


(a) Authenticated encryption time vs. size with 16-byte AD and 16-byte message

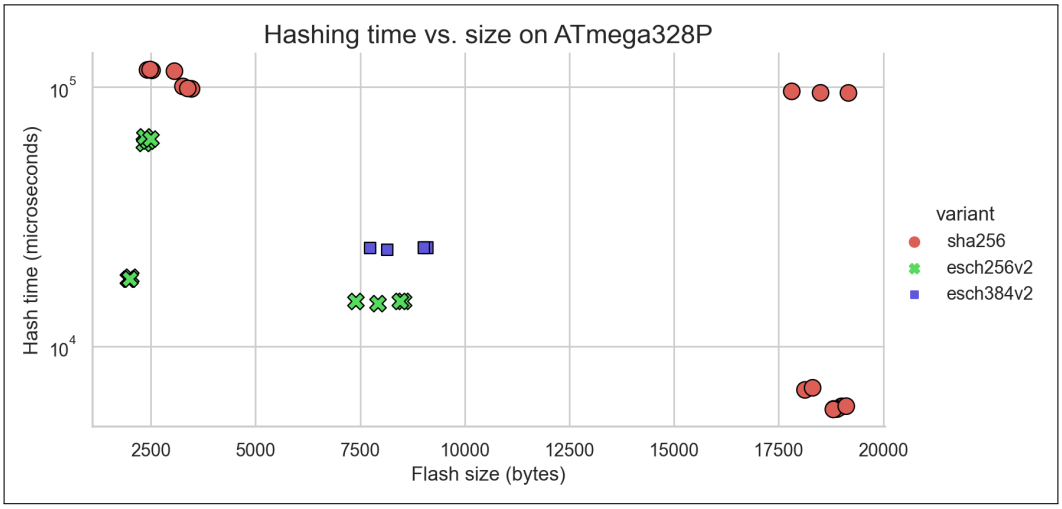


(b) Hashing time vs. size with 512-byte message

Fig. 14. Execution time vs. size exploration for Romulus

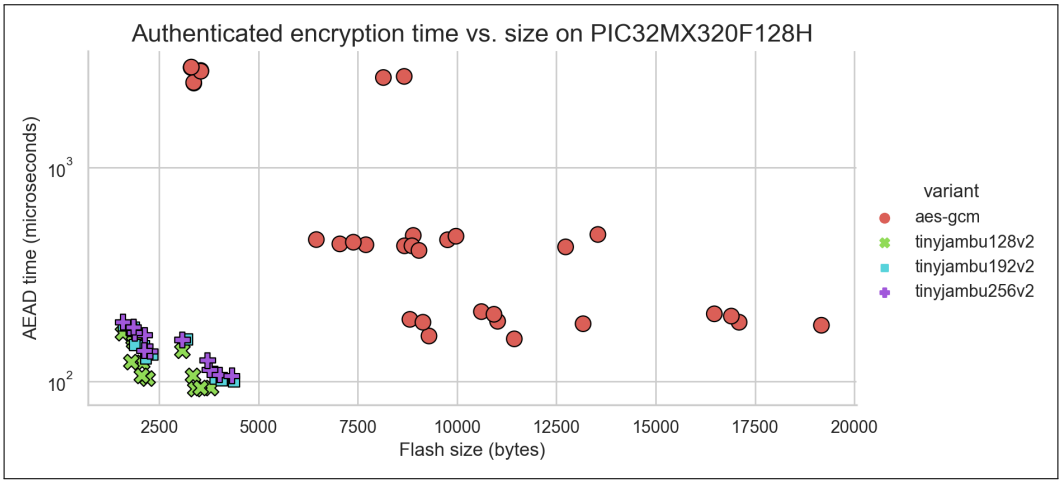


(a) Authenticated encryption time vs. size with 16-byte AD and 16-byte message

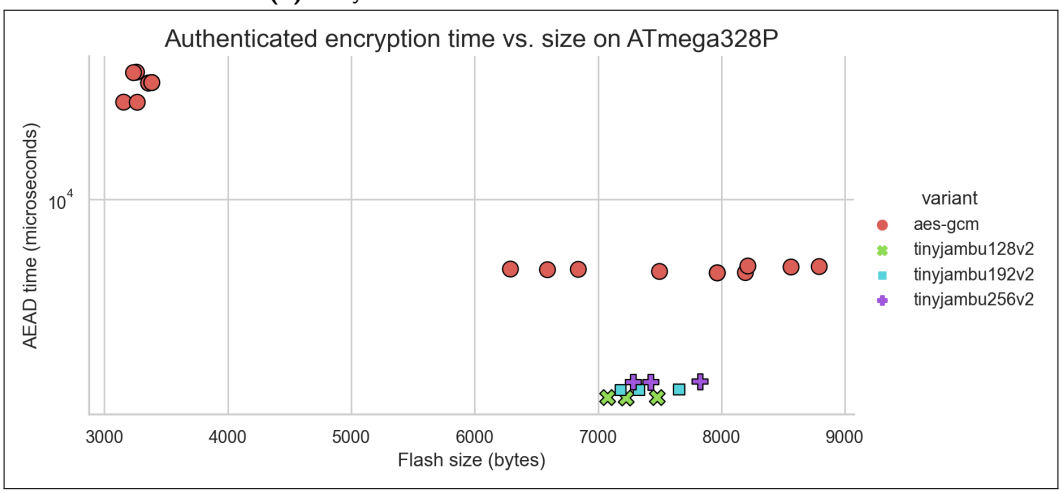


(b) Hashing time vs. size with 512-byte message

Fig. 15. Execution time vs. size exploration for SPARKLE

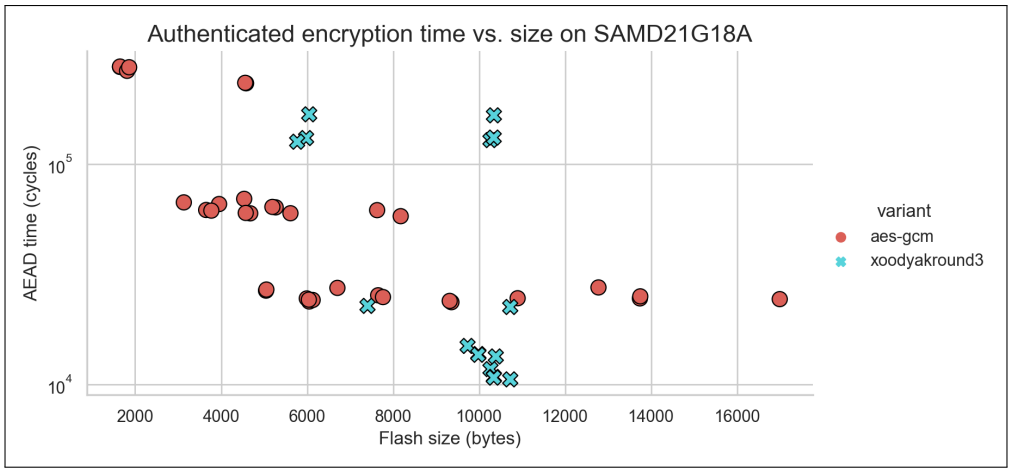


(a) TinyJAMBU variants on MIPS-based MCU

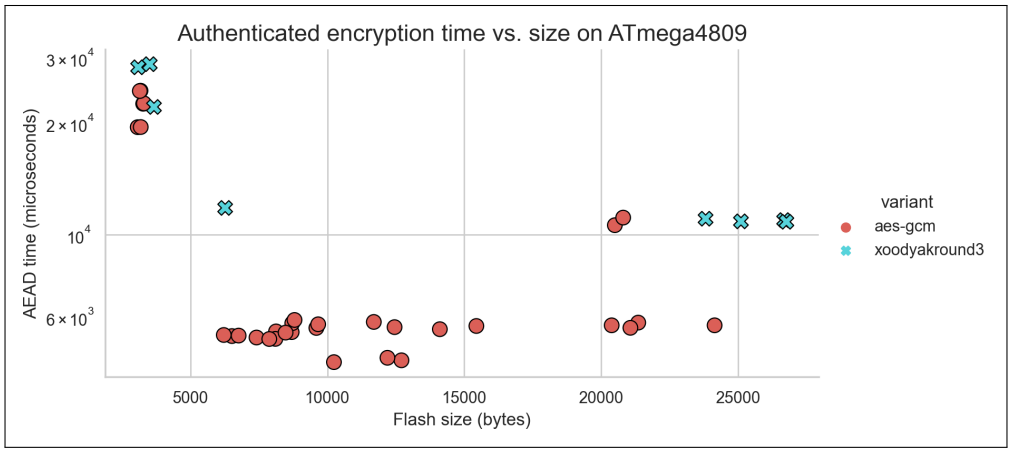


(b) TinyJAMBU variants on AVR-based MCU

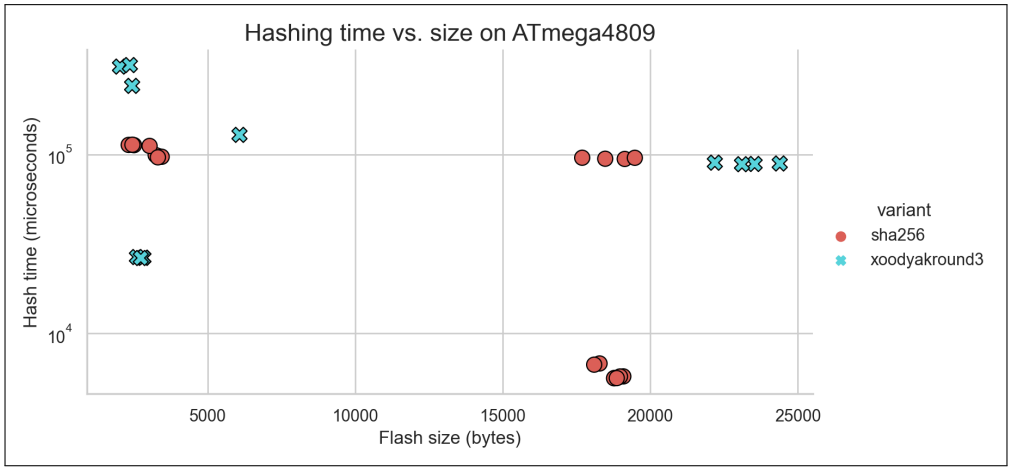
Fig. 16. Authenticated encryption time vs. size exploration for TinyJAMBU with 16-byte AD and 16-byte message



(a) Authenticated encryption time vs. size with 16-byte AD and 16-byte message on an ARM platform



(b) Authenticated encryption time vs. size with 16-byte AD and 16-byte message on an AVR platform



(c) Hashing time vs. size with 512-byte message on an AVR platform

Fig. 17. Execution time vs. size exploration for Xoodyak

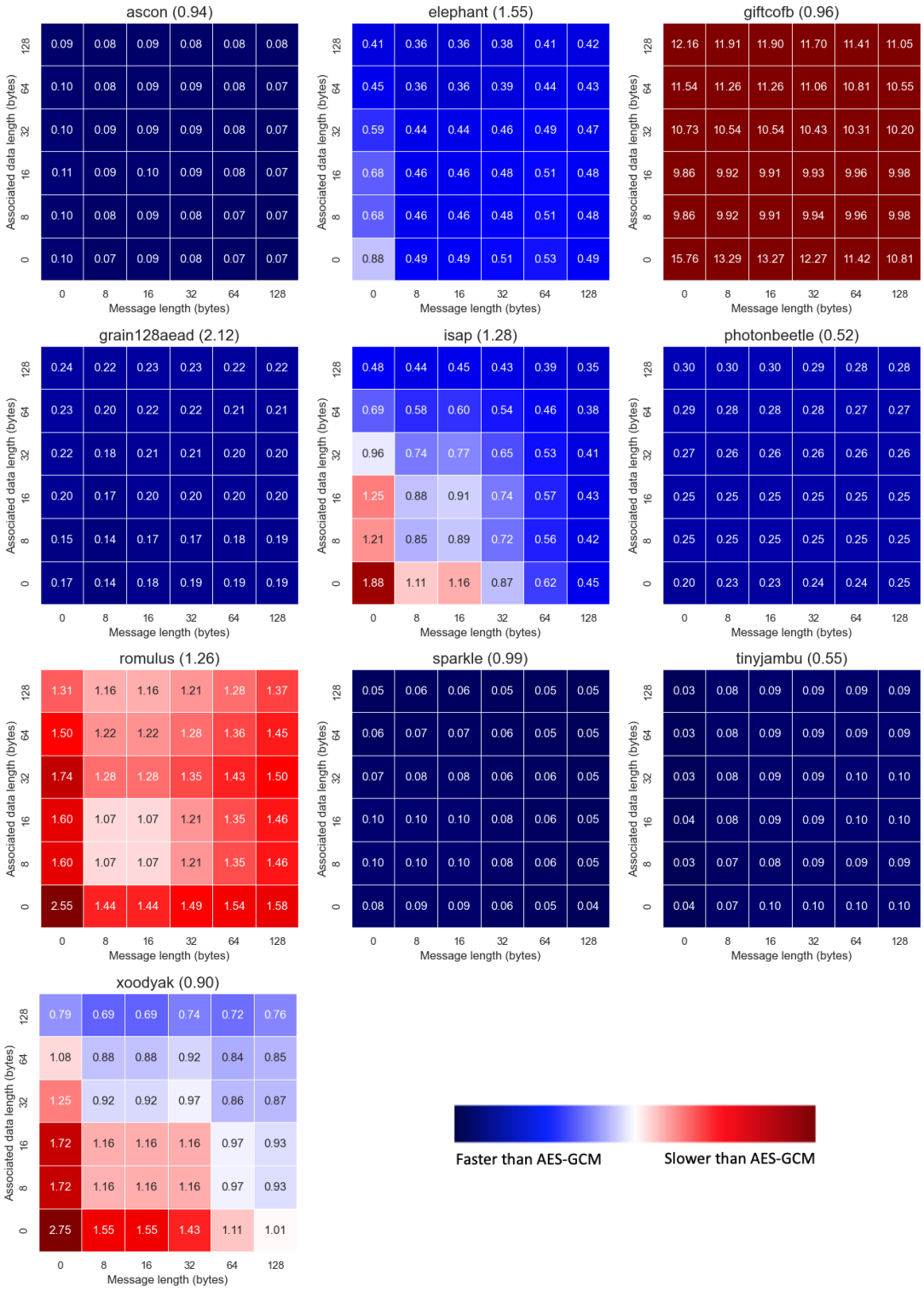


Fig. 18. Execution time ratio of smallest primary AEAD implementations to AES-GCM on ATmega4809

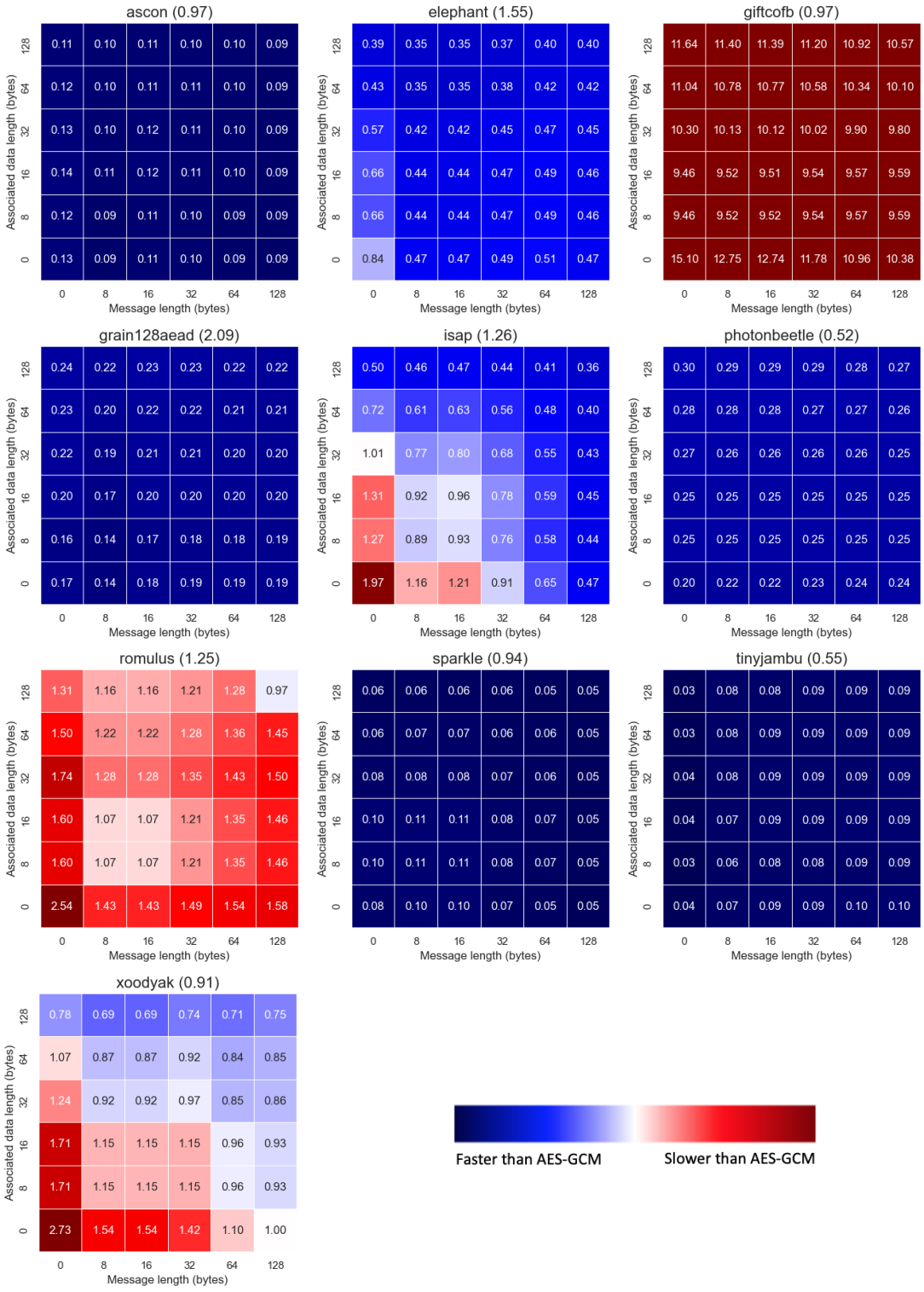


Fig. 19. Execution time ratio of smallest primary AEAD implementations to AES-GCM on ATmega328P

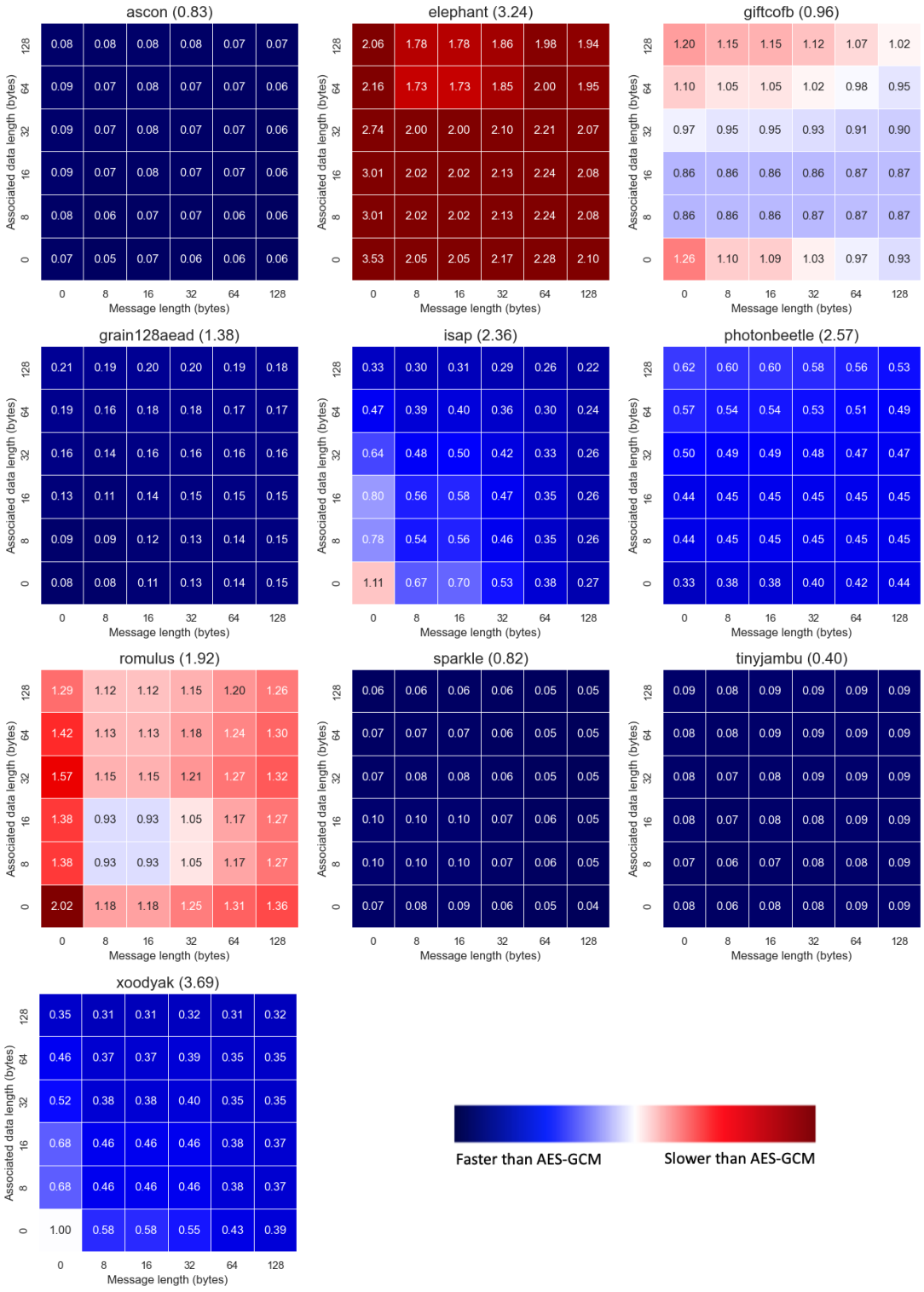


Fig. 20. Execution time ratio of smallest primary AEAD implementations to AES-GCM on SAMD21G18A

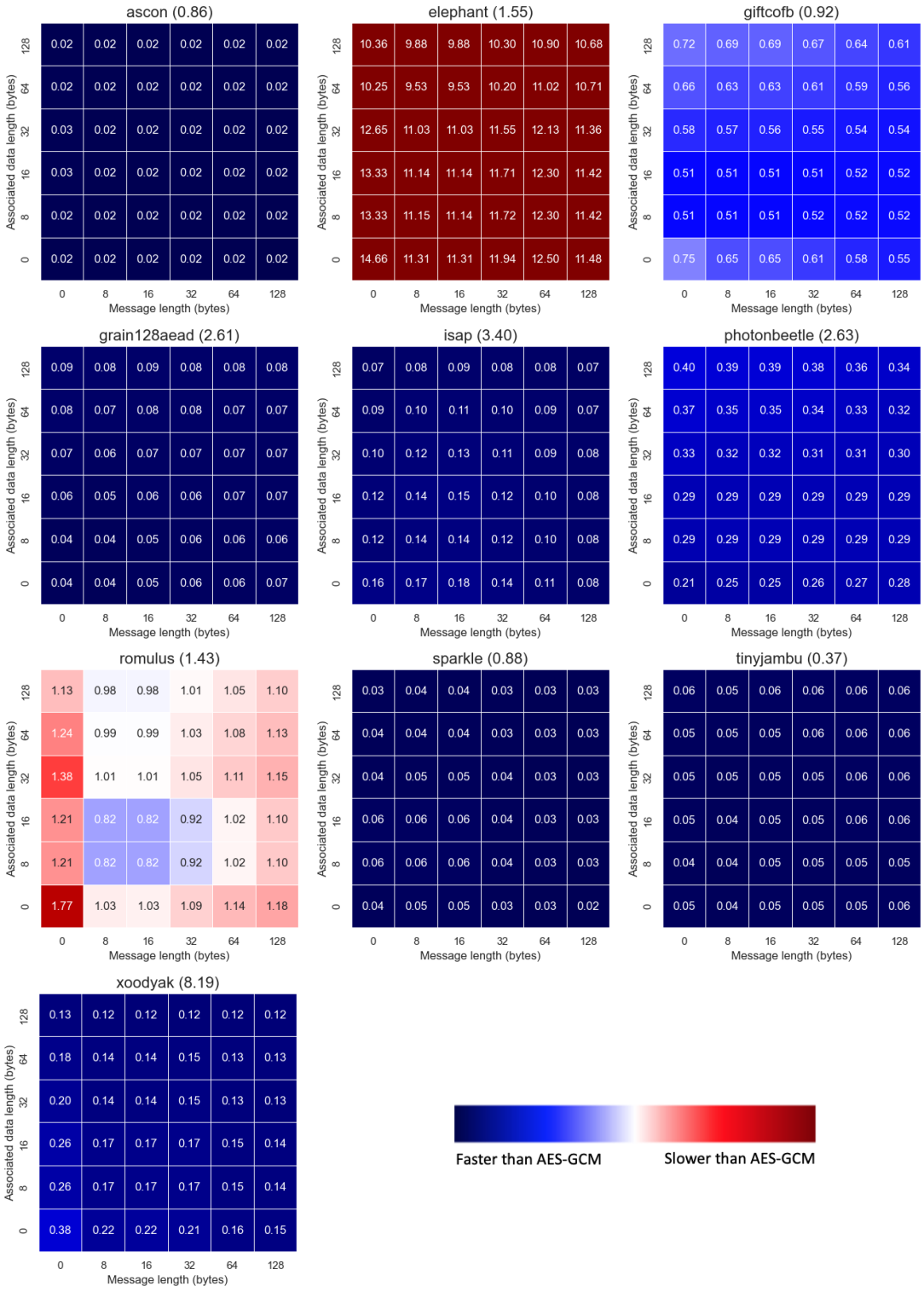


Fig. 21. Execution time ratio of smallest primary AEAD implementations to AES-GCM on AT91SAM3X8E

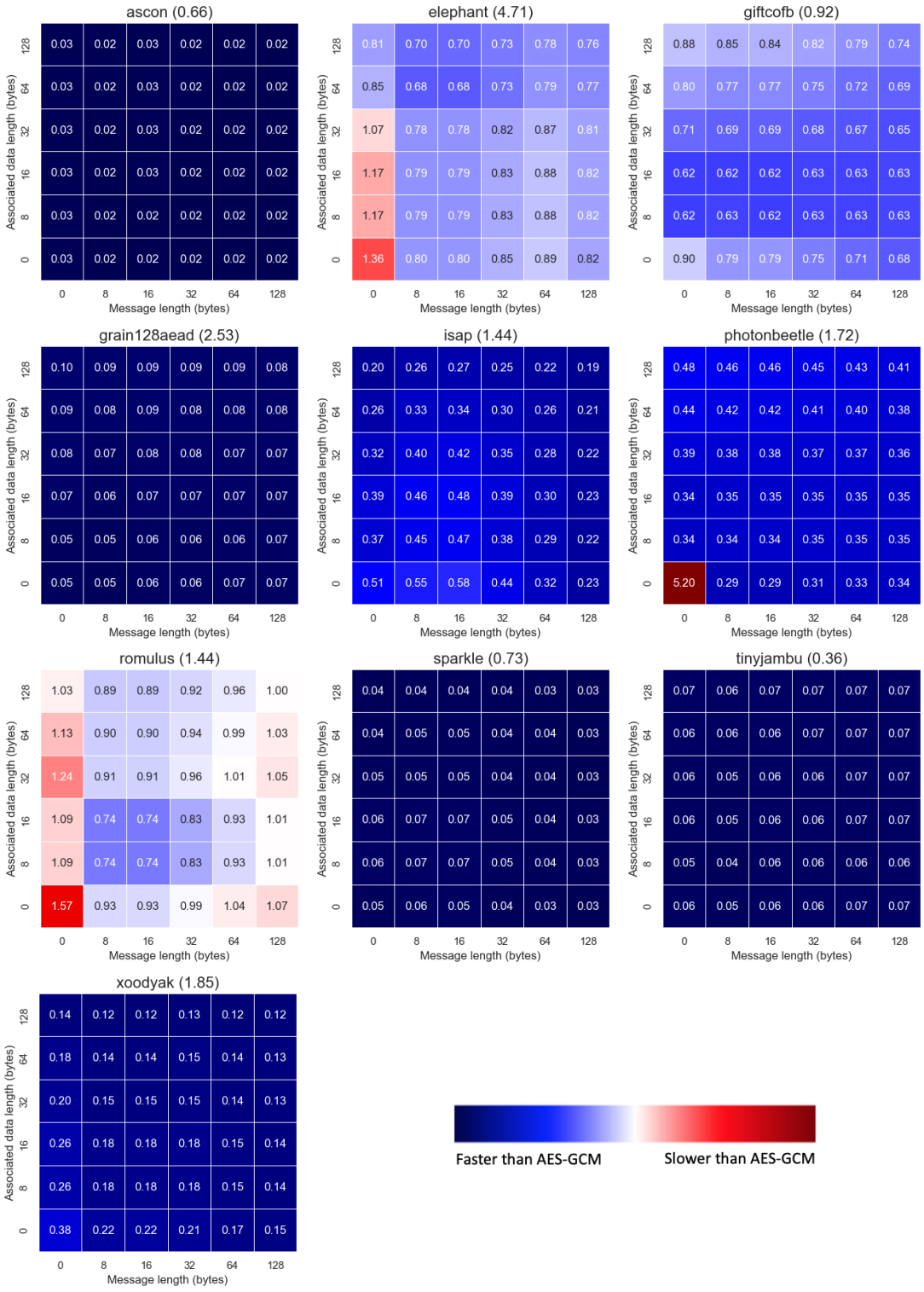


Fig. 22. Execution time ratio of smallest primary AEAD implementations to AES-GCM on nRF52840

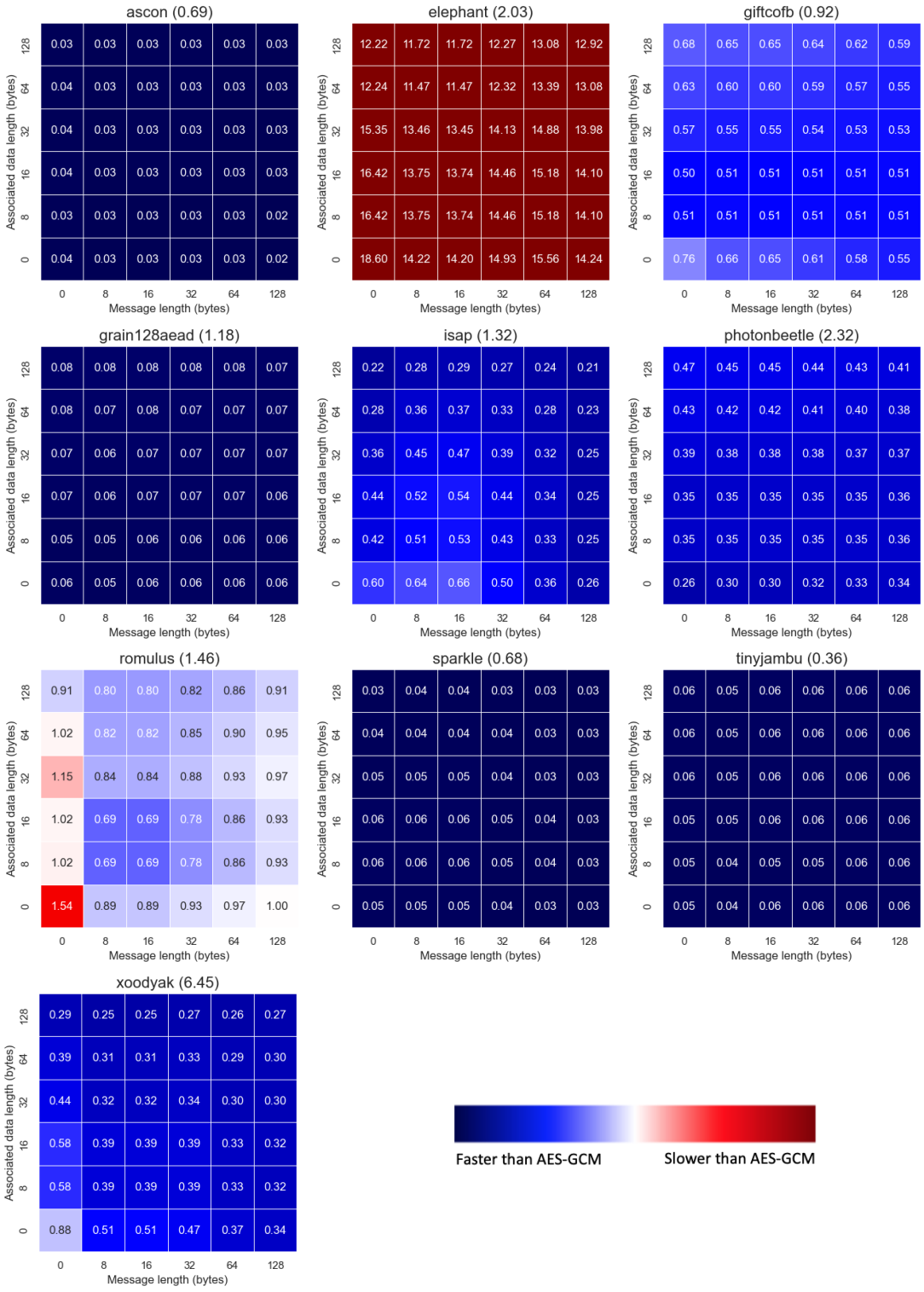


Fig. 23. Execution time ratio of smallest primary AEAD implementations to AES-GCM on PIC32MX320F128H

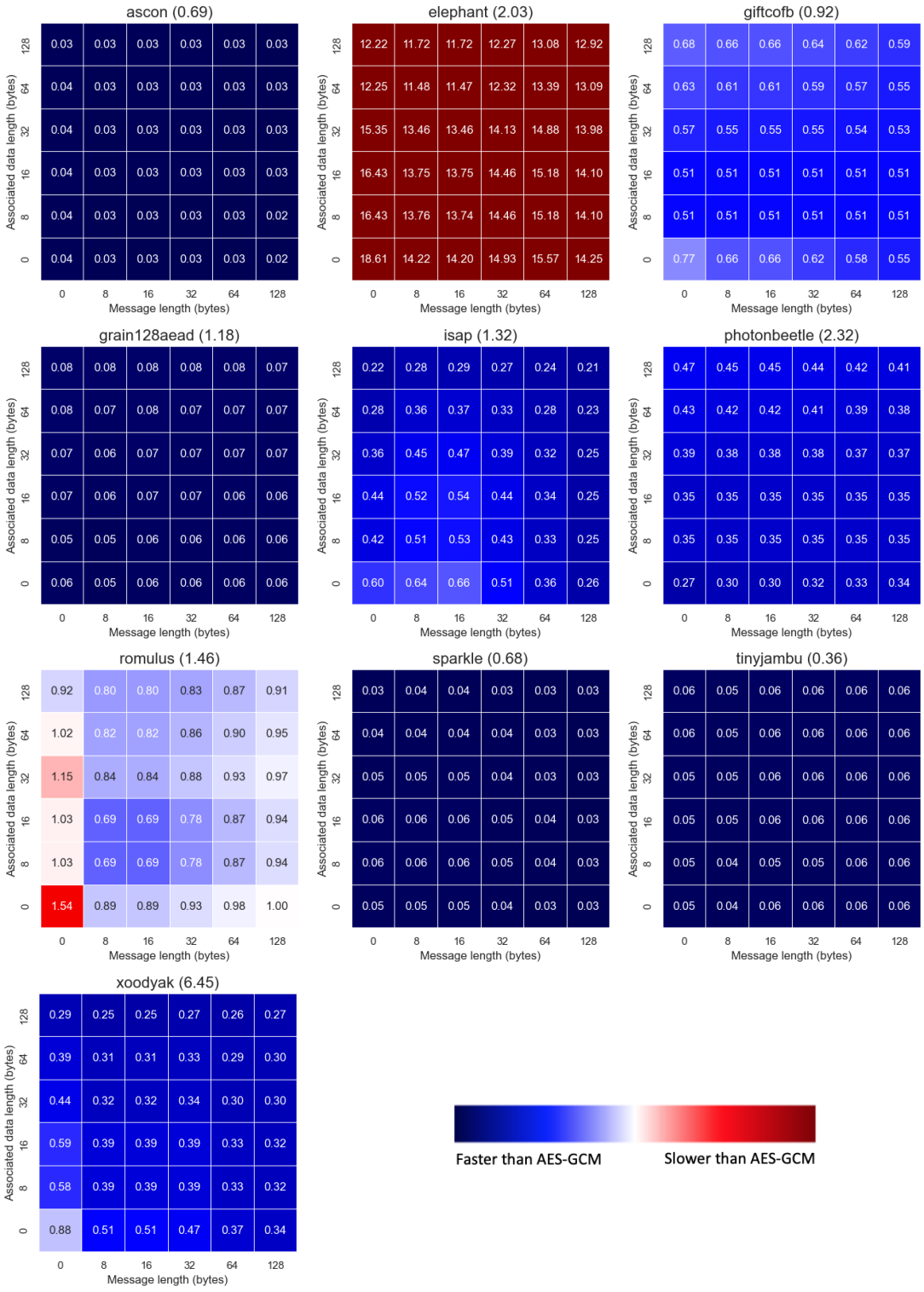


Fig. 24. Execution time ratio of smallest primary AEAD implementations to AES-GCM on PIC32MX340F512H

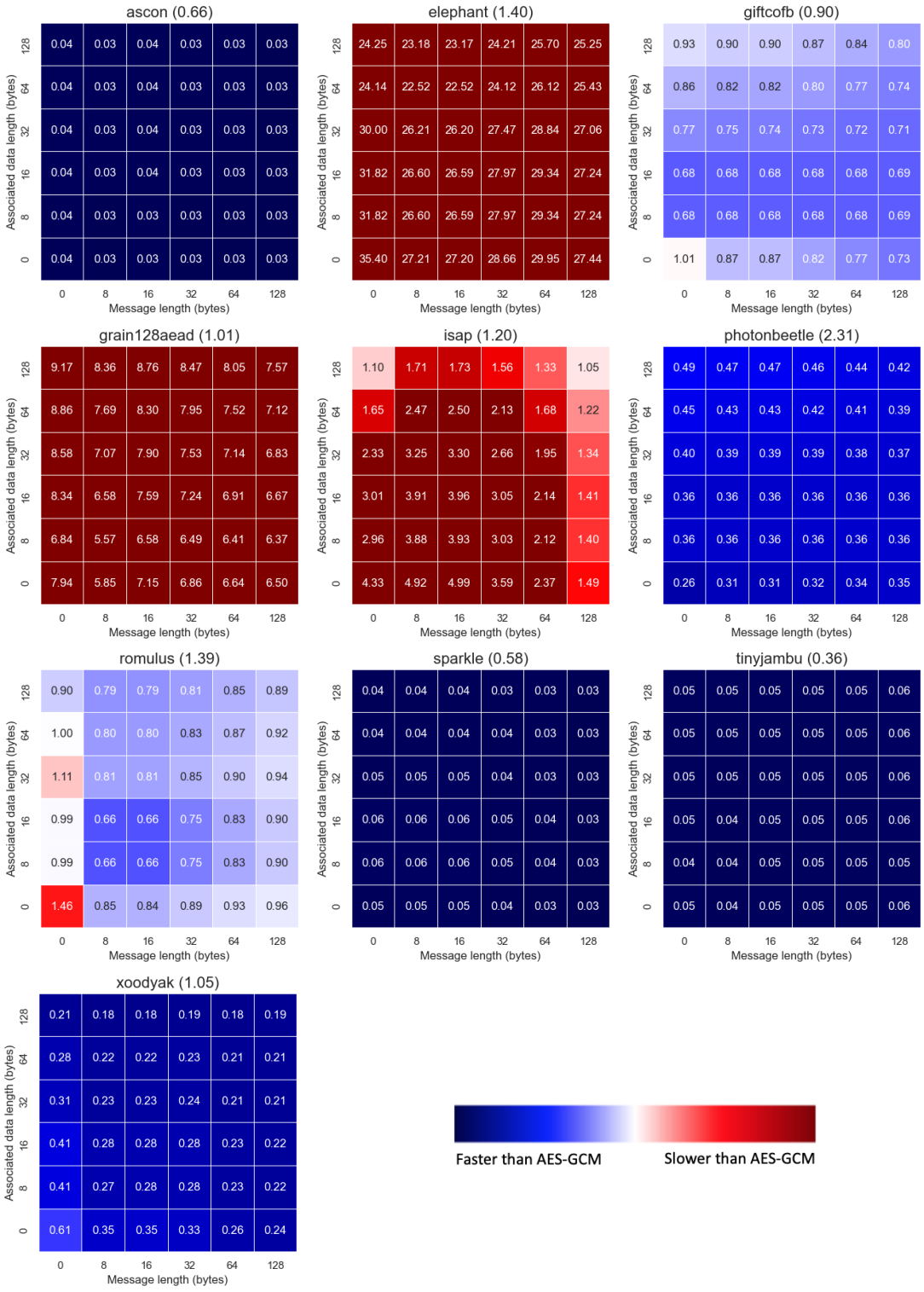


Fig. 25. Execution time ratio of smallest primary AEAD implementations to AES-GCM on ESP8266

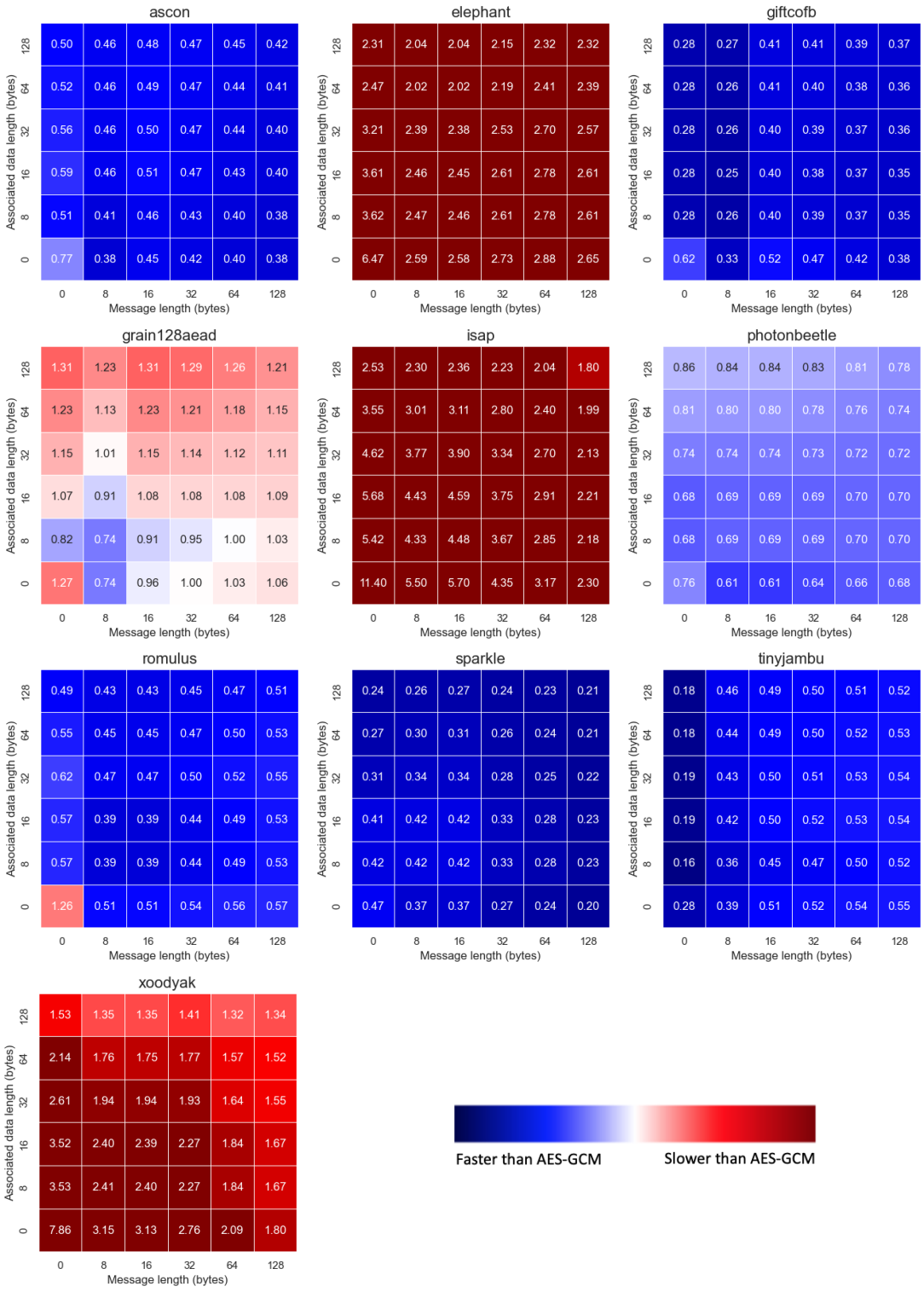


Fig. 26. Execution time ratio of fastest primary AEAD implementations to AES-GCM on ATmega4809

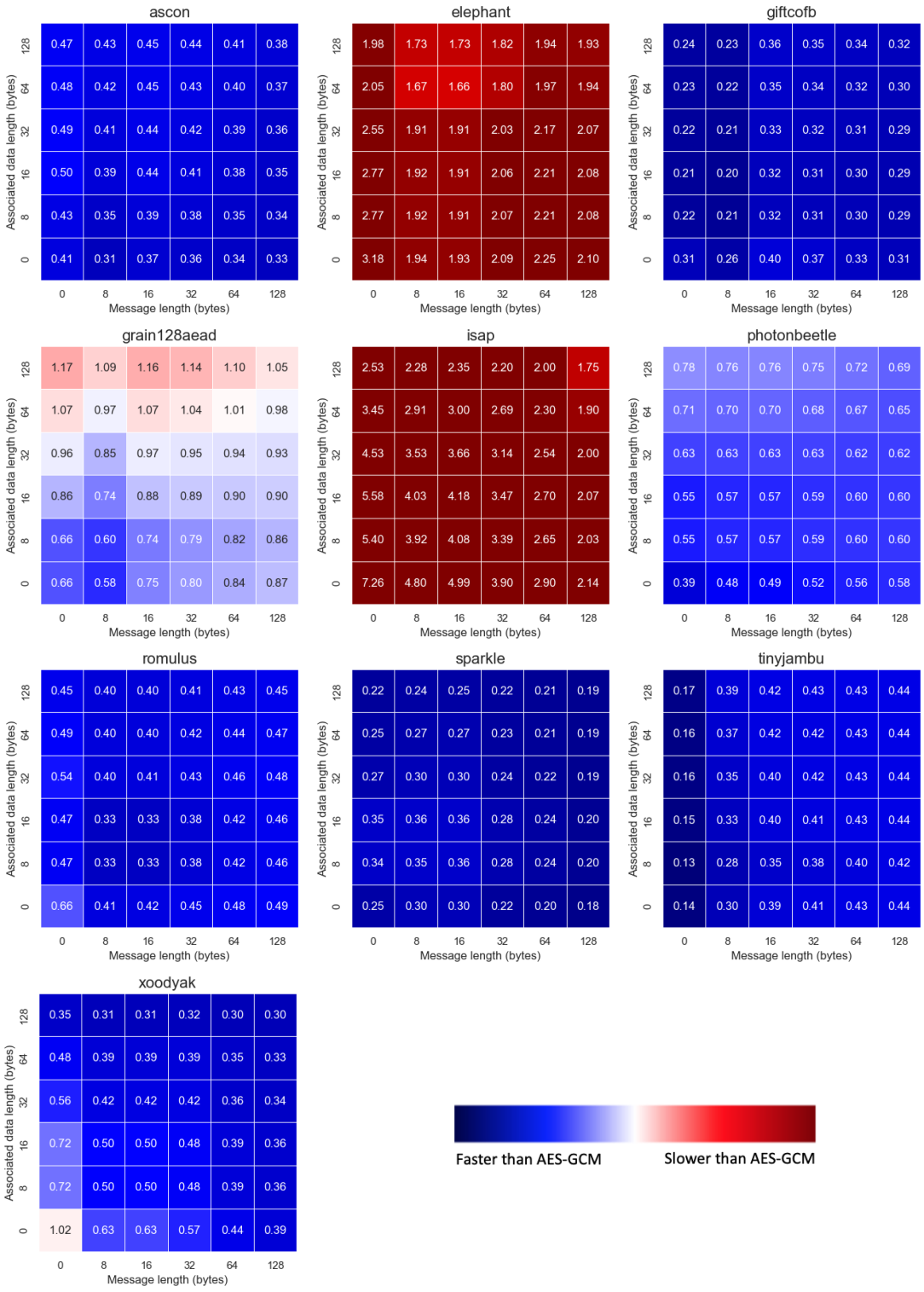


Fig. 27. Execution time ratio of fastest primary AEAD implementations to AES-GCM on ATmega328P

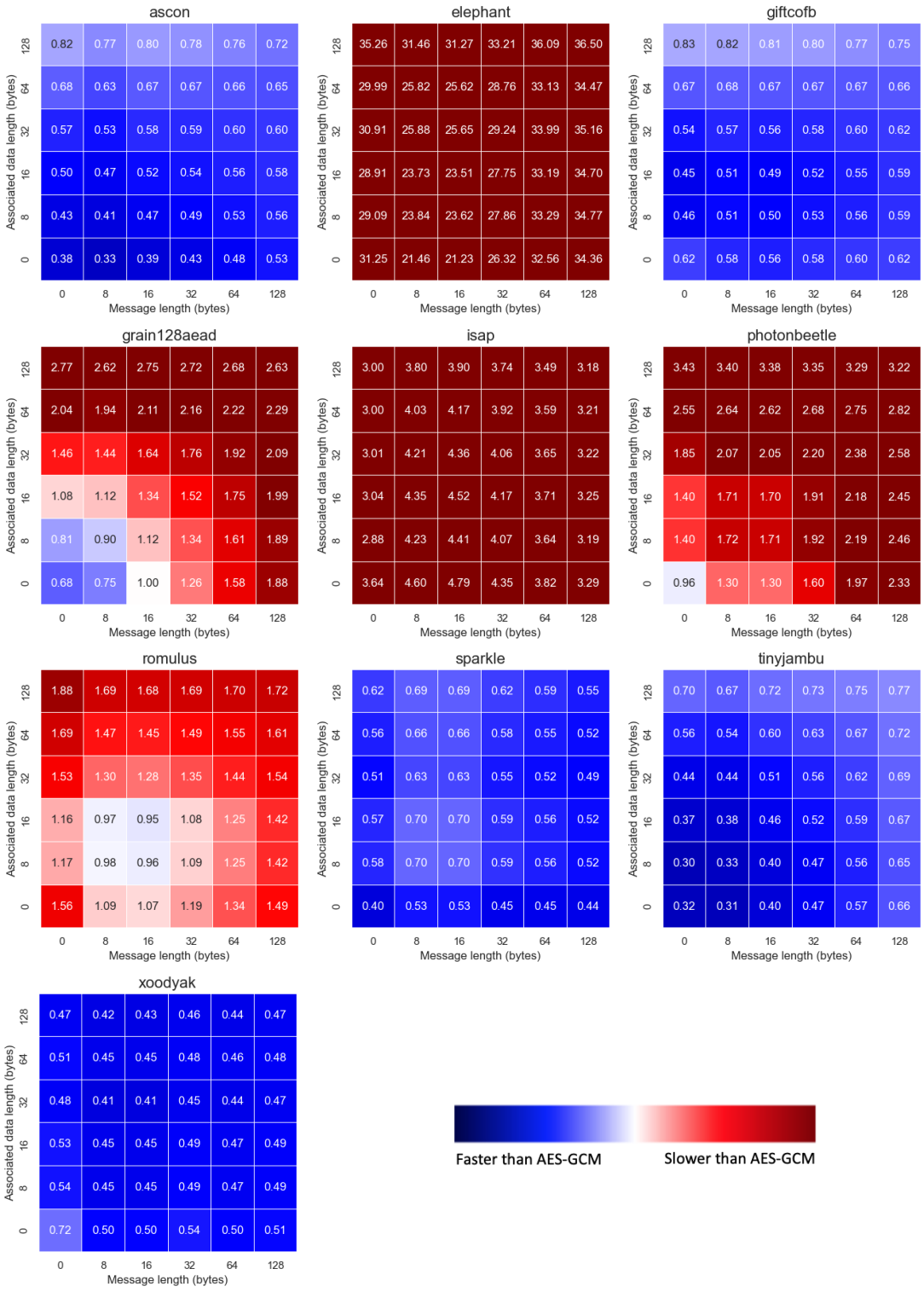


Fig. 28. Execution time ratio of fastest primary AEAD implementations to AES-GCM on SAMD21G18A

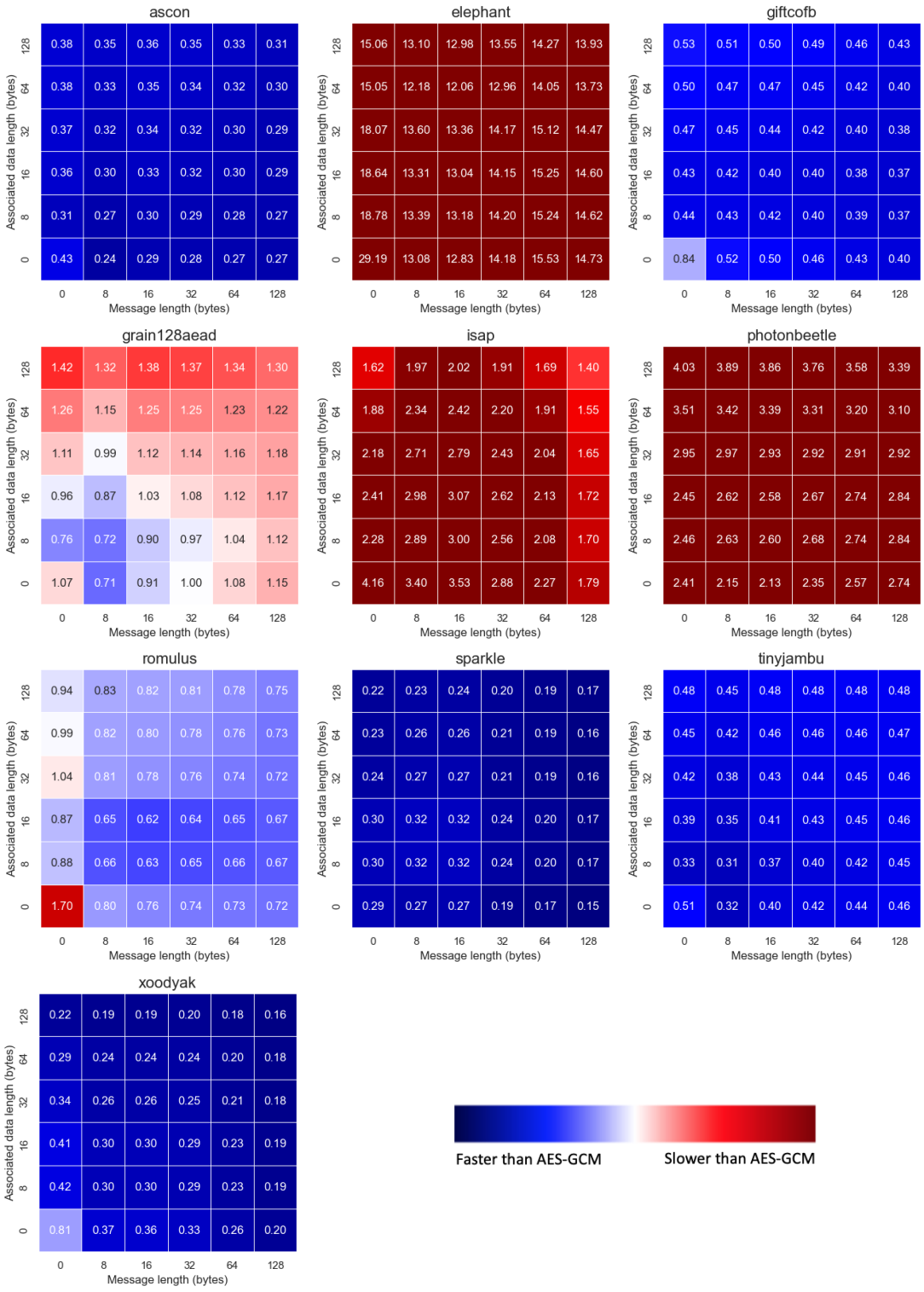


Fig. 29. Execution time ratio of fastest primary AEAD implementations to AES-GCM on AT91SAM3X8E

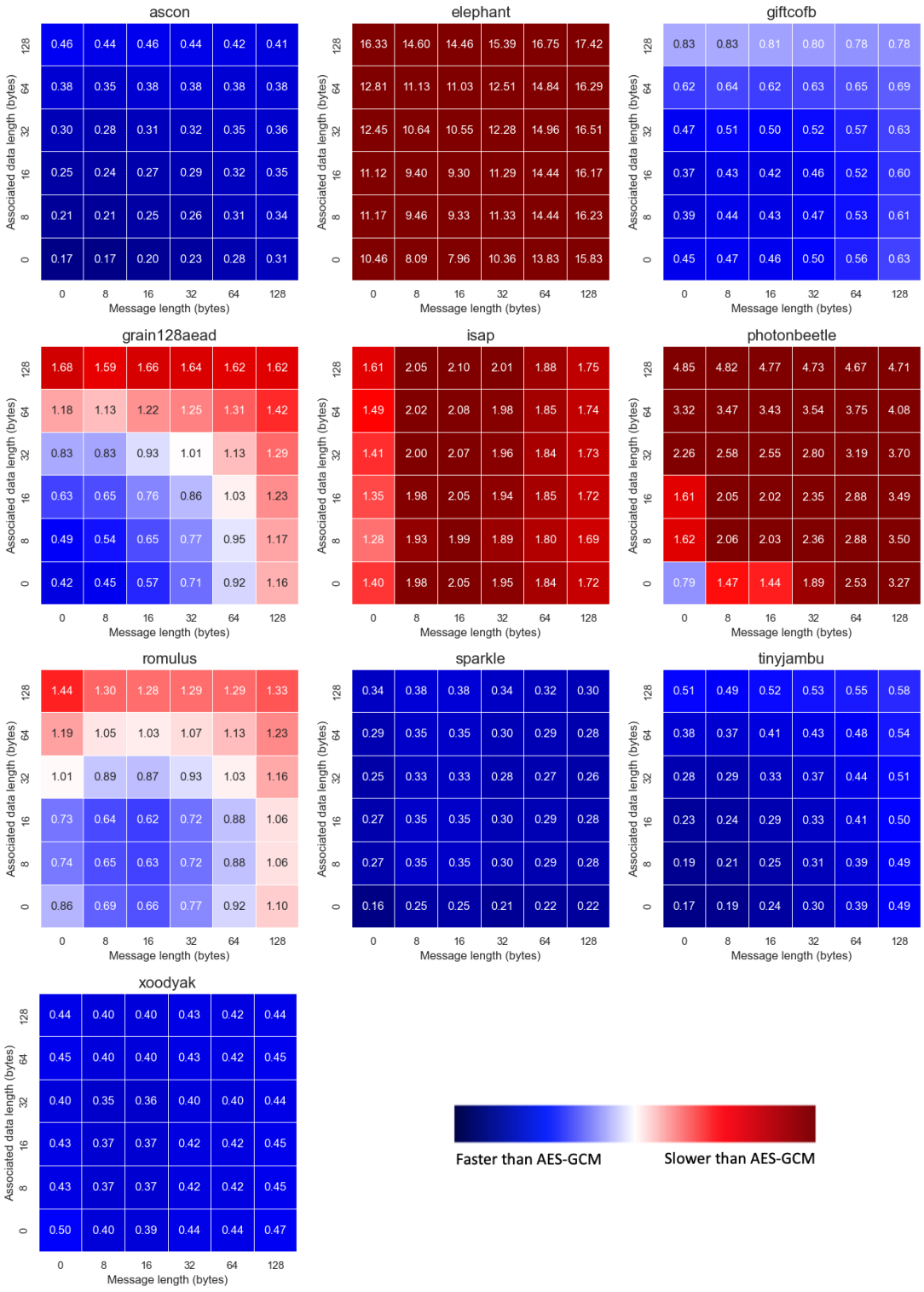


Fig. 30. Execution time ratio of fastest primary AEAD implementations to AES-GCM on nRF52840

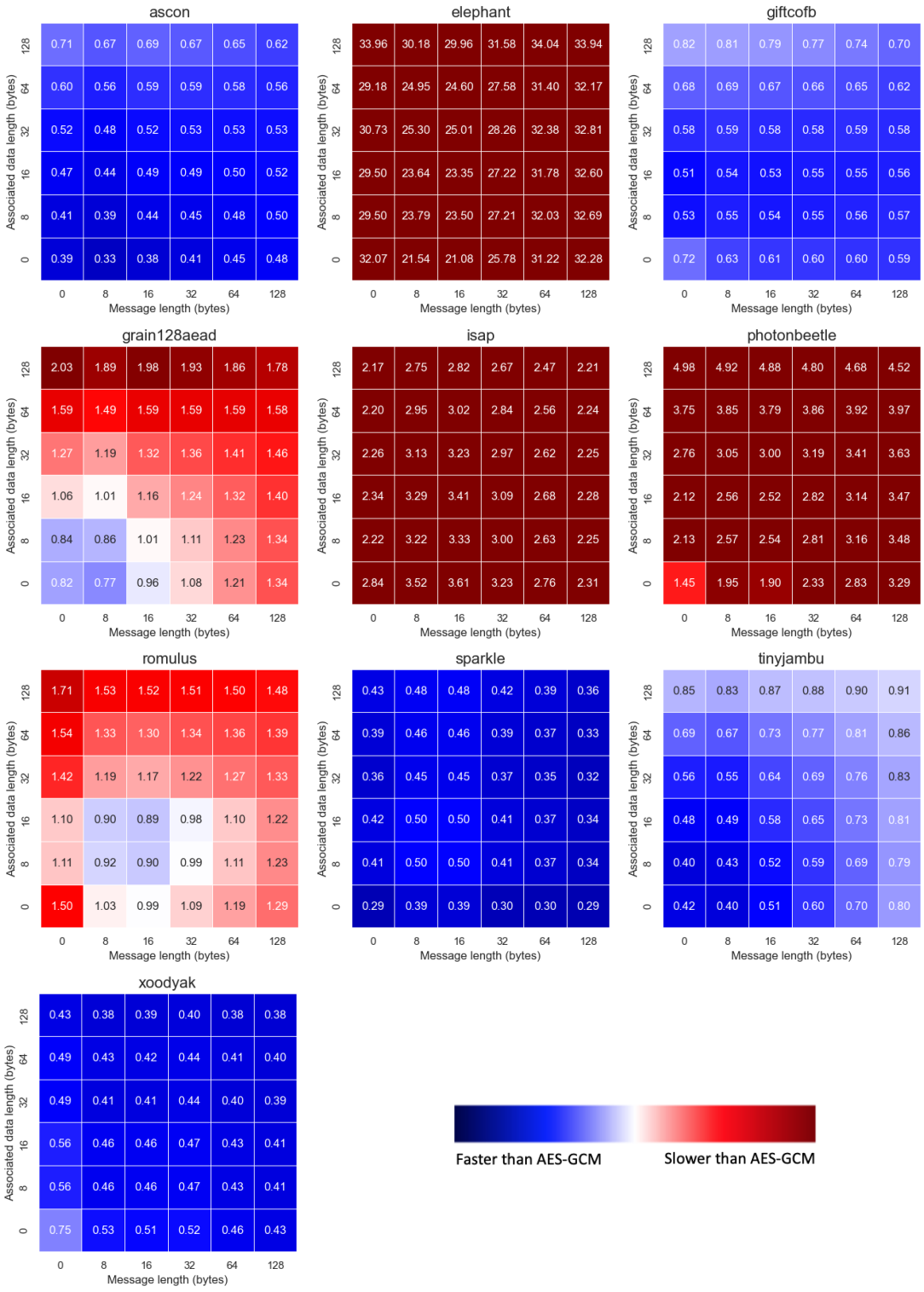


Fig. 31. Execution time ratio of fastest primary AEAD implementations to AES-GCM on PIC32MX320F128H

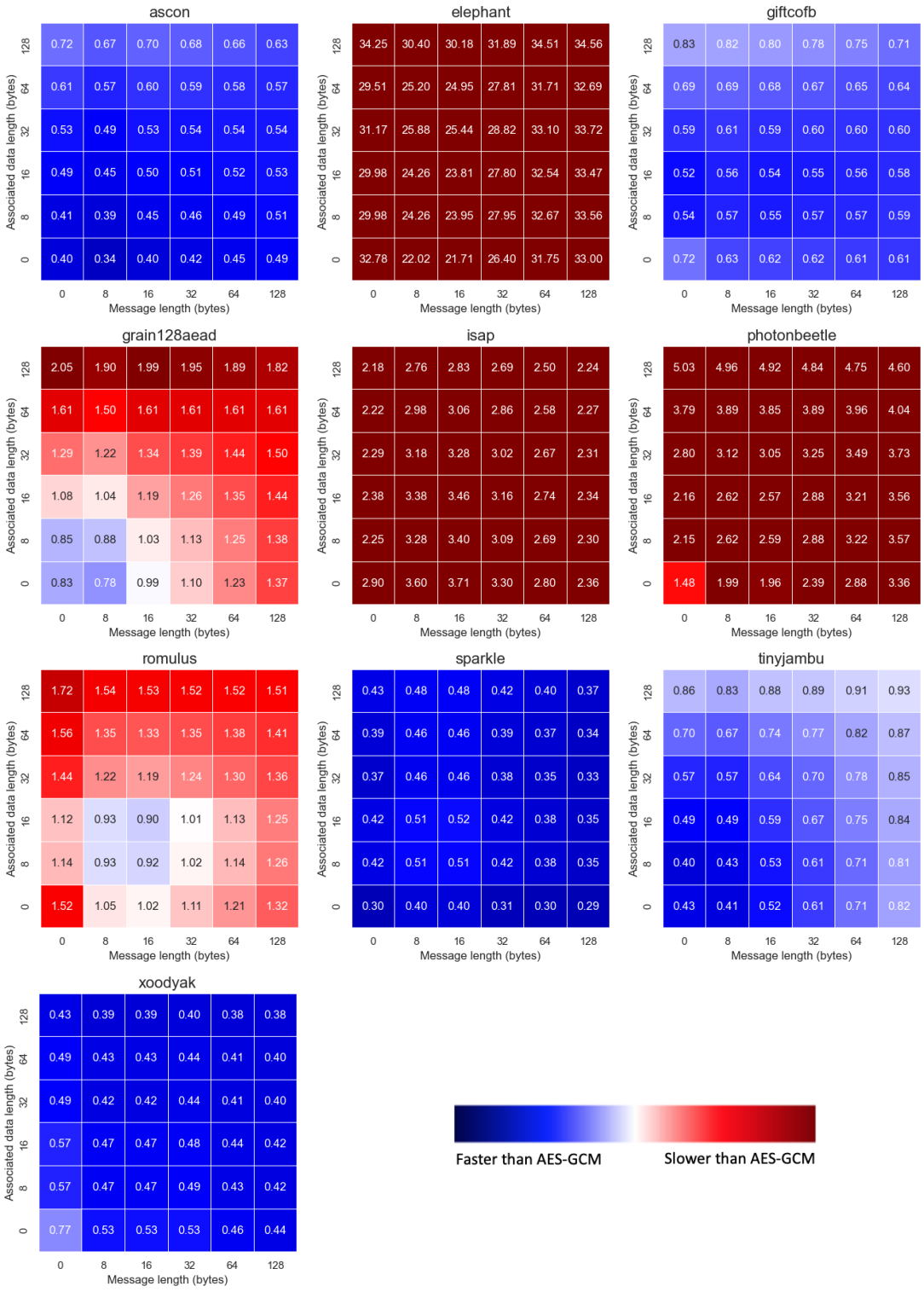


Fig. 32. Execution time ratio of fastest primary AEAD implementations to AES-GCM on PIC32MX340F512H

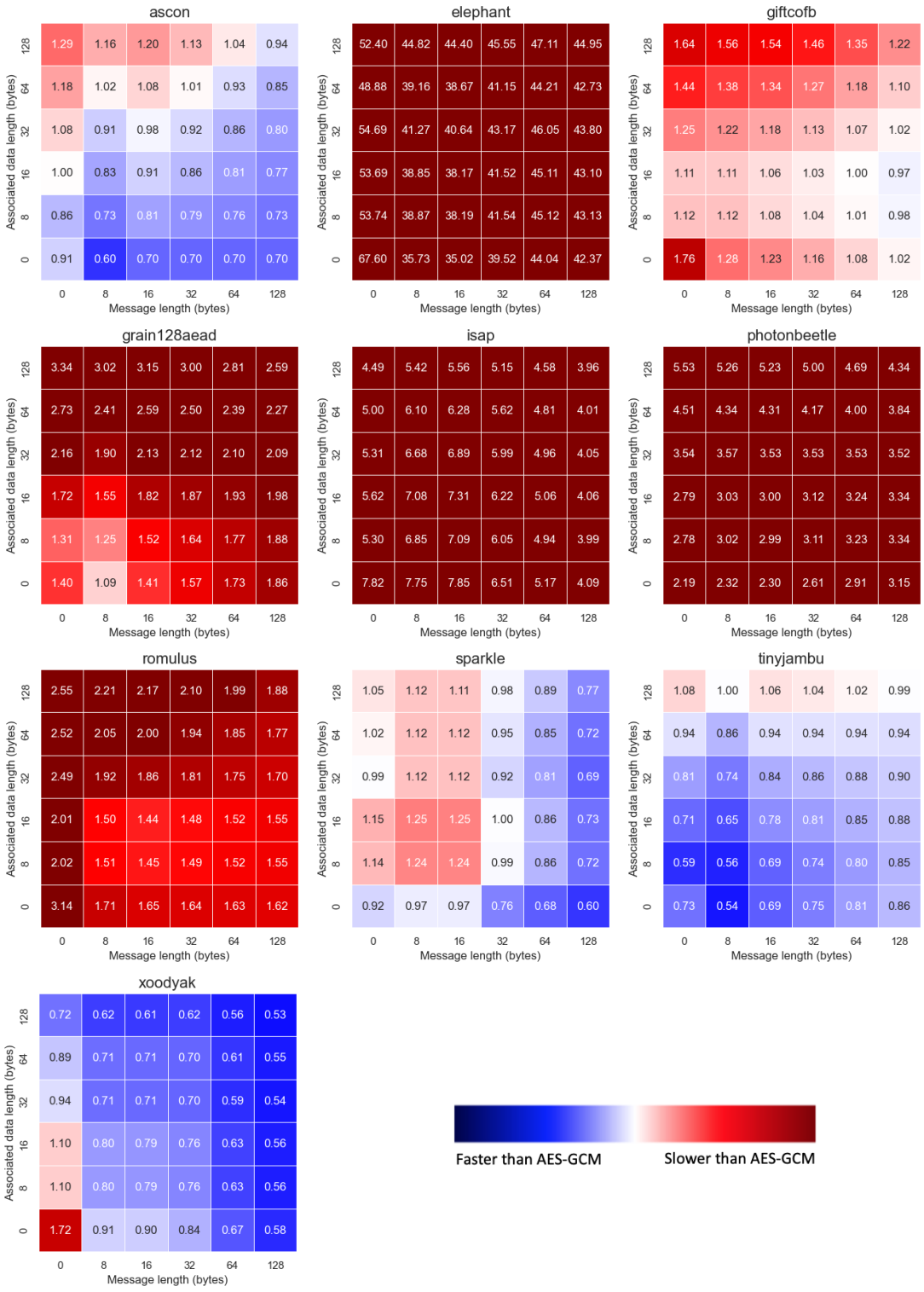


Fig. 33. Execution time ratio of fastest primary AEAD implementations to AES-GCM on ESP8266