electronics

MDPI

Review

A Review of the NIST Lightweight Cryptography Finalists and Their Fault Analyses

Hasindu Madushan [1], Iftekhar Salam [2] and Janaka Alawatugoda [3,4,*]

[1] Department of Computer Engineering, University of Peradeniya, Peradeniya 20400, Sri Lanka
[2] School of Computing and Data Science, Xiamen University Malaysia, Sepang 43900, Malaysia
[3] Research & Innovation Centers, Faculty of Resilience, Rabdan Academy, Abu Dhabi P.O. Box 114646, United Arab Emirates
[4] Institute for Integrated and Intelligent Systems, Griffith University, Nathan, QLD 4111, Australia
* Correspondence: jalawatugoda@ra.ac.ae

Abstract: The security of resource-constrained devices is critical in the IoT field, given that everything is interconnected. Therefore, the National Institute of Standards and Technology (NIST) initialized the lightweight cryptography (LWC) project to standardize the lightweight cryptography algorithms for resource-constrained devices. After two rounds, the NIST announced the finalists in 2021. The finalist algorithms are Ascon, Elephant, GIFT-COFB, Grain-128AEAD, ISAP, PHOTON-Beetle, Romulus, SPARKLE, TinyJambu, and Xoodyak. The final round of the competition is still in progress, and the NIST will select the winner based on their and third-party evaluations. In this paper, we review the 10 finalists mentioned above, discuss their constructions, and classify them according to the underlying primitives. In particular, we analyze these ciphers from different perspectives, such as cipher specifications and structures, design primitives, security parameters, advantages and disadvantages, and existing cryptanalyses. We also review existing analyses of these finalists with a specific focus on the review of fault attacks. We hope the study compiled in this paper will benefit the cryptographic community by providing an easy-to-grasp overview of the NIST LWC finalists.

Keywords: lightweight cryptography; NIST LWC competition; algorithms; IoT; fault analysis

Citation: Madushan, H.; Salam, I.; Alawatugoda, J. A Review of the NIST Lightweight Cryptography Finalists and Their Fault Analyses. *Electronics* **2022**, *11*, 4199. https://doi.org/10.3390/electronics11244199

Academic Editor: Cheng-Chi Lee

Received: 17 November 2022
Accepted: 5 December 2022
Published: 15 December 2022

1. Introduction

In the era of IoT, it is essential to have smaller, low-cost, and low-power devices for various purposes. For instance, a cardiac implant installed inside a patient's body must be small and operate for a long time without recharging or replacing the battery. These are commonly identified as resource-constrained devices, and security is essential for most of these devices. The concern is that the limited resources on these devices may cause performance issues when the standard cryptographic algorithms are running on them. Therefore, in recent years, researchers have been working on developing lightweight cryptography and various efficient cryptographic technologies [1–3]. Figure 1 illustrates the design trade-offs for lightweight cryptography. Its requirements are constrained by security, low-cost and high-performance. These requirements are balanced accordingly by adjusting the key size, the number of encryption rounds and the system architecture; key size can be lesser number of bits (e.g., 64 bits), achieving lower cost while sacrificing the level of security to some extent, or reducing the number of rounds (e.g., 16 rounds), achieving higher performance while sacrificing the level of security to some extent, or a parallelized system can be designed to achieve higher performance while incurring more cost, etc. Thus, the target of lightweight cryptography is to find a better balance between performance and security within cost costraints.

To find a better balance between performance and security for resource-constrained devices, the NIST initiated the Lightweight Cryptography (LWC) project in 2013 [4]. In the

LWC project, NIST and independent researchers study the performance of existing cryptographic standards and endeavour to develop new lightweight cryptography standards. In 2018, NIST published the submission requirements and evaluation criteria for the LWC competition [5]. After the call, 57 algorithms were submitted to the competition. The NIST eliminated one of the algorithms from these submissions as it did not fulfill the requirements. After the first round, 32 algorithms were advanced to the second round. Subsequently, the third round of the competition began in 2021 after NIST announced the finalists of the competition.

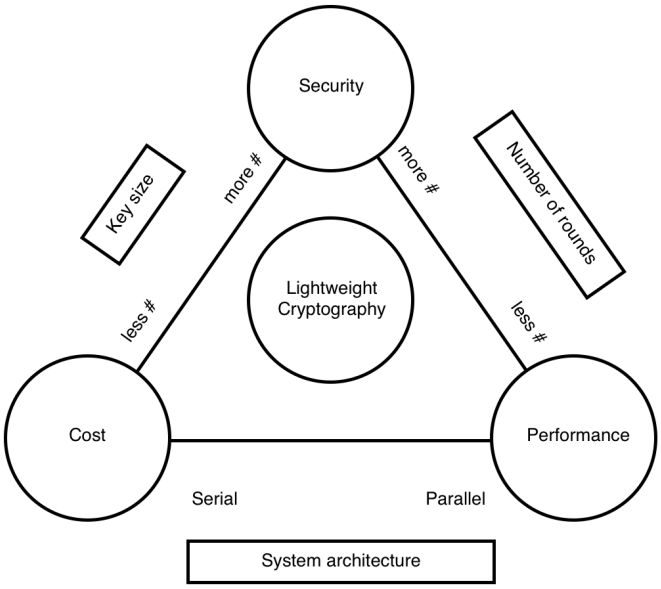

Figure 1. Lightweight cryptography: design trade-offs.

A complete specification of the algorithm and reference implementations are essential among the requirements specified by the NIST. The specification document shall contain the details of the algorithms, related mathematical operations, equations, design decisions, and a design rationale. The submission may be a family of algorithms and could introduce a hash function. Moreover, all proposals must be Authenticated Encrypted Authenticated Data (AEAD) [6] algorithms. In addition, the specification shall contain a statement of the security strength and details on known cryptographic attacks, including the complexities.

1.1. Related Works

Jimale et al. [7] conducted a systematic literature review for classifying authenticated encryption (AE) schemes. Their study comprises a broad range of AE schemes, including submissions from the CAESAR and NIST LWC competitions. They analysed the designs based on the design approaches, security properties, and functional features; however, their research did not cover third-party analyses of the AE schemes.

Elsadek et al. [8] evaluated the 10 NIST LWC finalists from the perspective of hardware design efficiency. In particular, their work analysed energy efficiency using bit/joule as one of the primary metrics. They also evaluated the throughput and area of these finalist algorithms. Their result concludes that some ciphers, for example, TinyJambu, Xoodyak and ASCON, outperform others in terms of energy efficiency. This work provides insight into some aspects of the performance of the finalist algorithms; however, along with the performance, other features such as security and cost should also be considered while making the final decision.

Pugh et al. [9] conducted a systematic review of the implementations of the NIST LWC submissions. Their study mainly focused on systematically testing the algorithms to identify potential bugs. Based on the testing framework presented in this work, they reported bugs in some early submissions.

To the best of our knowledge, there are currently no existing reviews that analyse the 10 finalists of the NIST LWC project from different perspectives, such as cipher specifications and structures, design primitives, security parameters, advantages and disadvantages, and existing cryptanalyses. We note that in the previous cryptographic competitions, for example, CAESAR competition [10], similar reviews have been conducted, which present an overview of different ciphers from the CAESAR competition. We believe similar analyses presented in this paper will help the cryptographic community to better understand the current status of the finalist algorithms.

1.2. Our Contribution

We conducted a detailed literature review of the 10 finalists. Our study focuses on finalist algorithms' specifications and their structures, design primitives, security parameters, advantages, and disadvantages. Our work also summarizes the existing cryptanalyses of these ciphers. Apart from summarizing the existing cryptanalyses, we specifically focused on reviewing the current and potential fault analyses of the NIST LWC finalists. The lightweight ciphers are primarily used in resource-constrained IoT devices. These devices' environments may not be physically secure, allowing attackers to access them physically and perform side-channel attacks, such as fault attacks. The fault attacks are highly effective against lightweight ciphers compared to classical cryptanalysis. The nature of the physical environment in which these ciphers operate makes them more vulnerable to such attacks. Therefore, it is crucial to evaluate the security of such ciphers against fault attacks. This motivates us to focus our review of security analyses towards the direction of fault attacks. Lastly, we have identified several open research problems that warrant further investigations.

1.3. Organization of the Paper

Section 2 presents the cryptographic primitives and constructions that are used in the 10 NIST LWC finalist algorithms. The discussion of these underlying primitives and constructions will be helpful to understand Section 3, where the specifications of all the finalist algorithms are presented. The section starts with a summary of a comparison of the major aspects of the algorithms. Then, a description of each one of the algorithms is provided within each subsection. Section 3 also contains a summary of the previously known cryptanalyses on the finalist algorithms. As mentioned earlier, we are only focusing on the fault attacks in this paper. Section 4 contains detailed descriptions of the fault attacks that might be applicable or already applied to lightweight cryptographic algorithms. Section 5 contains the conclusion of the paper.

2. Constructions and Primitives of Lightweight Cryptographic Algorithms

In this section, we review different design constructions, such as sponge, duplex, and photon. Furthermore, we also discuss different cryptographic primitives used in constructing the NIST LWC finalists.

2.1. Sponge

The Sponge construction proposed by Bertoni et al. [11] is a recent work and one of the most commonly used primitives among the finalists of the NIST LWC competition. Although its main purpose is to construct secure hash functions, certain properties make the Sponge construction suitable for authenticated encryption, especially lightweight ones. It is an iterative construction that acts akin to a random oracle but might suffer from the inner state collision. The structure of the Sponge function is simple and efficient, which makes it attractive for lightweight cryptography. The function takes a variable-length input and produces an infinite-length output. It consists of a fixed-size state and two main procedures; absorbing and squeezing. The state is divided into two parts; one is kept secret, while the other is used to produce the output using the input. Each iteration of the Sponge

function absorbs a fixed-length portion of the input, performs a transformation on the entire state, and produces the output in the squeezing phase.

As the designers claim, when the Sponge function is used as a hash function, the complexity to produce an inner collision is $2^{(c+3)/2}$ (c is the size of the inner state or the capacity). The complexity of an output collision is $2^{(c+3)/2}$ where n is the output length or the digest length. Bertoni et al. further claim that a sponge-based hash function is secure against pre-image and second pre-image attacks. They also defined the use of Sponge construction for authenticated encryption functions and stream ciphers. This is achieved by storing the secret value (the key) in the inner state since it will not be exposed to the output directly. The secrecy of the key will increase as the capacity increases. Sponge construction-based algorithms, such as SPONGET, KECCAK, and PHOTON, are the underlying primitives for several of the NIST LWC finalists.

2.1.1. SPONGENT

SPONGENT [12] is a permutation function based on the Sponge construction. It is used as a cryptographic primitive in Elephant [13] and a few other LWC algorithms. Moreover, the permutation used in SPONGENT is related to PRESENT [14]. It has 13 variants, including SPONGENT-128/256/128, SPONGENT-160/320/160, SPONGENT-224/448/224, and SPONGENT-88/80/8. The permutation consists of ICounter that is updated using a linear feedback shift register (LFSR), a substitution layer with a 4-bit S-box, and a permutation layer.

The designers Andrey et al. [12] claim that the design of the SPONGENT prevents the linear hull effect, which is the main weakness of the PRESENT cipher. Furthermore, it has an upper bound of the differential characteristic probability of 2^{-28} over six rounds. Another vital claim in their work is that it provides resistance against differential cryptanalysis. Moreover, the SPONGENT provides sound security against collision and pre-image attacks when used as a hash function. Further, it also provides security against linear attacks. Finally, the SPOGENT can be easily implemented in hardware using a minimum gate area as small as 738 GE for SPONGENT-88/80/8 variant.

2.1.2. KECCAK

KECCAK [15] is a family of Sponge-based cryptographic algorithms that are used in ISAP [16], Elephant [13], and many other lightweight cryptographic algorithms. Its design is simple and flexible, and it has many variations. Moreover, KECCAK-f permutation is bit-oriented and parallelizable on hardware that supports SIMD and CPU pipelining. The permutation used in KECCAK is known as KECCAK-f and has seven variations. These variations have state sizes varying from 25 to 1600 bits and have different numbers of rounds depending on the state size. The permutation consists of a few different mapping. As the designer indicated, these mappings provide differential propagation properties, correlation properties, and some other properties to the permutation that make it secure against many differentials and linear cryptanalyses, such as higher-order differential attacks, impossible differential attacks, differential-linear attacks, boomerang attacks, and integral attacks. Furthermore, the asymmetric properties of different rounds prevent slide attacks.

2.1.3. PHOTON

PHOTON [17] is another family of lightweight cryptographic hash functions designed for resource-constrained devices. Similar to SPONGENT and KECCAK, it also has a Sponge-based construction. Although it is introduced as a hardware-oriented design, it can also be easily implemented in software. It is used as the permutation function in the NIST LWC finalist algorithm PHOTON-Beetle [18] and other lightweight cryptographic schemes such as ORANGE [19]. It is a new version of the Sponge function, called the extended Sponge framework. It uses a different capacity in the squeezing phase than in the absorbing phase. The advantage of this construction is that it provides better pre-image resistance depending on the new capacity size used. The PHOTON uses a permutation similar to AES

to update the state. However, the implementation of the permutation requires much less gate area compared to the most lightweight implementation of AES permutation. Several variants of the PHOTON hash function were proposed based on the output size (from 64 to 256 bits).

The designers claim that the security of the permutation can be easily proven because of its AES-like structure, especially resistance against differential and linear cryptanalyses. The rebound and super-box attacks are recent and effective cryptanalysis methods on AES-like hash functions. According to the security analyses of these two types of attacks, the data and time complexities are very high for PHOTON. Moreover, the PHOTON provides resistance against cube testers and algebraic attacks.

2.2. Duplex

Duplex mode [20] is a cryptographic construction similar to the Sponge permutation. It is designed to use the Sponge construction to generate a message digest (MAC) more easily than the Sponge-based constructions. Moreover, it is one of the first complete authenticated encryption constructions that can be used for permutation-based ciphers without a key scheduling algorithm. Apart from this, it has some special use cases, such as pseudo-random bit generation. It is used to construct well-known lightweight cryptographic algorithms such as Ascon. The Duplex function takes three inputs: plaintext, a data header (or associated data), a key to produce the ciphertext and a tag as the message authentication code (MAC). This procedure is known as wrapping (SpongeWrap). The decryption process (known as unwrapping) expects the key used for wrapping, the ciphertext, associated data, and the tag as the input. Then, it produces the plaintext if the tag is correct or an error if the tag is invalid. It is easy to see that these properties correspond with the requirements of the NIST lightweight cryptographic competition, which makes it a suitable choice for lightweight AEAD algorithms. The duplex construction is based on Sponge construction. However, unlike the sponge construction, it can use the same state used for the previous encryption call. In other words, it maintains the state between encryption (or decryption) calls [20]. The authors call this "duplexing". Further, the instance of this construction is called the "duplex object".

It is shown that security against generic attacks can be easily proven. In addition, the designers also state that it provides the flexibility to choose the bit rate and the padding function. Moreover, a duplex construction allows a simple permutation as the transition function. However, it also has some disadvantages. Duplex-based encryption cannot be executed in parallel since it requires the state from the previous call. Another downside is that it does not support nonce naturally, which results in encrypting the same message sequence multiple times to produce the same output.

2.3. Tweakable Block Cipher

Many cryptographic algorithms have modes of operation that produce different outputs for the same inputs each time. However, the cryptographic primitives are not designed for such variation. Tweakable block cipher is a well-known technique proposed by Moses et al. [21] to construct a block cipher primitive with variability in the output. According to Moses et al., the main advantage of using such a cryptographic primitive is efficiency. The main difference between a regular block cipher and a tweakable block cipher is that a tweakable block cipher takes an extra parameter known as the tweak for an input. The tweak is a value that provides variations in the output for the same inputs.

A regular block cipher can be used to produce a secure and effective tweakable block cipher. However, constructing a tweakable block cipher from a standard block cipher requires careful design since a weak construction may worsen the security of the original block cipher. Moses et al. described a few methods to construct a tweakable block cipher from a regular one. For instance, XOR-ing the output of a block cipher with the tweak and encrypting again is a way to construct a secure cipher. The designers also suggest a few modes of operations for tweakable block ciphers, namely Tweak Block Chaining (TBC),

Tweak Chain Hash (TCH), and Tweak Authenticated Encryption (TAE). The NIST LWC finalist Romulus uses the tweakable block cipher Skinny as its underlying primitive (refer to the section detailing Romulus for more details).

There are a few research opportunities related to tweakable block ciphers. For instance, analyzing the security of the modes of operations associated with the tweakable block cipher and defining and analyzing a tweakable stream cipher are open problems.

2.4. Grain

Grain is a lightweight stream cipher first introduced by Hell et al. [22]. The idea of Grain stream cipher is used to construct the finalist Grain128-AEAD [23]. It was designed for hardware applications where a high encryption/decryption speed is required using a minimum gate count. It is efficient in software as well. The design of Grain consists of both linear and non-linear feedback registers.

3. Specifications of NIST LWC Finalist Algorithms

The 10 finalist algorithms of the NIST LWC project are Ascon, Elephant, GIFT-COFB, Grain-128AEAD, ISAP, PHOTON-Beetle, Romulus, SPARKLE, TinyJambu, and Xoodyak. We briefly review the specifications and constructions of these ciphers. Table 1 provides a comparison between the NIST LWC finalist algorithms in terms of the underlying primitive, mode of operation, sizes of the state, the key and the tag. As shown in Table 1, the majority (seven) of the finalists are sponge permutation-based constructions, whereas the finalists include two block cipher-based and one stream cipher-based constructions. More details on their constructions are followed in the below subsections.

3.1. Ascon

Ascon [24] is a block cipher encryption with two variations; Ascon-128 and Ascon-128a. It is also one of the algorithms in the final portfolio of the CAESAR competition [24]. The Ascon team has developed a new variation called Ascon-80pq, which is secure against quantum key search. The submission also consists of two hash functions, namely Ascon-HASH and Ascon-HASHA. The Ascon is based on a 320-bit permutation structure and is designed to perform well in hardware and software. The Ascon-128 and the Ascon-128a have a 128-bit key, 128-bit nonce, and 128-bit tag. The data block sizes of the Ascon-128 and the Ascon-128a are 64 bits and 128 bits, respectively. All these parameters are fixed. The initial value of the state consists of the key and nonce. The Ascon is based on a duplex mode of operation, more specifically, the MonkeyDuplex. The encryption and the decryption algorithms have a finalization phase to produce the Message Authentication Code (MAC).

The specification of Ascon states a few security claims. All variations of the Ascon provide 128-bit security. The designers claim that it is secure even when the nonce is reused a few times accidentally. They indicated an attacker could not recover the key, even if the internal state is known for a few rounds and the complexity of a key recovery attack is 2^{96} for the Ascon-128a and 2^{128} for the Ascon-128. The authors state the special variation, Ascon-80pq, has resistance against Grover's algorithm-based key search attacks due to the increased key size. The hash functions of Ascon have security against collision attacks and pre-image attacks and also provide resistance against length extension attacks and second pre-image attacks. Furthermore, Ascon provides robust security against timing attacks due to its bit-sliced S-boxes. Ascon also contains details about the expected performance; as the authors claim, Ascon has better performance since its operations are based on 64-bit words and only use bit-wise operations. They have measured the throughput as 4.9–7.3 Gbps on less than 10 kGE hardware. Additionally, Ascon does not need any inverse operation, which also makes it efficient.

Table 1. A summary of the specifications of the NIST LWC finalist algorithms

Name	Type	Variant	Underlying Primitive	State (Bits)	Key (Bits)	Mode of Operation	Rate/Block (Bits)	Tag (Bits)	Security (Bits)
Ascon	Sponge	Ascon-128	Ascon-p	320	128	Duplex	64	128	128
		Ascon-128a	Ascon-p	320	128	Duplex	128	128	128
Elephant	Sponge	Dumbo	Spongent	160	128	Elephant	160	64	112
		Jumbo	Spongent	176	128	Elephant	176	64	127
		Delirium	Keccak	200	128	Elephant	176	128	127
GIFT-COFB	Block	GIFT-COFB	GIFT-128	192	128	COFB	128	128	128
Grain-128AEAD	Stream	Grain-128AEAD	N/A	256	128	N/A	1	64	128
ISAP	Sponge	ISAP-K-128	Keccak	400	128	ISAP	144	128	128
		ISAP-A-128	Ascon-p	320	128	ISAP	64	128	128
		ISAP-K-128A	Keccak	400	128	ISAP	144	128	128
		ISAP-A-128A	Ascon-p	320	128	ISAP	64	128	128
PHOTON-Beetle	Sponge	PHOTON-Beetle-AEAD[128]	PHOTON256	256	128	Beetle	128	256	121
		PHOTON-Beetle-AEAD[32]	PHOTON256	256	128	Beetle	32	256	128
Romulus	Block	Romulus-N	Skinny-128-384	384	128	COFB	128	128	128
		Romulus-M	Skinny-128-384	384	128	COFB	128	128	128
		Romulus-T	Skinny-128-384	384	128	COFB	128	128	128
SPARKLE	Sponge	SCHWAEMM128-128	SPARKLE	256	128	SPARKLE	128	128	120
		SCHWAEMM256-128	SPARKLE	384	128	SPARKLE	256	128	120
		SCHWAEMM192-192	SPARKLE	384	192	SPARKLE	192	192	184
		SCHWAEMM256-256	SPARKLE	512	256	SPARKLE	256	256	248
TinyJambu	Sponge	TinyJambu	TinyJambu	128	128	TinyJambu	32	64	120
Xoodyak	Sponge	Xoodyak	Xoodoo	384	128	Cyclist	352	128	128

Besides the security and performance claims, the Ascon specification also contains a design rationale that provides great details about the design decisions. For instance, Ascon's Sponge-based structure gives several advantages, such as proven security, flexibility, and simplicity. Moreover, the specification provides a guide on how to select the correct member of the family. The authors have provided a summary of known attacks performed by third parties and their own analyses. The best-known analyses listed are zero-sum attack, integral attack, differential attack, linear attack, zero-correlation attack, impossible differential attack, and subspace trail attack. When the numbers of rounds are is 12, the complexity of these attacks is much higher than 2^{128}, which is acceptable in resource-constrained cases. Other than the attacks, it also analyses differential, linear and algebraic properties.

Nevertheless, there is a clear gap in practical side-channel attacks tested on the Ascon. Therefore, it is a direction for new research on side-channel attacks on the Ascon. Finally, the Ascon contains an implementation section that contains performance-related measurements such as cycles per byte. However, it does not include performance characteristics on low-end devices such as 8-bit microcontrollers.

3.2. Elephant

The second algorithm on the LWC finalist list is Elephant [13]. It is also a block cipher based on permutations. It uses the encrypt-then-MAC procedure for authentication. The Elephant's inverse-free design is parallelizable in software and hardware. It has three variations, namely, Dumbo, Jumbo, and Delirium. The main variation, Dumbo and Jumbo, are based on Spongent [12] hashing, while Delirium uses Keccak [25] as its primitive. All three variants use LFSR for masking. Furthermore, Dumbo, Jumbo, and Delirium have block sizes of 160-bits, 176-bits, and 200-bits, respectively. The main masking used in authenticated encryption consists of an LFSR and a permutation (Spongent or Keccak). This will be XORed with the plaintext to produce the ciphertext in the encryption, and the ciphertext will be XORed with the mask to recover the plaintext. The Elephant presents specifications for the permutations for all three variations – the tag size for Dumbo and Jumbo are 64 bits, and for Delirium, it is 128.

The Elephant contains detailed analyses of formal multi-user security of the authenticated encryption mode. The authors have derived an upper bound for the advantages of the adversary in breaking the security. The authors claim that Dumbo, Jumbo, and Delirium versions achieve 112-bit, 127-bit, and 127-bit security, respectively. The security analysis section of the Elephant contains a theoretical analysis of differential, linear, and integral cryptanalysis on Spongent and Keccak permutations.

However, it needs more information about the third-party cryptanalysis on the cipher. There are no details about the side-channel attacks on the Elephant. It only provides a list of third-party cryptanalysis of Spongent and Keccack. In other words, Elephant needs more third-party security analysis.

3.3. GIFT-COFB

GIFT-COFB [26] is a lightweight cryptography algorithm based on the GIFT block cipher. The GIFT-COFB supports authenticated encryption using Combined FeedBack (COFB) mode. It does not require expensive inverse operations to decrypt. The recommended parameters are a 128-bit block and a 128-bit tag. The cryptography primitive used in the GIFT-COFB is GIFT-128 [27]. It is a 40-round substitution-permutation network (SPN) cipher with a 128-bit key. The main round function of the GIFT-128 consists of four main phases: initialization, cell substitution, bits permutation, and round key addition. It also requires a key scheduling mechanism and round constants. In addition, the GIFT-128 has a look-up table (LUT) based variant [27] to improve the runtime by using more space.

The authors have presented a hardware performance. According to that, the hardware implementation of the GIFT-COFB needs a gate area of 3927 GE. Moreover, it requires 40 cycles to encrypt a block of plaintext. GIFT-COFB contains details about the VHDL-based implementation and synthesis. The power consumption of this implementation

is 156.3 µW. The energy efficiency details consist of a comparison of some of the other modes of the GIFT-COFB. As the authors stated, GIFT-COFB can be implemented efficiently in software using the bit-slice method. Further, it uses efficient bitwise operations such as xor for the permutation. However, the GIFT-COFB is not parallelizable by design. Besides the performance details provided by the authors, it also contains summaries of software implementations and benchmarking provided by third parties. The security analysis provides the upper bound for the advantage of an adversary who tries to break the GIFT-COFB authenticated encryption. The authors claim that the GIFT-COFB has 64-bit security against IND-CPA and 58-bit security against INT-CTXT.

3.4. Grain-128AEAD

Grain-128AEAD [23] is one of the few stream ciphers in the NIST LWC competition. It is based on the well known cipher Grain [22]. It consists of a 128-bit key and a 96-bit nonce. The structure of Grain-128AEAD is designed using two major components, the pre-output generator and the authenticator generator. The pre-output generator consists of a 128-bit NFSR and a 128-bit LFSR. The authenticator generator consists of an accumulator and a shift register. In the initialization phase, the NFSR and the LFSR are initialized with the key and the nonce. After the initialization, the pre-output generator generates a stream of bits which is later XORed with the plaintext to produce the ciphertext. An important feature mentioned in Grain-128AEAD is that it supports an AEAD mask. It allows us to define which bits are plaintext and which bits are additional data. As the authors state, this will be important when designing new communication protocols since the position of the header can be defined easily.

The Time/Memory/Data trade-off (TMD-TO) is a type of attack that many stream ciphers are not secure against. The internal state of the Grain-128AEAD cannot be reconstructed; hence a TMD-TO attack has a complexity of 2^{128}. The authors claim that it is also secure against algebraic attacks due to the large algebraic degree of the output function. A reused key/nonce pair might cause leakage of information about the plaintext. Therefore, it is not allowed in the Grain-128AEAD. As the authors claim, the Grain-128AEAD has countermeasures against correlation attacks. The reason is that it has a large state. However, it states that the Grain-128 cipher has been broken using a dynamic cube attack. The researchers expect that the improved initialization process will prevent cube and key recovery attacks and claim that the key recovery attack has a complexity of 2^{96}. A few successful fault attacks have been performed against the Grain.

The single bit-by-bit design of the Grain-128AEAD is highly efficient in hardware implementations. The researchers have implemented the algorithm using stm065v536. The Synopsys Design Compiler 2013 is used for synthesis and simulation. They have recorded the required gate count as 3638.5 GE. When the level of parallelization is 32, the gate count is 12,110.5. Additionally, the power requirement of the hardware is 313 nW, and the throughput is 50 kbit/s. Practical cryptanalysis of side-channel attacks is an open direction for research against Grain-128AEAD.

3.5. ISAP

ISAP [16] is a family of permutation-based authenticated ciphers. The four variants of the family are ISAP-A-128A, ISAP-K-128A, ISAP-A-128, and ISAP-K-128. The ISAP-A-128A and the ISAP-A-128 use Ascon-p as their primitive, while the ISAP-K-128A and the ISAP-K-128 use Keccak-p[400]. Like many other NIST LWC algorithms, ISAP is also powered by encrypt-then-MAC mode. The ISAP has a re-keying function to generate session keys. All members of ISAP have a key size of 128-bits. The ISAP-A-128-A and the ISAP-A-128 consist of a 320-bit state, while the ISAP-K-128A and the ISAP-K-128 have 400-bit states. Furthermore, the ISAP implementation does not require inverse operations.

The authors claim all members of the ISAP provide 128-bit security, and the ISAP is specially designed to provide security against passive side-channel attacks. It also states that the re-keying function prevents differential power attacks (DPA) and fault attacks.

The specific Sponge-based structure helps to improve the resistance against simple power analysis (SPA) attacks. Nevertheless, the nonce must never be reused with the same plaintext. Both the Ascon-p and the Keccak cryptographic primitives used in the ISAP have proven security. There are many published security analyses.

As the authors claim, the design of the ISAP is suitable for lightweight software and hardware implementation. The hardware implementation of the ISAP-K-128A only needs a gate area of 12 kGE in the TSMC65nm cell library. The software implementation of ISAP-A-128A only takes 450 cycles per byte on an AVR ATmega328p microcontroller [16]. Both hardware and software implementations provide robust security against certain fault attacks, including DFA, SFA, and SIFA. Moreover, the researchers also have taken countermeasures against tag comparison..

3.6. PHOTON-Beetle

PHOTON-Beetle [18] is another Sponge-based permutation cipher. It is a family of ciphers designed using the Beetle [28] mode. The PHOTON-Beetle-Hash is the family of hash functions introduced with it. The PHOTON-Beetle has a 256-bit internal state and uses a 128-bit tag for authentication. Two recommended variations of the PHOTON-Beetle are PHOTON-Beetle-AEAD[128] and PHOTON-Beetle-AEAD[32]. They use 128-bit data blocks and 32-bit data blocks, respectively. The recommended hash function is PHOTON-Beetle-Hash[32].

According to the security claims, the PHOTON-Beetle-AEAD[128] has 121-bit security against the IND-CPA and the INT-CTXT [29]. Likewise, the PHOTON-Beetle-AEAD[32] provides 128-bit security. Moreover, a collision attack and pre-image attack on the PHOTON-Beetle-Hash has a time complexity of 112-bits and 128-bits. There are two third-party security analyses of the PHOTON-Beetle-AEAD. Dobraunig and Mennink [30] have performed a generic key recovery attack with a data complexity of $2^{122.8}$ and a time complexity of 2^{124}. The second analysis is a collision attack on the PHTON-Beetle-Hash by Mege [30], in which the data complexity is $2^{111.5}$. The Sponge mode used in the PHOTON-Beetle is specifically designed for lightweight applications. Unlike the original Sponge mode, the Beetle Sponge mode uses combined feedback of the permutation to update the state. The reason behind using this mode in the PHOTON-Beetle is that it prevents the attacker from processing multiple blocks simultaneously.

The software implementation developed by the researchers of the PHOTON-Beetle was executed on an 8-bit AVR microcontroller to measure the performance. They implemented a ROM-optimized version and a speed-optimized version for each member of the PHOTON-Beetle family. The RAM usage of the implementation is around 86 bytes. The ROM-optimized version uses only 2416 bytes of flash memory, while the speed-optimized version uses 4364 bytes. The ROM-optimized and speed-optimized versions need 8128.03 and 4835.35 cycles per byte of data, respectively. However, the PHOTON-Beetle documentation lacks details on third-party security analysis of side-channel attacks and hardware implementation.

3.7. Romulus

The next finalist algorithm in the list is Romulus [31], a tweakable block cipher (TBC) [21] based on the Skinny block cipher [32]. Romulus has three main variations; Romulus-N, Romulus-M, and Romulus-T. Among these, Romulus-N is the main variant. Romulus-M provides security against misused nonce, and Romulus-T is leakage-resilient. It also consists of a hash function known as Romulus-H. The mode of the Romulus structure is very similar to the COFB. All three variants of the Romulus encryption have a 128-bit key, a 128-bit nonce, and a 128-bit tag. The data block size is 128 bits. It allows encrypting/decrypting 2^{59} bytes of data, including the associated data using a single key-value pair. Furthermore, the Romulus-H produces 256-bit hash values for any input.

As the designers claim, Romulus-N and Romulus-M provide 128-bit security. Romulus-M provides 128-bit security even if the nonce is reused. Additionally, Romulus-T offers

121-bit security. They further claim that the hash function Romulus-H has 121-bit security against collision, pre-image, and second pre-image attacks. The variant Romulus-M is secure against the release of unverified plaintext (RUP) attacks. Another claim that authors have made is that the heavily protected implementation of the Skinny-128-384+ used in the Romulus-T prevents information leakages for many side-channel attacks. Moreover, the underlying block cipher, Skinny, is proven to be secure against related-tweaky attacks.

The hardware implementation of the most lightweight variant of the Romulus developed by the researchers, the Romulus-N, only needs 6325 GEs. The Romulus-M also requires a similar amount of gate area. However, there is a research opportunity to implement the Romulus-T on hardware and measure the performance. Further, the study done by the authors lacks more performance-related metrics, such as throughput and cycles. In addition to hardware implementation details, it presents the software implementation on an 8-bit AVR microcontroller, specifically ATmega644. It, too, lacks important performance-related measurements such as RAM usage and the number of cycles per data byte.

3.8. SPARKLE

SPARKLE [33] is a family of permutation ciphers submitted to the NIST lightweight cryptography competition. It consists of an authenticated cipher named Sponge-based cipher for Hardened but Weightless Authenticated Encryption on Many Microcontroller (SCHEWAEMM), a hash function called Efficient Sponge-based and Cheap Hashing (ESCH). The SPARKLE is constructed using an ARX structure. However, it has improved against linear attacks. The three variants of the SCHWAEMM that use 128-bit, 256-bit, and 192-bit key sizes are known as SCHWAEMM256-128, SCHWAEMM256-256, and SCHWAEMM192-192, respectively. All of them use a 256-bit state and a 256-bit nonce. The ESCH has a state size of 384-bits. The ARX structure in the SPARKLE performs XOR and other operations on 32-bit words to improve the performance on low-end and high-end microcontrollers.

SCHWAEMM256-128, SCHWAEMM192-192 and the SCHWAEMM256-256 provide 120-bit, 184-bit and 248-bit security, respectively. Nevertheless, the security levels are only valid when the nonce is unique. The authors believe a complete state recovery attack will not be successful in the nonce misused case since the adversary can only control a part of the state. According to the specification, the permutation is protected against several differential attacks by bounding the MEDCP using a truncated tail. Furthermore, it can be theoretically proven that the SPARKLE is secure against linear attacks, Boomerang attacks, impossible differential attacks, Yoyo games, and zero-correlation attacks.

As the authors state, Alzette, the ARX-box used in the SPARKLE, has properties that make it possible to run the algorithm parallel up to a certain level. However, it is possible only when the CPU supports Single Instruction Multiple Data (SIMD). The implementations were executed on 8-bit AVR and 32-bit ARM microcontrollers. The C implementation of the SPARKLE256 with 10 rounds needs 992 cycles per byte in the AVR and 46 cycles per byte in the ARM. In addition, it uses static RAM of 40 bytes, and the code size is 348 bytes.

3.9. TinyJAMBU

TinyJAMBU [34] is an AEAD encryption scheme based on the JAMBU, one of the most popular contestants in the CAESAR competition. It uses smaller block sizes and a more lightweight keyed permutation. The main variation, TinyJAMBU-128, uses a 128-bit key and a 128-bit state. The main component of the keyed permutation is a nonlinear feedback register (NFRS). The NFSR is used to update the current state. Moreover, the number of rounds is 640 or 1024, depending on the operation phase of the cipher. The special feature of this NFSR's feedback primitive is that 32 updates can be executed in parallel on a 32-bit CPU. The encryption and decryption algorithms have four main phases, initialization, processing associated data, processing plaintext/ciphertext, and finalization/verification. A 64-bit tag is produced in the finalization phase of the encryption algorithm. The other two variants, TinyJAMBU-192 and TinyJAMBU-256, have 192-bit and 256-bit keys.

According to the specification, the TinyJAMBU-128 has 112-bit security, while the TinyJAMBU-192 and the TinyJAMBU-256 have 168-bit and 224-bit security, respectively. The researchers claim that the TinyJAMBU prevents the recovery of the key even when the nonce is misused, and the maximum forgery advantage is 2^{-15}. Moreover, the TinyJAMBU provides strong protection against differential forgery attacks. The probability of successfully performing a differential key recovery attack is as low as 2^{-83}.

The researchers have implemented the TinyJAMBU-128 in hardware to measure the performance. The NIST LWC Hardware API is used for this purpose, and the Synopsys Design Compiler is used for synthesis. The gate area needed for TinyJambu-128 is 3223 GE for 8 rounds per clock cycle. The throughput for the message is 135 Mbps. The software performance has been measured on the ARM Cortex-M4F microcontroller. The code size of software implementation is 872 bytes, and it consumes 394 cycles per byte when encrypting 16 bytes of data.

3.10. XOODYAK

XOODYAK [35] is a cryptographic primitive designed for AEAD encryption, hashing, pseudo-random bit generation, etc. The XOODYAK uses the XOODO as the permutation, which has a 384-bit internal state. It is also inspired by the Keccak-p permutation. There are two different modes of the XOODYAK, hash mode and the keyed mode. When initialized with a key, it can be used in keyed mode. The mode of operation used in the XOODYAK is called Cyclist. The Cyclist object contains the state of the primitive.

As mentioned in the specification, encrypting and decrypting require 91.2 and 91.3 cycles per byte, respectively. The results presented are for an ARM Cortex-M0 microcontroller. Additionally, the code size for the XOODYAK is 3494 bytes. Likewise, researchers have also analyzed the performance of the XOODYAK for ASIC and FPGA. The minimum gate area required by the ASIC implementation is recorded as 8101 GE with a frequency of 200 MHz. The FPGA implementations were executed on a Xilinx Artix-7 board. The XOODYAK specification lacks details about practical cryptanalysis.

3.11. Summary of Existing Cryptanalyses of NIST LWC Finalist Algorithms

We summarize the existing cryptanalytic techniques applied to the NIST LWC finalist algorithms. Table 2 presents a summary of known cryptanalyses of the finalist algorithms. The summary incorporated here mainly includes the result from the application of the attacks to the cipher itself; there may be other published results on the underlying primitive/permutation function that are not included in Table 2. For this reason, it may seem from Table 2 that there are no published analyses of ISAP; however, we note that there are several third-party results in the existing literature on the underlying primitives of ISAP, namely, Keecak and Ascon-p.

We can observe from Table 2 that a variety of classical and side-channel attacks, e.g., linear, differential, differential-linear, interpolation, cube/cube-like, slide, distinguishing attacks, and correlation power analysis, are explored on different candidates. The goals of these attacks include key recovery, forgery, state recovery, and distinguishing the output from a random output. In most cases, the attacks were applied to round-reduced or tweaked versions of the ciphers. Based on the publicly available analyses, all the candidates still show to have their claimed security, i.e., no published attack yet that breaks the security claim of the 10 finalists.

From Table 2, it is also apparent that there are only a limited number of investigations of side-channel attacks on the NIST LWC finalists. Future research can focus on this direction to analyse these algorithms from the perspective of side-channel leakage and resistance against side-channel attacks.

Table 2. Summary of known analyses of the finalist algorithms

Method	Cipher	Results
Linear	GIFT-COFB	KRA – 15-round with TC/DC/MC: $2^{90.7}/2^{62}/2^{96}$ [36]; KRA – 16-round with TC/DC/MC: $2^{122.8}/2^{62.1}/2^{47}$ with success probability 80.01% . [37]
	TinyJambu	KRA – TC: practical and DC: $2^{96.8}$ with success probability 82%. [38]
Differential	Ascon	FA, TC: 2^{101}, for 4 rounds of 12. [39]
Differential-Linear	Ascon	KRA, TC: $2^{31.4}$, for 7 rounds of 12. [40]
	TinyJambu	FA – 338-round with probability $2^{-62.68}$, diffential on 384-round with probability $2^{-70.64}$. [41]
	Xoodyak	KRA - TC: $2^{23.34}/2^{22.04}$ and DC: $23.34/2^{22.04}$ for 4/5-round, respectively, [42]; 4-round rotational differential-linear distinguisher for Xoodoo. [43]
Interpolation	Elephant	Applied to DELIRIUM, TC: $2^{98.3}$, MC: 2^{70}. [44]
Cube/cube-like	Ascon	KRA – TC: 2^{123} for 7-round (nonce-respecting) [45]; KRA/SRA – 7-round Ascon-128a (nonce-misuse) with TC: $2^{116.2}$ and DC: 2^{117} [46]
	TinyJambu	KRA – 428-round and DA – 437-round [47]; KRA – 440-round and DA – 476-round [48]; FA – TC/DC: $2^{42}/2^{32}$ using $2^{12}/2^{10}$ related keys for TinyJAMBU-192/256, respectively, [49]; KRA – TC/DC: $2^{23}, 2^{20}, 2^{18}$ using related key differentials for TinyJAMBU-128/192/256, respectively, [49]
	Grain-128AEAD	KRA – TC/DC: $2^{123}/2^{96}$ for 190-round, DA – TC/DC: 2^{96} for 189-round [50]; KRA – TC/DC: $2^{126.26}/2^{96}$ for 191-round [51]; DA – TC: $2^{28.22}/2^{34.69}$ for 205/235-round Grain-128a under single-key/weak-key setup [52]
	Xoodyak	KRA – TC: $2^{43.8}$ for 6-round (out of 12). [53]
Slide	TinyJambu	KRA, TC: 2^{64} [54]
Side channel	Elephant	KRA using CPA – recovers the key for the 160-bit variant in about a minute using 35 power traces [55]
	GIFT-COFB	KRA with DCSCA – TC: $2^{18.39}$ and MC: $2^{25.39}$, the TC is $2^{13.39}$ if 32 encryptions per session are assumed
Others	GIFT-COFB	KRA – TC:2^4 for 2-round GIFT-COFB [56]; FA – success probability $qd/2^{n/2}$, given qd forgery attempts, also shows FA possible with $2^{n/2}$ attempts using a single known-plaintext encryption query [57]; DA – 2 and 6 rounds of GIFT-COFB [58]; updated bound for privacy [59]
	PHOTON-Beetle	KRA, QC: $2^{22.8}$ [60]; related key FA with an updated security bound of the authentication [59]
	Romulus	MA on Romulus-M [61]
	SPARKLE	MITM based DA – Sparkle-256/384/512 with 4/4/5 steps, respectively, with practical TCs and MCs [62]
	TinyJambu	DA [63] – TC/DC: 2^{23} for 544-round in secret-key settings, TC/DC: $2^{16}/2^{23}$ for full-round 128/192-bit versions in known-key settings, TC/DC: 2^{23} for 1152-round 256-bit version in known-key settings
	Xoodyak	DA–12-round Xoodoo (permutation of Xoodyak) with TC:2^{33} [64]

Key: KRA—key recovery attack; FA—forgery attack; DA—distinguishing attack; SRA—state recovery attack; CPA—correlation power analysis; DCSCA—differential ciphertext side-channel attack; MA—matching attack; MITM—meet-in-the-middle; TC—time complexity; DC—data complexity; MC—memory complexity; QC—query complexity.

4. Fault Analysis of NIST LWC Finalist Algorithms

A fault attack is a powerful type of attack that has many variations. Researchers pay considerable attention to securing their cryptographic scheme against fault attacks. Baski et al. [65] present a detailed introduction to fault attacks and brief descriptions of fault attacks on symmetric cryptosystems. Note that all the NIST LWC competition algorithms are symmetric key ciphers. A fault analysis consists of two main tasks; injecting fault and analyzing the output and/or input to recover the secret key, the internal state, or the input. Every fault attack requires a fault model, and different fault models can be used to describe the effect of the fault on the cipher. For instance, the precise bit flip model is used to flip a bit on an exact location. Single/multiple fault adversary and random/deterministic fault models are other well-known fault models used in practice [65].

Baski et al. explain how different alteration methods work. There are three types of data alteration methods; volatility, modification of operation, and modification of operand. There are many methods of injecting faults into a device that runs the encryption algorithm. Clock glitches, power glitches, laser, optics, and electromagnetic (EM) emanation are a few of them. The most practical method is the clock/power glitch, which suddenly changes the input power or the clock length for a short period. Other forms, such as a laser, may require comparatively expensive, large equipment and expert knowledge of hardware. After injecting the faults, the attacker analyzes the output. Based on the methods used for

the analyses, fault attacks can be divided into a few categories; we explain them and their applications to the NIST LWC finalists in the following sections.

4.1. Differential Fault Attacks (DFA)

Biham et al. [66] first introduced differential fault attack (DFA) in 1997. It was to break the Data Encryption Standard (DES) using only 50 to 200 faulty and non-faulty ciphertexts. The attack is conducted by encrypting the same unknown plaintext twice, one with a fault and the other without a fault. The fault should be a single bit. Biham et al. inject the fault in the 16-th round of the DES. Next, many of the incorrect values for each 6-bit key can be eliminated with the help of the difference distribution table of the S-boxes. Then, the same procedure can be applied to rounds 14 and 15 to recover the last 48-bit sub-key. Then other 8-bits of the key can be easily recovered using the brute force method. An alternative approach to recovering the remaining key bits will be applying the attack again. The authors claim this allows for the recovery of larger keys, such as the 168-bit keys in 3DES. However, due to the differential nature, this method will only work in the nonce-misused case.

Salam et al. [67] presented three differential fault attack models to Grain-128AEAD to recover the state of the Grain using a small number of faulty outputs. To identify potential fault targets, they generated and analyzed the algebraic normal form (ANF) of consecutive keystream bits. The first attack is called the "Bit-flipping attack". In this attack, a single-bit fault is injected, and the derivatives of the faulty and non-faulty results are used to recover the state bit. This method can recover the 223 state bits by injecting $2^{6.64}$ faults, and the data complexity is $2^{7.80}$. The next model is "Probabilistic random fault attack," which is very similar to the bit-flipping method but uses a random bit-flipping instead of a deterministic bit-flipping model. The probabilistic random bit-flipping approach could recover 223 bits of the state using $2^{10.45}$ with a data complexity of $2^{11.60}$. The third attack is a "Deterministic random fault attack". This method also uses a random bit-flipping model. However, the value of the random fault can be conclusively determined. This model can also recover 223 state bits using $2^{9.39}$ faults with a data complexity of $2^{12.98}$.

SBCMA — Semi-Blind Combined Middle-round Attack is a differential-like fault attack recently proposed by Hou et al. [68]. The attack injects faults in an S-box and observes the side-channel leakage for two rounds to recover the secret key. Compared to other fault attacks, SBCMA does not require knowledge of the inputs and outputs of the cipher. Two different classes of the attack were considered in this work:

- SBCMA A – observes differentials in each S-box
- SBCMA B – observes differentials in each Quotient group

These two classes were evaluated against the lightweight AEAD scheme GIFT-COFB. The SBCMA A is reported to require 94.32 faults in 13.79 sessions with a known fault mask and 76.96 faults in 11.62 sessions for random faults with chosen fault mask to recover the master secret key. On the contrary, the SBCMA B is reported to require 145.58 faults in 20.20 sessions with a known fault mask and 100.08 faults in 14.51 sessions for random faults with chosen fault mask to recover the master secret key. Note that the fault mask here refers to the input difference to the S-box.

Recently, DFA has been investigated against the finalist candidate PHOTON-Beetle [69]. This work applies two fault models: (1) random fault and (2) known fault. Under the random fault, it is assumed that the attacker can inject fault at a specific time by which one nibble of the internal state is changed to a random one. The known fault model also changes the internal state to a random one, but the faulty value is known to the attacker, e.g., assuming the attacker perfectly knows the statistical distribution of the fault. The reported results in this work claim to retrieve the secret key with approximately $2^{37.15}$ and $2^{11.05}$ faulty queries for random and known fault models, respectively. The offline time and memory complexities for these attacks are 2^{16} and 2^{10} nibbles for the random fault, and 2^{11} and 2^{9} nibbles for the known fault. Further, the work also reports the number of faults can be reduced to 640 if a precise bit-flip model were assumed; however, this model is less practical.

4.2. Collision Fault Attacks (CFA)

Collision Fault Attacks (CFA) [70] is a combination of fault attacks and collision attacks, which involves finding collisions by injecting faults. The model is used against the AES to recover the secret key. In this attack, the attacker should be able to inject a fault to a specific bit in the state and should be able to collect faulty and non-faulty ciphertext outputs. After finding the collisions, the attacker can determine the key byte, which corresponds to the byte containing the faulty state bit. There are five attack models presented. The attack recovers the full 128-bit key using only 32 faults. The authors also propose a second attack suitable for the particular case where the system is protected by a memory encryption mechanism (MEM). This attack needs 285 faults to recover the key. An important point presented is that the MEM would not increase the security by a significant margin as expected. The third and fifth attacks require 1024 and 4096 faults, respectively.

In 2020, a collision fault attack was presented by Liu et al. [71] to the GIFT-COFB encryption scheme. First, a CFA was applied to the primitive (GIFT-128) used in the GIFT-COFB. The complete key of GIFT-128 can be recovered using the round keys of the first and the second round keys. The attack requires the fault to be injected into a specific bit. According to the properties of the S-box used in GIFT-128, a collision should exist for a correct ciphertext in the set of faulty ciphertexts. Then, the attacker can determine 2 bits of the round key. The authors show that this attack can recover the key using only 64 faulty ciphertexts and with a data complexity of 2^{10}.

4.3. Statistical Fault Attacks (SFA)

Statistical fault attacks (SFA) are firstly introduced in Fuhr et al. [72]. This method is effective in the nonce-based AEAD encryption schemes since it does not need faulty and correct ciphertext of the same plaintext. Another interesting fact about this attack is that, in most cases, plaintext does not have to be known. The attack was demonstrated against the AES and recovered the complete key using 1–18 faults. The fault is required to be injected into a selected byte. An important requirement of the attack is that the faulty byte is extremely biased (toward 0 or 255). First, the attacker collects several ciphertexts with faults. The fault can be injected just before the last or the second last round after adding the round constant. Then, an expression to the faulty byte can be obtained by inverting the MixColumn, ShiftRow, and SubByte operations of the last (or second last round). The input to this expression will be a byte of the faulty ciphertext and a byte of key-guess. Fuhr et al. [72] propose three distinguishers to select the correct key byte out of the guessed ones.

The first distinguisher is the *maximum likelihood method*; the likelihood for each key byte is calculated for all corresponding faulty ciphertext bytes, and the key byte with maximum likelihood is chosen. However, this method requires perfect knowledge of fault distribution. When the fault distribution is unknown, yet the faulty value is biased towards 0 or 255, the *Hamming weight* distinguisher can be used. When the attacker only knows the distribution of the faulty value is biased, the *Square Euclidean Imbalance* distinguisher can be used. This procedure can be repeated 16 times with different fault positions to recover all 16 key bytes of the AES. The probability of recovering the correct byte by an attack on round 9 is high as 99%.

The original SFA method proposed by Fuhr et al. only applies to a cryptographic primitive such as the AES. Dobraunig et al. [73] present statistical fault attacks on advanced modes of operations such as GCM, CCM, EAX, and OCB. The outputs to the primitives can be determined using ciphertext and plaintext in CCM, EAX, and GCM modes. Then the same method in Fuhr et al. [72] can be applied to recover the key. The attack on OCB mode is possible by using a partial plaintext block, which is padded with a known value (typically 0s or 0s followed by 1). Using this known value, the output of the primitive can be determined; hence, the attack in Fuhr et al. is applicable. However, the SFA attack on XEX-like construction can be complicated since the output of the primitive is masked with a secret value. A successful method is proposed against the scheme AES-COPA [74], which

uses a XEX-like structure. In this case, the secret mask only depends on the key. Here, the trick is to guess the value for the combination of the secret mask and the key and apply the SFA to determine the correct one. Then, the SFA can be applied again to recover the key bytes. The authors also explain how to adopt the SFA for the tweakable block ciphers. However, this method requires knowledge of the tweak.

4.3.1. Statistical Ineffective Fault Attack (SIFA)

Several methods have been introduced to detect the occurrence of faults. One is the redundancy-based detection of faults. Statistical ineffective fault attack (SIFA) [75] is a fault attack that can be used to recover the secret of a cryptosystem where fault detection is applied. This method is somewhat similar to the SFA. However, the attacker should collect only the ciphertext where the fault is ineffective. The attack was initially applied to AES. The attack requires that the distribution of the diagonal of the fault distribution table [75] should be non-uniform. After collecting the ciphertexts where the fault is ineffective, the same procedure in Fuhr et al. [72] is applied to determine the key.

The SIFA on the Ascon AEAD cipher is presented in Dobraunig et al. [75]. The original SIFA cannot be applied to the Ascon due to the permutation structure. The proposed method uses a double-fault model. In this model, two faults are injected into two selected 5-bit S-boxes of the last round of the Ascon permutation of the finalization phase. Instead of ciphertext, tags (MAC) are collected where the faults are ineffective. Then, the output of these S-boxes can be calculated for each key hypothesis by taking the inverse of the linear diffusion layer of the Ascon permutation [24]. Since the diffusion layer can be modeled as a multiplication with a matrix, the inverse can be achieved by taking the inverse of the matrix. However, the output of a single S-box will depend on the 68-bit of the key. Therefore, choosing the correct 68 bits of the key would require a time complexity of 2^{68}. As a solution, the authors present a key dividing strategy to reduce this complexity. It uses a set of linear binary equations to derive the key.

4.3.2. Single Event Transient Fault Analysis (SETFA)

Single Event Transient Fault Analysis (SETFA) [76] is a recent fault analysis method for the Elephant authenticated encryption scheme proposed by Joshi et al. [76]. This method targets a hardware implementation of the cipher. The attack consists of five steps, (1)–identification of fault points (hot spots), (2)–choosing fault combinations, (3)–fault injection, (4)–encrypting with faults, and (5)–fault analysis. The attack was applied to Dumbo, a variant of Elephant. However, the faults used in this method are based on set 0 or set 1. In other words, the attacker should be able to select the exact value of a specific bit of the intermediate state. The authors claim the attack on Dumbo can recover the correct keys with a probability of 30% where the number of ciphertext queries is 140.

4.3.3. Fault Intensity Map Analysis (FIMA)

Fault Intensity Map Analysis (FIMA) is a recent method introduced by Ramezanpour et al. [77] that uses information from fault bias and the correlation between fault distribution and intensity to recover the secret. In general, it combines several existing techniques fault sensitivity analysis (FSA), differential fault intensity analysis (DFIA), ciphertext-only fault analysis (CFA), and statistical ineffective fault analysis (SIFA). Ramezanpour et al. demonstrated the application of FIMA to Ascon. They claimed that combining several techniques can recover the 128-bit secret key of Ascon with lesser than half of the fault injections than the previous techniques that rely only on the bias. They also show that this new method performs six-times better than other previous methods when countermeasures, e.g., ineffective and error detection, are in place. The method was further extended with a deep neural network (NN)-based fault intensity map analysis (FIMA-NN) and tested against AES S-box [78]. FIMA-NN also shows promising results and can be further investigated to evaluate the NIST LWC finalists.

4.4. Summary of Fault Analyses of the LWC Finalists

We summarize the different fault analyses applied to the NIST LWC finalists in Table 3. Notice that several approaches have emerged in recent years apart from the traditional differential fault attack. Some of these new approaches have not been fully explored yet against all the NIST LWC finalists. For instance, the recently introduced technique FIMA combines different fault techniques, e.g., biased distribution with intensities, for applying the fault attack with lesser faults. Further research can be conducted towards this direction to understand the resistance of the NIST LWC finalists against fault attacks.

Table 3. Summary of fault analyses of the LWC finalists

Cipher	Fault Attack	Model	Results
ASCON	SIFA [79]	Random fault, biased fault distribution	KRA—with 12.5 to 2500 faulty outputs for highly biased to more uniform distributions, respectively
	FIMA [77]	Biased fault distribution with varying intensities	KRA—claimed to require 50% lesser faults than previous techniques
	SSFA [80]	Bit-reset, multi-byte	KRA—QC: 70 to 100
Elephant	SETFA [76]	Stuck-at fault (set to 0 or 1) at specific wires	KRA on Dumbo—85–250 queries
GIFT-COFB	CFA [71]	Bit-flip	KRA—with 64 faulty ciphertexts, DC: 2^{10}
	SBCMA A	Random with chosen fault mask [68]	KRA—QC: $2^{6.27}$ in 11.62 sessions
	SBCMA A	Known fault mask [68]	KRA—QC: $2^{6.6}$ in 13.79 sessions
	SBCMA B	Random with chosen fault mask [68]	KRA—QC: $2^{6.64}$ in 14.51 sessions
	SBCMA B	Known fault mask [68]	KRA—QC: $2^{7.19}$ in 20.20 sessions
Grain-128AEAD	DFA	Random [81]	SRA—Grain-128a using 4-10 faults
		Bit-flipping [67]	SRA—QC: $2^{6.64}$ and DC: $2^{7.80}$ faults
		Probabilistic random [67]	SRA—QC: $2^{10.45}$ and DC: $2^{11.60}$ faults
		Random [67]	SRA—QC: $2^{9.39}$ and DC: $2^{12.98}$ faults
PHOTON-Beetle	DFA	Random fault [69]	KRA—QC: $2^{37.15}$, TC: 2^{16} and DC: 2^{10}
		Known fault [69]	QC: $2^{11.05}$, TC: 2^{11} and DC: 2^{9}
		Bit-flipping [69]	QC: $2^{9.32}$

Key: KRA—key recovery attack; SRA—state recovery attack; SIFA—statistical ineffective fault attack; SETFA—single event transient fault attack; FIMA—fault intensity map analysis; SSFA—sub-set fault analysis; CFA—ciphertext-only fault analysis; SBCMA A—semi-blind combined with middle-round attack (observes differentials in each S-box); SBCMA B—semi-blind combined with middle-round attack (observes differentials in each Quotient group); DFA—differential fault attack; TC—time complexity; DC—data complexity; QC—query complexity

5. Conclusions

This paper reviews the specifications of the NIST LWC finalist algorithms and their structures. A good number of these algorithms employ Sponge-based constructions or a related construction, such as the Duplex. The reason for this common choice is the simplicity and efficiency of the Sponge constructions. Many of the algorithms use hash functions as cryptographic primitives. For instance, the KECCAK is used in the Elephant and the ISAP. The tweakable block cipher is another construction used in algorithms such as the Romulus. Additionally, algorithm families have different variations with different purposes. For instance, the Romulus-T is designed to provide security in the case of a misused nonce, and the Ascon-80pq provides security against quantum attacks. However, many of these algorithms need further investigations on cryptanalysis, especially on side-channel and fault attacks.

To the best of our knowledge, none of the finalist algorithms are broken based on the third-party analyses. All the NIST LWC finalist algorithms have shown a good security margin against classical cryptanalysis methods such as linear and differential analysis. It is worthwhile to focus on powerful fault attacks for analyzing these ciphers. Many NIST LWC competition finalist algorithms lack proper and practical analysis against side-channel

and fault attacks. Different fault attack techniques, such as DFA, CFA, SFA, FIMA and SIFA, can be tested on the aforementioned algorithms. Specifically, fault attacks have not been applied to Ascon, Elephant, TinyJambu, Xoodyak, ISAP, Romulus, and SPARKLE. For Elephant, a fault attack has been applied but the attack model is stringent and there is possibility to explore further in terms of other fault models. Similarly, Grain-128AEAD has only been investigated against differential fault attack; other fault attack techniques have not been explored against this cipher. We note that NIST does not require security claims against side-channel/fault attacks for the proposals submitted to the NIST LWC competition. However, in the submission requirement, NIST also stated that the ability to provide resistance against side-channel/fault attacks easily and at a low cost is highly desired. Therefore, investigating the application of different fault attacks on these ciphers will allow us to better compare the NIST LWC finalists.

With the advancement of science and technology, large-scale quantum computers will be available in the near future. Thus, quantum-safe cryptosystems will be essential for secure communication in the future. Therefore, it is worthwhile to investigate quantum-safe lightweight cryptography for future standardization. This can be identified as one of the future challenges of NIST lightweight cryptography standardization.

Author Contributions: Conceptualization, I.S., J.A. and H.M.; methodology, I.S., J.A. and H.M.; investigation, I.S., J.A. and H.M.; writing—original draft preparation, H.M.; writing—review and editing, I.S., J.A. and H.M.; supervision, I.S. and J.A; project administration, J.A.; funding acquisition, J.A. All authors have read and agreed to the published version of the manuscript.

Funding: Iftekhar Salam's work is supported by the Ministry of Higher Education Malaysia through the Fundamental Research Grant Scheme (FRGS), project no. FRGS/1/2021/ICT07/XMU/02/1, as well as the Xiamen University Malaysia Research Fund under Grants XMUMRF/2019-C3/IECE/0005 and XMUMRF/2022-C9/IECE/0032. APC is supported by the Rabdan Academy, UAE.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Chen, M.; Liu, W.; Wang, T.; Zhang, S.; Liu, A. A game-based deep reinforcement learning approach for energy-efficient computation in MEC systems. *Knowl.-Based Syst.* **2022**, *235*, 107660. [CrossRef]
2. Chen, M.; Liu, W.; Zhang, N.; Li, J.; Ren, Y.; Yi, M.; Liu, A. GPDS: A multi-agent deep reinforcement learning game for anti-jamming secure computing in MEC network. *Expert Syst. Appl.* **2022**, *210*, 118394. [CrossRef]
3. Wang, Y.; Lou, X.; Fan, Z.; Wang, S.; Huang, G. Verifiable Multi-Dimensional (t,n) Threshold Quantum Secret Sharing Based on Quantum Walk. *Int. J. Theor. Phys.* **2022**, *61*, 24. [CrossRef]
4. NIST, Submission Requirements and Evaluation Criteria for the Lightweight Cryptography Standardization Process. 2018. pp. 1–17. Available online: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/final-lwc-submission-requirements-august2018.pdf (accessed on 3 July 2022)
5. Lightweight Cryptography | CSRC. Available online: https://csrc.nist.gov/Projects/lightweight-cryptography/ (accessed on 3 July 2022).
6. So What Is AEAD? Furthermore, Why Is It So Important for Encryption? | by Prof Bill Buchanan OBE | ASecuritySite: When Bob Met Alice | Medium. Available online: https://medium.com/asecuritysite-when-bob-met-alice/so-what-is-aead-and-why-is-it-so-important-for-encryption-8e2bf16eed6f (accessed 3 August 2022).
7. Jimale, M.A.; Z'aba, M.R.; Kiah, M.L.M.; Idris, M.Y.I.; Jamil, N.; Mohamad, M.S.; Rohmad, M.S. Authenticated encryption schemes: A systematic review. *IEEE Access* **2022**, *10*, 14739–14766. [CrossRef]
8. Elsadek, I.; Aftabjahani, S.; Gardner, D.; MacLean, E.; Wallrabenstein, J.R.; Tawfik, E.Y. Hardware and Energy Efficiency Evaluation of NIST Lightweight Cryptography Standardization Finalists. In Proceedings of the 2022 IEEE International Symposium on Circuits and Systems (ISCAS), Austin, TX, USA, 27 May–1 June 2022; pp. 133–137.
9. Pugh, S.; Raunak, M.S.; Kuhn, D.R.; Kacker, R. Systematic testing of lightweight cryptographic implementations. In Proceedings of 2019 Lightweight Cryptography Workshop, Gaithersburg, ML, USA, 4–6 November 2019; National Institute of Standards and Technology: Gaithersburg, ML, USA, 2019.
10. Abed, F.; Forler, C.; Lucks, S. General classification of the authenticated encryption schemes for the CAESAR competition. *Comput. Sci. Rev.* **2016**, *22*, 13–26. [CrossRef]
11. Bertoni, G.V.A.G.; Daemen, J.; Peeters, M. Sponge Functions. In Proceedings of the ECRYPT Hash Workshop, Barcelona, Spain, 24–25 May 2007.

12. Bogdanov, A.; Knežević, M.; Leander, G.; Toz, D.; Varici, K.; Verbauwhede, I. SPONGENT: The design space of lightweight cryptographic hashing. *IEEE Trans. Comput.* **2013**, *62*, 2041–2053. [CrossRef]
13. Tim, B.; Chen, Y.L.; Dobraunig, C.; Mennink, B. Elephant v2 Specification. Submission to NIST LWC Project. 2021. Available online: https://www.esat.kuleuven.be/cosic/elephant/ (accessed on 8 December 2022).
14. Bogdanov, A.; Knudsen, L.R.; Leander, G.; Paar, C.; Poschmann, A. PRESENT: An Ultra-Lightweight Block Cipher. In Proceedings of the International Workshop on Cryptographic Hardware and Embedded Systems, Vienna, Austria, 10 September 2007; Springer: Berlin/Heidelberg, Germany, 2007.
15. Keccak Team. Available online: https://keccak.team/keccak.html (accessed on 4 August 2012).
16. Dobraunig, C.; Eichlseder, M.; Mangard, S.; Mendel, F.; Mennink, B.; Primas, R. ISAP v2.0. Submission to NIST LWC Project. 2021. Available online: https://isap.iaik.tugraz.at (accessed on 18 July 2022).
17. Guo, J.; Peyrin, T.; Poschmann, A. The PHOTON family of lightweight hash functions. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 14 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6841, pp. 222–239. [CrossRef]
18. Bao, Z.; Chakraborti, A.; Datta, N.; Guo, J.; Nandi, M.; Peyrin, T.; Yasuda, K. PHOTON-Beetle Authenticated Encryption and Hash Family. Submission to NIST LWC Project. 2021. Available online: https://www.isical.ac.in/~lightweight/beetle/ (accessed on 8 December 2022).
19. Chakraborty, B.; Nandi, M. ORANGE. Submission to NIST LWC Project. 2021. Available online: https://csrc.nist.gov/CSRC/media/Projects/Lightweight-Cryptography/documents/round-1/spec-doc/orange-spec.pdf (accessed on 8 December 2022).
20. Bertoni, G.; Daemen, J.; Peeters, M.; Van Assche, G. Duplexing the sponge: Single-pass authenticated encryption and other applications. In Proceedings of the International Workshop on Selected Areas in Cryptography, Toronto, ON, Canada, 11–12 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; Volume 7118, pp. 320–337. [CrossRef]
21. Liskov, M.; Rivest, R.L.; Wagner, D. Tweakable Block Ciphers. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 17–21 August 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 31–46.
22. Hell, M.; Johansson, T.; Meier, W. Grain: A stream cipher for constrained environments. *Int. J. Wirel. Mob. Comput.* **2007**, *2*, 86–93. [CrossRef]
23. Hell, M.; Johansson, T.; Maximov, A.; Meier, W.; Sonnerup, J.; Yoshida, H. Grain-128AEADv2—A lightweight AEAD stream cipher. Submission to NIST LWC Project. 2021. Available online: https://grain-128aead.github.io/ (accessed on 8 December 2022).
24. Dobraunig, C.; Eichlseder, M.; Mendel, F.; Schläffer, M. Ascon v1.2. Submission to NIST LWC Project. 2021. Available online: https://ascon.iaik.tugraz.at (accessed on 12 July 2022).
25. Bertoni, G. Keccak sponge function family main document. *Submiss. NIST* **2009**, *3*, 320–337.
26. Banik, S.; Chakraborti, A.; Iwata, T.; Minematsu, K.; Nandi, M.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT-COFB v1.1. Available online: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/gift-cofb-spec-final.pdf (accessed on 15 December 2022)
27. Banik, S.; Pandey, S.K.; Peyrin, T.; Sasaki, Y.; Sim, S.M.; Todo, Y. GIFT: A Small Present (Full version). In Proceedings of the International Conference on Cryptographic Hardware and Embedded Systems, Taipei, Taiwan, 25–28 September 2017; Springer: Cham, 2017; Volume 10529. [CrossRef]
28. Chakraborti, A.; Datta, N.; Nandi, M.; Yasuda, K. Beetle family of lightweight and secure authenticated encryption ciphers. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 218–241. [CrossRef]
29. Smart, N.P.; Paterson, K.; Cramer, R. *Cryptography Made Simple*; Springer: Berlin/Heidelberg, Germany, 2016; Volume 53. [CrossRef]
30. Dobraunig, C. Key Recovery Attack on PHOTON-Beetle. Round 2 Official Comments. Available online: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/round-2/official-comments/photon-beetle-round2-official-comment.pdf (accessed 9 December 2022)
31. Guo, C.; Iwata, T.; Khairallah, M.; Minematsu, K.; Peyrin, T. Romulus v1.3 Specification. Available online: https://romulusae.github.io/romulus/ (accessed on 18 July 2022).
32. Beierle, C.; Jean, J.; Kölbl, S.; Leander, G.; Moradi, A.; Peyrin, T.; Sasaki, Y.; Sasdrich, P.; Sim, S.M. The SKINNY Family of Block Ciphers and Its Low-Latency Variant MANTIS. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 15–18 August 2016; Springer: Berlin/Heidelberg, Germany, 2016. [CrossRef]
33. Beierle, C.; Biryukov, A.; Santos, L.C.d.; Großschadl, J.; Moradi, A.; Perrin, L.; Shahmirzadi, A.R.; Udovenko, A.; Velichkov, V.; Wang, Q. Schwaemm and Esch: Lightweight Authenticated Encryption and Hashing using the Sparkle Permutation Family Corresponding Submitter. 2AD. Available online: https://sparkle-lwc.github.io/ (accessed on 18 July 2022).
34. Wu, H.; Huang, T. TinyJAMBU: A Family of Lightweight Authenticated Encryption Algorithms (Version 2). Submission to NIST LWC Project. 2021. Available online: https://csrc.nist.gov/CSRC/media/Projects/lightweight-cryptography/documents/finalist-round/updated-spec-doc/tinyjambu-spec-final.pdf (accessed on 9 December 2022).
35. Daemen, J.; Hoffert, S.; Mella, S.; Peeters, M.; Assche, G.V.; Keer, R.V. Xoodyak, a lightweight cryptographic scheme. Submission to NIST LWC Project. 2021. Available online: https://keccak.team/xoodyak.html (accessed on 9 December 2022).
36. Zong, R.; Dong, X.; Chen, H.; Luo, Y.; Wang, S.; Li, Z. Towards key-recovery-attack friendly distinguishers: Application to GIFT-128. *IACR Trans. Symmetric Cryptol.* **2021**, *2021*, 156–184. [CrossRef]

37. Sun, L.; Wang, W.; Wang, M. Linear Cryptanalyses of Three AEADs with GIFT-128 as Underlying Primitives. *IACR Trans. Symmetric Cryptol.* **2021**, *2021*, 199–221. [CrossRef]
38. Li, M.; Mouha, N.; Sun, L.; Wang, M. Revisiting the Extension of Matsui's Algorithm 1 to Linear Hulls: Application to TinyJAMBU. *IACR Trans. Symmetric Cryptol.* **2022**, *2022*, 161–200. [CrossRef]
39. Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M. Cryptanalysis of ASCON. *Topics in Cryptology—CT-RSA 2015*; Springer: Berlin/Heidelberg, Germany, 2015; Volume 9048, pp. 371–387. [CrossRef]
40. Tezcan, C. Analysis of Ascon, DryGASCON, and Shamash permutations. *Int. J. Inf. Secur. Sci.* **2020**, *9*, 172–187.
41. Saha, D.; Sasaki, Y.; Shi, D.; Sibleyras, F.; Sun, S.; Zhang, Y. On the Security Margin of TinyJAMBU with Refined Differential and Linear Cryptanalysis. *IACR Trans. Symmetric Cryptol.* **2020**, *2020*, 152–174. [CrossRef]
42. Dunkelman, O.; Weizman, A. Differential-linear cryptanalysis on xoodyak. NIST Lightweight Cryptography Workshop 2022. Available online: https://csrc.nist.gov/csrc/media/Events/2022/lightweight-cryptography-workshop-2022/documents/papers/differential-linear-cryptanalysis-on-xoodyak.pdf (accessed 9 December 2022)
43. Liu, Y.; Sun, S.; Li, C. Rotational Cryptanalysis from a Differential-Linear Perspective. In *Advances in Cryptology—EUROCRYPT 2021*; Canteaut, A., Standaert, F.X., Eds.; Lecture Notes in Computer Science; Springer: Cham, Switerland, 2021; Volume 12696. [CrossRef]
44. Zhou, H.; Zong, R.; Dong, X.; Jia, K.; Meier, W. Interpolation Attacks on Round-Reduced Elephant, Kravatte and Xoofff. *Comput. J.* **2021**, *64*, 628–638. [CrossRef]
45. Rohit, R.; Hu, K.; Sarkar, S.; Sun, S. Misuse-free key-recovery and distinguishing attacks on 7-round ascon. *IACR Trans. Symmetric Cryptol.* **2021**, *2021*, 130–155. [CrossRef]
46. Chang, D.; Hong, D.; Kang, J.; Turan, M.S. Resistance of Ascon Family against Conditional Cube Attacks in Nonce-Misuse Setting. *IEEE Access* **2022**. [CrossRef]
47. Teng, W.L.; Salam, I.; Yau, W.C.; Pieprzyk, J.; Phan, R.C.W. Cube attacks on round-reduced TinyJAMBU. *Sci. Rep.* **2022**, *12*, 5317. [CrossRef]
48. Dutta, P.; Rajasree, M.S.; Sarkar, S. Weak-keys and key-recovery attack for TinyJAMBU TinyJAMBU. *Sci. Rep.* **2022**, *12*, 16313. [CrossRef]
49. Dunkelman, O.; Lambooij, E.; Ghosh, S. Practical Related-Key Forgery Attacks on the Full TinyJAMBU-192/256. Cryptology ePrint Archive. 2022. Available online: https://eprint.iacr.org/2022/1122 (accessed 9 December 2022)
50. Hao, Y.; Leander, G.; Meier, W.; Todo, Y.; Wang, Q. Modeling for Three-Subset Division Property Without Unknown Subset. In *Advances in Cryptology—EUROCRYPT 2020*; Lecture Notes in Computer Science; Canteaut, A., Ishai, Y., Eds.; Springer: Cham, Switzerland, 2020; Volume 12105._17 [CrossRef]
51. Hu, K.; Sun, S.; Todo, Y.; Wang, M.; Wang, Q. Massive Superpoly Recovery with Nested Monomial Predictions. In *Advances in Cryptology—ASIACRYPT 2021*; Lecture Notes in Computer Science; Tibouchi, M., Wang, H., Eds.; Springer: Cham, Switzerland, 2021; Volume 13090._14 [CrossRef]
52. Dalai, D.K.; Pal, S.; Sarkar, S. Some Conditional Cube Testers for Grain-128a of Reduced Rounds. *IEEE Trans. Comput.* **2022**, *71*, 1374–1385. [CrossRef]
53. Zhou, H.; Li, Z.; Dong, X.; Jia, K.; Meier, W. Practical Key-Recovery Attacks On Round-Reduced Ketje Jr, Xoodoo-AE Furthermore, Xoodyak. *Comput. J.* **2020**, *63*, 1231–1246. [CrossRef]
54. Sibleyras, F.; Sasaki, Y.; Todo, Y.; Hosoyamada, A.; Yasuda, K. Birthday-Bound Slide Attacks on TinyJAMBU's Keyed-Permutations for All Key Sizes. In Proceedings of the International Workshop on Security, Tokyo, Japan, 31 August 2022; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13504, pp. 107–127. [CrossRef]
55. Vialar, L. Fast Side-Channel Key-Recovery Attack against Elephant Dumbo. Cryptology ePrint Archive. 2022. pp. 1–13. Available online: https://eprint.iacr.org/2022/446 (accessed on 8 December 2022).
56. Zhong, Y.; Guin, U. Chosen-Plaintext Attack on Energy-Efficient Hardware Implementation of GIFT-COFB. In Proceedings of the 2022 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 27–30 June 2022; pp. 73–76. [CrossRef]
57. Khairallah, M. Security of COFB against Chosen Ciphertext Attacks. *IACR Trans. Symmetric Cryptol.* **2022**, *2022*, 138–157. [CrossRef]
58. Rajan, R.; Roy, R.K.; Sen, D.; Mishra, G. Deep Learning-Based Differential Distinguisher for Lightweight Cipher GIFT-COFB. In *Machine Intelligence and Smart Systems*; Algorithms for Intelligent Systems; Agrawal, S., Gupta, K.K., Chan, J.H., Agrawal, J., Gupta, M., Eds.; Springer: Singapore, 2022; pp. 397–406. [CrossRef]
59. Inoue, A.; Iwata, T.; Minematsu, K. Analyzing the Provable Security Bounds of GIFT-COFB and Photon-Beetle. In *Applied Cryptography and Network Security*; Lecture Notes in Computer Science; Ateniese, G., Venturi, D., Eds.; Springer: Cham, Switzeland, 2022; Volume 13269._4 [CrossRef]
60. Dobraunig, C.; Mennink, B. Tightness of the suffix keyed sponge bound. *IACR Trans. Symmetric Cryptol.* **2020**, *2020*, 195–212. [CrossRef]
61. Habu, M.; Minematsu, K.; Iwata, T. Matching attacks on Romulus-M. *IET Inf. Secur.* **2022**, *16*, 459–469. [CrossRef]
62. Schrottenloher, A.; Stevens, M. Simplified MITM modeling for permutations: New (quantum) attacks. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 2022; Lecture Notes in Computer Science; Dodis, Y., Shrimpton, T., Eds.; Springer: Berlin/Heidelberg, Germany, 2022; Volume 13509, pp. 717–747. [CrossRef]

63. Dunkelman, O.; Ghosh, S.; Lambooij, E. Full Round Zero-sum Distinguishers on TinyJAMBU-128 and TinyJAMBU-192 Keyed-permutation in the Known-key setting. Cryptology ePrint Archive. 2022. Available online: https://eprint.iacr.org/2022/1567 (accessed on 9 December 2022)
64. Liu, F.; Isobe, T.; Meier, W.; Yang, Z. Algebraic Attacks on Round-Reduced Keccak. In Proceedings of the Information Security and Privacy: 26th Australasian Conference, ACISP 2021, Virtual Event, 1–3 December 2021; Springer-Verlag: Berlin/Heidelberg, Germany, pp. 91–110. [CrossRef]
65. Baksi, A.; Bhasin, S.; Breier, J.; Jap, D.; Saha, D. Fault Attacks in Symmetric Key Cryptosystems. Cryptology ePrint Archive 2020. Available online: https://eprint.iacr.org/2020/1267 (accessed on 9 December 2022)
66. Biham, E.; Shamir, A. Differential fault analysis of secret key cryptosystems. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 1997; Springer: Berlin/Heidelberg, Germany, 1997; Volume 1294, pp. 513–525. [CrossRef]
67. Salam, I.; Ooi, T.H.; Xue, L.; Yau, W.C.; Pieprzyk, J.; Phan, R.C.W. Random Differential Fault Attacks on the Lightweight Authenticated Encryption Stream Cipher Grain-128AEAD. *IEEE Access* **2021**, *9*, 72568–72586. [CrossRef]
68. Hou, X.; Breier, J.; Bhasin, S. SBCMA: Semi-Blind Combined Middle-Round Attack on Bit-Permutation Ciphers With Application to AEAD Schemes. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3677–3690. [CrossRef]
69. Jana, A.; Paul, G. Differential Fault Attack on PHOTON-Beetle. In Proceedings of the 2022 Workshop on Attacks and Solutions in Hardware Security (ASHES'22), Los Angeles, CA, USA, 11 November 2022; Association for Computing Machinery: New York, NY, USA, pp. 25–34. [CrossRef]
70. Blömer, J.; Krummel, V. Fault based collision attacks on AES. In Proceedings of the International Workshop on Fault Diagnosis and Tolerance in Cryptography, Online, 16 September 2021; Springer: Berlin/Heidelberg, Germany, 2006; Volume 4236, pp. 106–120. [CrossRef]
71. Liu, S.; Guan, J.; Hu, B. Fault attacks on authenticated encryption modes for GIFT. *IET Inf. Secur.* **2022**, *16*, 51–63. [CrossRef]
72. Fuhr, T.; Jaulmes, E.; Lomne, V.; Thillard, A. Fault attacks on AES with faulty ciphertexts only. In Proceedings of the 2013 Workshop on Fault Diagnosis and Tolerance in Cryptography, Los Alamitos, CA, USA, 20 August 2013; pp. 108–118. [CrossRef]
73. Dobraunig, C.; Eichlseder, M.; Korak, T.; Lomné, V.; Mendel, F. Statistical fault attacks on nonce-based authenticated encryption schemes. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Singapore, 6–10 December 2016; Springer: Berlin/Heidelberg, Germany, 2016; Volume 10031, pp. 369–395. [CrossRef]
74. Andreeva, E.; Bogdanov, A.; Luykx, A.; Mennink, B.; Tischhauser, E.; Yasuda, K. AES-COPA v1. 2014. Available online: https://competitions.cr.yp.to/round1/aescopav1.pdf (accessed on 9 December 2022)
75. Dobraunig, C.; Eichlseder, M.; Korak, T.; Mangard, S.; Mendel, F.; Primas, R. SIFA: Exploiting ineffective fault inductions on symmetric cryptography. *IACR Trans. Cryptogr. Hardw. Embed. Syst.* **2018**, *2018*, 547–572. [CrossRef]
76. Joshi, P.; Mazumdar, B. Single Event Transient Fault Analysis of ELEPHANT Cipher. 2021. pp. 1–5. Available online: http://arxiv.org/abs/2106.09536 (accessed on 8 December 2022).
77. Ramezanpour, K.; Ampadu, P.; Diehl, W. FIMA: Fault Intensity Map Analysis. In *Constructive Side-Channel Analysis and Secure Design*; Lecture Notes in Computer Science; Polian, I., Stöttinger, M., Eds.; Springer: Cham, Switzerland, 2019; Volume 11421. [CrossRef]
78. Ramezanpour, K.; Ampadu, P.; Diehl, W. Fault intensity map analysis with neural network key distinguisher. *J. Cryptogr. Eng.* **2021**, *11*, 273–288. [CrossRef]
79. Ramezanpour, K.; Ampadu, P.; Diehl, W. A statistical fault analysis methodology for the Ascon authenticated cipher. In Proceedings of the 2019 IEEE International Symposium on Hardware Oriented Security and Trust (HOST), McLean, VA, USA, 5–10 May 2019; pp. 41–50. [CrossRef]
80. Joshi, P.; Mazumdar, B. SSFA: Subset fault analysis of ASCON-128 authenticated cipher. *Microelectron. Reliab.* **2021**, *123*, 114155 [CrossRef]
81. Sarkar, S.; Banik, S.; Maitra, S. Differential fault attack against grain family with very few faults and minimal assumptions. *IEEE Trans. Comput.* **2015**, *64*, 1647–1657. [CrossRef]