JOURNAL OF CRITICAL REVIEWS

ISSN-2394-5125 VOL 7, ISSUE 10, 2020

ARCHITECTURE OPTIMIZATION AND PERFORMANCE COMPARISON OF NONCE-MISUSE-RESISTANT AUTHENTICATED ENCRYPTION ALGORITHMS

[1] K.MAHESH,[2] G.VASANTHI,[3]G P.PRADEEP KUMAR

[123]Assistant Professor

Department of ECE

Dr. K V Subba Reddy Institute Of Technology

ABSTRACT:

This study compares the performance of new authenticated encryption (AE) algorithms with those of current standards in order to provide enhanced security and resource efficiency. These algorithms provide a crucial characteristic known as nonce-misuse resistance, which enhances the security of current AE standards. In addition to a proposal from the Crypto Forum Research Group, this document discusses algorithm to architecture mappings of a number of contenders from the current Competition for AE: Security, Applicability, and Robustness.

The design of a well-known standard, the Advanced Encryption Standard in Galois Counter mode (AES-GCM), is contrasted with implementations of the architectures on platforms for both field-programmable gate arrays and application-specific integrated circuits. The provided optimisations are relevant to AE generally and nonce-misuse-resistant designs specifically. Additionally, a codesign strategy for hardware and software is addressed. The implementations made possible by the suggested optimisations show that new AE algorithms may provide speed on par with that of AES-GCM while boosting security and resource efficiency for certain use-case situations.

Index Terms—Authenticated encryption (AE), Competition for AE: Security, Applicability, and Robustness (CAESAR), Deoxys, nonce-misuse resistance, pipelineable on-line encryption with authentication tag (POET), PRIMATE-APE. Advanced Encryption Standard in Galois Counter mode (AES-GCM), AES-GCM-synthetic IV (SIV), and authenticated encryption (AE).

I. INTRODUCTION

WITH the advent of the Internet of Things (IoT) era, billions of devices will be connected to each other and to a common network. Hence, it is of utmost importance to ensure security and privacy of communication between the devices as well as between the device and the cloud server. Moreover, there is a growing trend to bring computing to the edge of the IoT rather than the cloud, termed as edgecentric computing [1]. Hence, resource efficient and strongly secure cryptographic algorithms which can ensure the security of communication and can be implemented on the hardware of the device itself have become critical. Also, algorithms that require smaller key sizes, less frequent change of keys, and better resilience are desired.

Authenticated encryption (AE) algorithms combine the process of authentication and encryption to create a single algorithm which is secure and resource efficient. It is well understood that confidentiality of data does not suffice, and it is important to ensure authentication of source as well as data integrity [2]. Hence, AE algorithms are growing in importance with the changing device scenarios and platforms. When data such as packet headers and message numbers are also included as part of the plaintext, the resulting algorithms are termed AE with associated data (AEAD) algorithms. These additional data require only authentication and are termed associated data. The general equations of AEAD are expressed as follows:

$$\mathrm{Enc}_K(N, \mathrm{AD}, \mathrm{PT}) = (\mathrm{CT}, \mathrm{Tag}) \quad (1)$$

$$\mathrm{Dec}_K(N, \mathrm{AD}, \mathrm{CT}, \mathrm{Tag}) = (\mathrm{PT}, \perp). \quad (2)$$

In these equations, K represents the secret key, AD represents the associated data, PT refers to the plaintext, and N is a unique nonrepeating number termed nonce. These inputs are applied to the encryption algorithm Enc to produce the outputs: CT, which represents the encryption of the plaintext, and Tag, which are utilized for authentication purposes. The nonce is used to transmit more than one message block using the same secret key. For the decryption algorithm Dec, N, AD, CT, Tag, and K are used as inputs to retrieve PT. In addition, Tag is verified and a valid/invalid message is generated (represented as ⊥ in the equation).

Some of the existing standards for AEAD algorithms include Advanced Encryption Standard in Galois Counter mode (AES-GCM), Advanced Encryption Standard in Counter with cipher block chaining message authentication code mode, encrypt-then-authenticate-then-translate, Offset Codebook (OCB) mode, etc. We do not delve into the details of these algorithms and refer the reader to the existing literature [3]–[6]. Instead, we focus on AES-GCM which has been deployed in most practical applications such as Transport Layer Security (TLS) and Secure Socket Layer. Using AESGCM as a baseline architecture, we present the architectures of several new algorithms which have been proposed as a part of the ongoing Competition for AE: Security, Applicability, and Robustness (CAESAR) competition as well as from Crypto Forum Research Group (CFRG). The chosen algorithms incorporate an important property termed nonce-misuse-resistance.

II.RELATED WORK

In this section, we discuss the shortcomings of current AEAD algorithms and motivation for this paper.

A. New AEAD Schemes

Several issues have been identified in existing AE algorithms. Some of these are highlighted below.

1) Existing algorithms are still too large in terms of area or consume too much energy. This is especially true if the AE schemes are to be implemented on a devicewhich has minimal resources.

2) Several security properties such as nonce-misuse resistance, decryption misuse resistance, robustness against leakage of plaintext, detection of forgery attempts, and so on are desirable and missing in existing AEAD algorithms.

3) Cryptanalytical efforts have found groups of weak keys in the most widely adopted AES-GCM algorithm [9].

4) Better performance while maintaining the same level of security of existing standards or better security with the same performance are both desirable. This is especially true due to increase in number and reduction in the size of devices in modern applications. To alleviate some of these problems, a call for novel authenticated algorithms was put forth in the form of CAESAR competition. The goal of this competition is to identify a portfolio of AE algorithms which offer advantages over AES-GCM and are suitable for widespread adoption [10].

The competition is ongoing and currently in its final round with seven finalists. The first round had 54 submissions out of which 29 candidates were selected for the second round. In the third round, 15 potential candidates were recognized. These candidates have different properties with respect to security, resource consumption, and underlying constructions. A summary for the candidates and their important properties can be found in [11] and [12]. A candidate that we discuss in this paper, Deoxys, has been selected to the final round of the competition. Note that even though some of the candidates selected for this paper have not been selected for the final round of the CAESAR competition, these are useful candidates for several applications. Since the competition considers several parameters apart from security, these candidates have been excluded from the next round. We also consider an algorithm submitted to CFRG after the CAESAR competition first round submission deadline.

Several independent implementations of the algorithms in both hardware and software can be found in the literature. To bring these implementations under the same platform, there is an ongoing effort being carried out by the SUPERCOP software benchmarking team and the ATHENa GMU hardware platform team [13]. Our work specifically focuses on in-depth analysis of nonce-misuse resistance schemes whose details and significance will be discussed next.

B. Importance of Nonce-Misuse Resistance

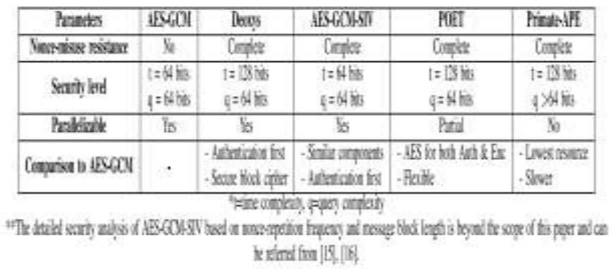
From the description of AEAD schemes, it is observed that nonces are critical to the security of symmetric key cryptographic algorithms. The construction of these algorithms is such that using the same key, multiple messages can be encrypted by using different nonces. Hence, the nonce must not repeat under the same key. Even though this appears to be a simple requirement, it is not easy to satisfy as has been observed in many practical applications [14].

There are several approaches that can be used to ensure nonces do not repeat while using the same key. One solution is to update the keys frequently. However, regular exchange of secret keys between two parties is not an easy task and many

ISSN-2394-5125 VOL 7, ISSUE 10, 2020

applications will not have the capability to do so. Specifically, with systems involving IoT devices where millions of devices are interconnected with each other, the most practical implementation would program one secret key and utilize it for the lifetime of the device. The second approach to ensure nonces do not repeat is to derive the nonces from counters. With sufficiently large counters, the nonce values will be based on each increment of the counter and will not repeat. However, if the counter is made to overflow by injecting faults or other forms of attacks, the nonces start to repeat breaking the security of the algorithm. Finally, the nonce can

TABLE I SUMMARY OF CANDIDATE FEATURES AND COMPARISON WITH AES-GCM



Parameters	AES-GCM	Deoxys	AES-GCM-SIV	POET	Primate-APE
Nonce-misuse resistance	No	Complete	Complete	Complete	Complete
Security level	t = 64 bits q = 64 bits	t = 128 bits q = 64 bits	t = 64 bits q = 64 bits	t = 128 bits q = 64 bits	t = 128 bits q >64 bits
Parallelizable	Yes	Yes	Yes	Partial	No
Comparison to AES-GCM	-	- Authentication first - Secure block cipher	- Similar components - Authentication first	- AES for both Auth & Enc - Flexible	- Lowest resource - Slower

*t=time complexity, q=query complexity

**The detailed security analysis of AES-GCM-SIV based on nonce-repetition frequency and message block length is beyond the scope of this paper and can be referred from [15], [16].

be made unique by using random number generators. However, the random number generator should be of high quality and sufficiently large. Ensuring this requirement is met again is a challenging task with the ever-shrinking sizes of devices. Attacks due to nonce-misuse have been demonstrated in the literature in important applications such as the TLS [14]. Thus, algorithms which can inherently provide some form of noncemisuse resistance have become favorable and will continue to increase in importance as IoT devices become more prevalent.

III. ALGORITHM AND ARCHITECTURAL DESCRIPTIONS

Next, we briefly discuss the algorithm and architecture of AES-GCM standard. The discussions of nonce-misuseresistant algorithms and architectures will be based on this standard.

A. AES-GCM

The AES-GCM algorithm is represented using the block diagram of Fig. 1. This algorithm is a block cipher-based

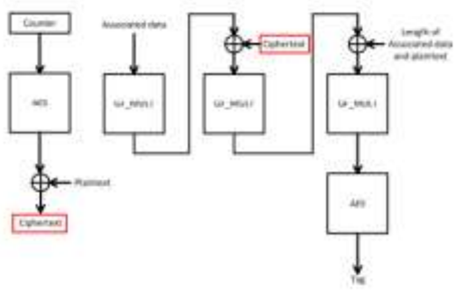


Fig. 1. Description of the algorithm of AES-GCM. Left: cipher text block generation using a counter. Right: associated data processing and processing of plaintext blocks to generate the tag.

AE scheme. This implies that a block cipher such as AES is used to perform all encryption operations. AES in the counter (CTR) mode of operation handles plaintext in a block by block manner to generate cipher text blocks. The first value of the counter is dependent on the nonce, and the counter is incremented to process subsequent message blocks. The authentication process is handled by a Galois field multiplier which performs polynomial multiplication on the associated data and the generated cipher text blocks. A final encryption results in the generation of the tag value. Note that the counter value is encrypted using the AES block and the plaintext is just XORed with the output. This implies that the security of the algorithm is completely dependent on the nonce being different for each message. If the same nonce repeats for two different messages, a simple XOR operation can differentiate between them. Thus, information is leaked, and the system is not nonce-misuse resistant.

A serial implementation aimed at low area and power consumption is used to design the architecture of AES-GCM. This means that the architecture makes use of a single AES block and Galois field multiplier block. The blocks of the data path of AES-GCM are illustrated in Fig. 2. The plaintext blocks are processed serially and require correct control to direct data in and out of the blocks. All architectures created in this paper follow the serial implementations. Based on the different properties of the architecture, optimizations are then added on top of the serial architecture. This is discussed in Section IV.

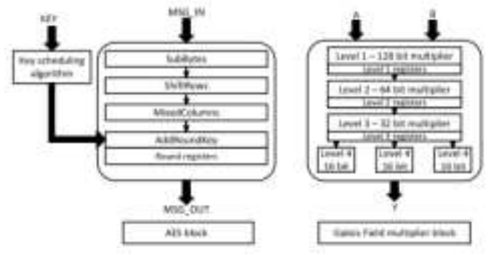


Fig. 2. Data path of AES-GCM consisting of AES block and Galois fiel multiplier block.

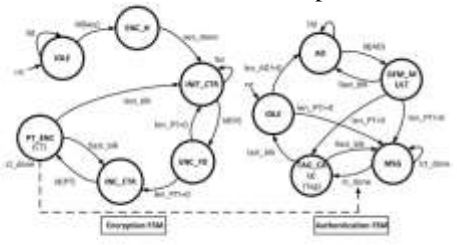


Fig. 3. FSMs of AES-GCM for encryption and authentication in a blockinterleaved manner. Signal ct_done is used for synchronization between the two FSMs.

associated data blocks and waits for the ct_done signal. Upon receiving the signal, the processing of cipher text blocks is performed in an interleaved manner with the encryption process to generate the tag value. The ability to parallelly process multiple message blocks is an important property of the AES-GCM algorithm since it results in a good performance and resource utilization.

B. Deoxys

Deoxys is a block-cipher-based AE algorithm utilizing a tweakable block cipher, Deoxys-BC. The tweakable block cipher is based on AES but uses a key and a tweak value. It has more rounds than standard AES, claiming better security. The Deoxys algorithm has two modes: one for which nonce must not be reused and one which is nonce-misuse resistant. The nonce-misuse-resistant version of Deoxys provides full 128-bit security for unique nonces and birthday bound security when nonce is reused. Existing hardware and software implementation also shows that it is suitable for short messages having low precomputation overhead.

The basic steps of Deoxys authentication and encryption are described in Fig. 4. Since the block cipher in Deoxys is a custom built tweakable block cipher, the inputs include an additional tweak value which needs to be provided each time the encryption block is called. From the block diagram, we observe that the associated data blocks are processed first followed by the plaintext blocks. Finally, incorporating the nonce value into the tweak value, the tag is generated. Note that, for the encryption process, the nonce value is used as input to the block cipher. The tag value is incorporated into the tweak value. Thus, there is a dependence between the tag generation and cipher-text generation in Deoxys. This will lead

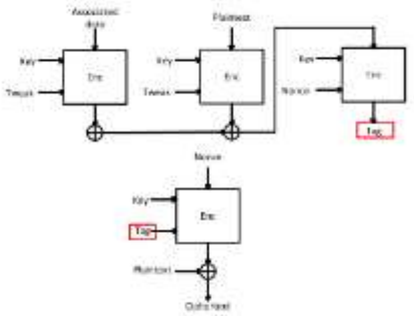


Fig. 4. Description of the algorithm of Deoxys. The encryption block in this figure is the tweakable Deoxys block cipher with a key and tweak as inputs. Top: tag generation. The generated tag is used for processing of plaintext blocks to generate cipher text blocks.

to important differences in the architectural implementation and optimizations.

IV. OPTIMIZATIONS

In this section, we discuss some of the optimizations that can be applied to the discussed AEAD schemes. These optimizations are either targeted to utilize the properties of the FPGA platforms or the properties of the algorithm itself. In Section V, we present the results of optimization on the algorithms and discuss which of the optimizations are most suitable for each algorithm.

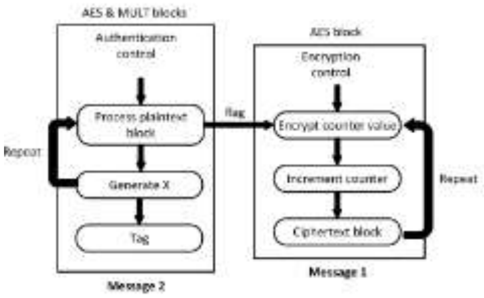


Fig. 12. Parallel processing of messages in AES-GCM-SIV.

A. Parallel Processing of Messages

From the architecture of AES-GCM, we have observed that it is inherently parallel in nature. This means that when a cipher text block is being

ISSN-2394-5125 VOL 7, ISSUE 10, 2020

generated by the encryption algorithm, parallel processing of data by the authentication algorithm can occur.This is an important property which results in low cycles per byte for AES-GCM. However, by modifications, such as pipelining and parallelism, this property can be utilized to optimize other algorithms.

AES-GCM-SIV utilizes AES and multiplier blocks for encryption and authentication operations. Hence, similar to AES-GCM, the two operations can be separated and performed in parallel. However, in AES-GCM-SIV, authentication occurs first and there is a data dependence between authentication and encryption. This means that first the authentication process must be performed completely to generate the tag and then the cipher text blocks can be generated through encryption. To break this dependence, we can process two messages in parallel and pipeline the design such that the two processes occur using two different FSMs. This concept is illustrated in Fig. 12. Synchronization is necessary for the parallel processing of messages. This can be achieved by using flags as indicated in the figure. After the associated data blocks of the second message are processed, the processing of plaintext blocks using the multiplier begins. At this time, the processing of plaintext blocks of message 1 can be carried out by the second state machine. Note that the tag of the first message needs to be stored for the encryption block to access during processing of message 2.

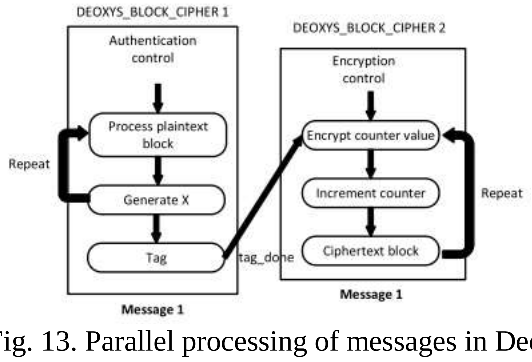


Fig. 13. Parallel processing of messages in Deoxys.

processing of messages but will also increase the resource consumption because of duplication.

B. Implementation of S-Box in Memory

The discussed algorithms utilize lookup tables (LUTs) in the form of S-Boxes of block ciphers. Each S-Box maps 1 byte of a message to 1 byte of the substitution value. Since each message block is of size 16 bytes (128 bits), 16 such S-Boxes are necessary to process a complete message block. These LUTs can be implemented using a straightforward implementation where the LUTs are synthesized using pure combinational logic elements. However, we note that most modern-day device platforms such as FPGAs and microcontrollers have some form of memory available on board. This memory can be used to port the LUTs of S-Boxes. This results in reduction of logic elements and utilization of the available memory blocks of the device platform.

V. RESULTS

In this section, we describe the experimental setup adopted for all implementations and discuss the corresponding results obtained after applying optimizations discussed in Section IV.

Results are reported both in terms of FPGA and ASIC implementations wherever applicable. This ensures platform obliviousness while making the final comparisons between different algorithms. Note that the described results mainly compare the encryption operation of the AEAD algorithms.

Comparison of the decryption or a combination of both is scope for future work.

A. Experimental Setup

All results on the FPGA platform are obtained using Altera’s Cyclone V family of FPGA which are built on TSMC’s 28-nm low-power process technology. Specifically, Altera Cyclone 5CSEMA4U23C6 incorporated in an ATLAS SoC board is utilized. The Cyclone V family of FPGAs allows for low-area, low-cost implementations of algorithms. The implementations are written in Verilog Hardware Descriptive Language and simulated using the ModelSim tool. Synthesis is performed using the Altera Quartus Prime tool and timing measurements are performed using the TimeQuest timing analyzer. While running synthesis, area optimization is enabled, and a target frequency of 50 MHz is used. Power measurements are performed using the PowerPlay power analyzer tool.

For every experiment performed, the following measurements are reported.

1) The area consumption is reported both in terms of LUTs, which are the basic combinational elements of FPGAs, as well as register count. The resource utilization of the cipher as a percentage of the total adaptive logic modules (ALMs) and registers of the FPGA is also presented.

2) The output of power play power analyzer tool is reported in terms of power in milliwatt. Before

running the tool, appropriate inputs in terms of value change dump files of the simulation of modules is provided. Only dynamic power is considered since the leakage power reported by the tool is for the entire fabric of the FPGA and does not correctly represent the leakage power of the design. Details of the generation of power values can be referenced from [7].

3) We report the time of operation in terms of cycles per byte which is defined as the number of cycles required to process each byte of the message. This measure is independent of the FPGA platform and is an important measure of the performance. It is calculated using the following equation:

$$\text{Cycles\_per\_byte} = \frac{\text{Cycles\_per\_msg}}{\text{Size\_of\_msg\_in\_bytes}}. \qquad (4)$$

The value for the number of cycles is obtained through simulation.

B. Implementation 1: Direct Implementation of the Algorithms Using Combinational Logic Only

The architectural descriptions provided in Section III are directly mapped to hardware. This results in implementations which are unoptimized but provide a good first comparison between all the algorithms. The results are tabulated in Table III for a clock speed of 50 MHz. Note that ASIC implementations are also included in the same table for comparison.

From the results, we observe that in terms of area,

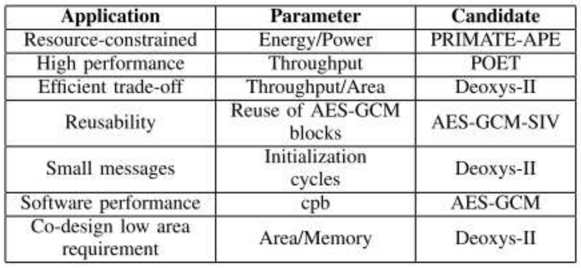
PRIMATE-APE is the smallest. This can be attributed to the fact that the PRIMATEs are sponge-based designs and, hence, are expected to be lightweight. The next smallest architecture with respect to area is the Deoxys algorithm. With the modifications of the algorithm as applied to version v 1.4.1, the area of the algorithm has reduced further. Both versions of Deoxys have the advantage of utilizing just one block cipher for both authentication and encryption processes. This avoids multiplication which is a resource-intensive operation. AES-GCM-SIV and POET are slightly more resource consuming in terms of area compared to AES-GCM. The power consumption trends are almost similar to the area consumption trends with PRIMATE and Deoxys having the least FPGA power consumption values.

VII. CONCLUSION

We have shown the usage of both FPGA and ASIC platforms to develop a number of nonce-misuse-resistant algorithms in this work. Comparing the options to the AES-GCM method, all of them exhibit noncemisuse resistance. This is due to the fact that in AES-GCM, the AES algorithm in CTR mode employed the nonce directly. Only XORing the plaintext produced this outcome. As a result, every repeat of the nonce revealed the plaintext. The tag, which varies even if the nonce is repeated, determines the cypher text in both Deoxys and AES-GCM-SIV. The message block is provided as an input to the block cipher/sponge block in POET and PRIMATE, eliminating the cypher text's direct reliance on the nonce. As a result, all of these algorithms are nonce-misuse resistant when compared to AES-GCM.

The significant architectural changes brought about by the addition of nonce-misuse resistance were explored, and solutions to the problems were also offered. We have suggested the individuals most suited for each application scenario as shown in Table VIII based on the findings from Section V. Future work in this field will look at how to better optimise these algorithms' architectures. This study gave a preliminary description of side-channel analysis. Future study should focus on side-channel assaults that are experimental and countermeasure suggestions.

TABLE VIII RECOMMENDATIONS OF CANDIDATES BASED ON OBSERVED RESULTS



Application	Parameter	Candidate
Resource-constrained	Energy/Power	PRIMATE-APE
High performance	Throughput	POET
Efficient trade-off	Throughput/Area	Deoxys-II
Reusability	Reuse of AES-GCM blocks	AES-GCM-SIV
Small messages	Initialization cycles	Deoxys-II
Software performance	cpb	AES-GCM
Co-design low area requirement	Area/Memory	Deoxys-II

REFERENCES

[1] P. G. Lopez et al., "Edge-centric computing: Vision and challenges,"ACM SIGCOMM Comput. Commun. Rev., vol. 45, no. 5, pp. 37–42, Oct. 2015.

[2] M. Bellare and C. Namprempre, "Authenticated encryption: Relations among notions and analysis of the generic composition paradigm," in Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur. Berlin, Germany: Springer, 2000, pp. 531–545.

[3] J. Salowey, A. Choudhury, and D. McGrew, AES Galois Counter Mode (GCM) Cipher Suites for TLS, document RFC 5288, 2008.

[4] D. McGrew and D. Bailey, AES-CCM Cipher Suites for Transport Layer Security (TLS), document RFC 6655, 2012.

ISSN-2394-5125 VOL 7, ISSUE 10, 2020

[5] M. Bellare, P. Rogaway, and D. Wagner, "The EAX mode of operation," in Proc. Int. Workshop Fast Softw. Encryption. Berlin, Germany: Springer, 2004, pp. 389–407.
[6] P. Rogaway, M. Bellare, and J. Black, "OCB: A block-cipher mode of operation for efficient authenticated encryption," ACM Trans. Inf. Syst. Secur., vol. 6, no. 3, pp. 365–403, Aug. 2003.
[7] S. Koteshwara, A. Das, and K. K. Parhi, "FPGA implementation and comparison of AES-GCM and Deoxys authenticated encryption schemes," in Proc. IEEE Int. Symp. Circuits Syst. (ISCAS), May 2017, pp. 1–4.
[8] S. Koteshwara, A. Das, and K. K. Parhi, "Performance comparison of AES-GCM-SIV and AES-GCM algorithms for authenticated encryption on FPGA platforms," in Proc. Asilomar Conf. Signals, Syst. Comput., Oct. 2017, pp. 1331–1336.
[9] H. Handschuh and B. Preneel, "Key-recovery attacks on universal hash function based MAC algorithms," in Proc. Annu. Int. Cryptol. Conf. Berlin, Germany: Springer, 2008, pp. 144–161.
[10] D. J. Bernstein. (2014). CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness. [Online]. Available: http://competitions.cr.yp.to/caesar.html.