SEAA: Secure and Efficient Authenticated Encryption Algorithm for Satellite Application

Abid Murtaza, S. Jahanzeb Hussain Pirzada, Tongge Xu, and Liu Jianwei

School of Cyber Science and Technology, Beihang University (BUAA), Beijing, China

Summary
Satellite communication is a fundamental part of modern-day communication infrastructure. Hence the information security is also critical in satellite communication. Authenticated Encryption (AE) algorithm provides confidentiality, integrity, and authenticity of data, all primary data security services with a single algorithm. The AES-GCM algorithm is the most famous and widely used AE algorithm, which is also recommended to be used in satellite application. However, the AES-GCM architecture is not nonce misuse attack resistant. Also, most of the existing AE algorithms (including GCM) are based on block cipher design, which is vulnerable to side-channel attacks due to using the same key for a longer duration. Re-keying is an effective approach to protect the algorithm against side-channel attack. On the other hand, satellites have limited computational capacity, high throughput requirements, and higher latency attributes. Therefore, there is a need for the AE algorithm, which is not only secure against known attacks but also fast and capable of supporting high throughput requirements of satellite communication. In this paper, we have proposed an AE algorithm architecture named SEAA (Secure and Efficient Authenticated encryption Algorithm), in which the re-keying approach is incorporated such that, the SEAA becomes resistant side-channel and nonce misuse attacks at the same time without any additional protection or overhead. Additionally, the proposed algorithm also reduces the computational complexity of the traditional AES-GCM algorithm to increase the efficiency of the proposed AE algorithm. The experimental results of the proposed algorithm in software and hardware (FPGA) show that the computational performance (speed and throughput) of the proposed algorithm are better than the famous AES-GCM, despite being secure against known attack applicable to AES-GCM. Hence, SEAA is a suitable AE algorithm for satellite application.

Key words:
Authenticated Encryption, Nonce misuse, Side-channel Attack, Algorithm, AES-GCM

1. Introduction

From With the rapid growth of information processing and communication technologies, data exchange also increased radically in recent decades. This has also resulted in the ever-increasing demand for ensuring information security. The security of the modern-day communication system is based on the CIA-AAA model (Confidentiality, Integrity, Availability, Authentication, Authorization, and Accounting. The former two are related to data protection and are focus of our work, while the latter four are related to the user's authentication and access to system resources. For data protection, confidentiality ensures that apart from the intended receiver, nobody could get into the content of a message. While data origin authenticity and integrity services ensure that the data is received only from a particular sender, and is not forged in the transmission path. Confidentiality is achieved by encrypting the message with an encryption algorithm (Symmetric-key or asymmetric key Algorithm). Data authenticity and integrity services are together provided through data authentication algorithms, which are either hash-based or cipher based in case of symmetric key algorithms, or digital signature-based in case of public-key cryptography.

Satellites are a fundamental part of modern-day communication infrastructure, providing communication, navigation, remote sensing, earth observation, weather monitoring, and other services for civil, commercial and defense purposes. The security of satellite communication is critical, and compromise in security could have serious problems. Therefore, the Consultative Committee for Space Data System (CCSDS) recommends using security algorithm and protocols for securing satellite communication according to the security classification of a space mission as either high, moderate, or minimum security missions [1]. On the other hand, satellites also have the limited computational capacity and high throughput requirements (for communicating extensive data, e.g., stored images); hence, the security solutions for satellite communication, should be fast and lightweight. Understandably encryption without data origin and content authentication is not secure, in particular in satellite application [2]. Therefore, both encryption and data authentication services are used to secure satellite communication. Authenticated Encryption (AE) algorithm provides all these encryption, origin of data authentication, and data integrity services altogether by a single algorithm. Among the existing AE algorithms, the most famous AE algorithm is Advanced Encryption Standard, Galois Counter Mode (AES-GCM), which was designed by McGrew and Viega and standardized by the National Institute of Standards and Technology (NIST) and International Organization for Standardization (ISO) [3], [4]. The CCSDS also recommends using AES-GCM when

Manuscript received December 5, 2019
Manuscript revised December 20, 2019

encryption in combination with data integrity and origin authentication is required in space missions [5]. The AES-GCM utilizes AES in counter mode (AES-CTR) encryption, while polynomial multiplication on Associated Data (AD) is performed in the Galois field multiplier for authentication tag generation. Therefore, GCM can provide high-speed AE in both software and hardware [6], [7], as well as it can be parallelized and pipelined [8], methods that can be very advantageous in the space community [5].

Despite its computational efficiency, AES-GCM is vulnerable against nonce misuse attack. In a nonce misuse attack, if the same nonce is used for two different messages, the information could be leaked. The AES-GCM algorithm uses a simple exclusive or (XOR) operation on plaintext for the generation of ciphertext. So, in case of the use of the same nonce for different messages, a simple XOR can differentiate between the messages. The criticality of nonce misuse attack in practical applications such as Transport Layer Security (TLS) has already been demonstrated [9]. Moreover, GCM is also vulnerable against side-channel attacks, in particular, Differential Power Analysis (DPA) attack, because of using the same key for longer duration in block cipher (AES) [10]. In side-channel attacks, information about the key is extracted by observing the physical properties of the device such as power utilization or the electromagnetic field [11], which could lead to a successful recovery of the key. The recent demonstration of these attacks is presented in [12] and [13].

Apart from AES GCM, many new lightweight AE algorithms have been proposed by researchers in Competition for Authenticated Encryption: Security, applicability, and reliability (CAESAR) [10] and otherwise in recent years, to comply with the demands for lightweight AE algorithm for resource-constrained devices and applications, such as the Internet of Things (IoT). However, most of the proposed AE schemes have some computational limitations or security concerns. For example, many of the proposed schemes are not complete nonce-misuse resistant [14] and also have vulnerabilities against side-channel attacks [15]. On the other hand, the schemes which are secure against nonce-misuse attack (such as Primate [16]) are slow by design and not suitable for large data as in case of satellite application (images) [17]. Also, most of the AE schemes are based on nonce, which is additional overhead data and processing costs, which is critical for satellite application.

Therefore, there still a need for a lightweight AE algorithm that is not only capable of parallel processing to support high data rates but also secure against both active nonce misuse attack as well as passive side-channel attacks. This paper proposed such an algorithm named SEAA, where not only the security vulnerabilities of existing AE schemes have been addressed, but also, the computations of the algorithm have been simplified for improved computational performance.

The remaining of this article is arranged as follows: In section 2, related work has been reviewed. In section 3, we have proposed our algorithm with details. Section 4 shows the implementation results. Section 5 provides a security analysis of the proposed algorithm. Section 6 concludes the paper.

2. Related Work

AES-GCM algorithm is the most famous AE algorithm and used in many modern-day security protocols (such as IEEE 802.1AE, IETF IPSec, SSH, and TLS/SSL, etc.) [18], because of being computationally efficient due to parallel in architecture without any optimization. However, there have been many optimized and efficient implementations of AES-GCM on FPGA hardware [19],[20],[21]. However, due to the vulnerability of GCM against nonce-misuse attack, the GCM-SIV variant of the GCM algorithm has been proposed, which provides complete nonce misuse resistance [22]. However, firstly in GCM-SIV, plaintext needs to be processed twice, which decreases its computational performance than AES-GCM [23]. Secondly, both GCM and GCM-SIV are vulnerable to side-channel attacks.

CAESAR was started in 2012 for the selection of the most suitable AE algorithm for different categories, including the lightweight category. 57 AE schemes were presented to this competition, and six were declared as winners. For the lightweight application category (resource-constrained case), 2 AE schemes were announced as winners, i.e., Ascon [24] and ACORN [25]. There are several studies on the proposed AE schemes in CAESAR to evaluate the performance comparison of different proposed AE schemes of the competition [14], [26]. The computational performance of both the CAESAR winners, Ascon and ACORN, is better and at a lower cost than AES GCM [27]. However, both the winners of CAESAR (Ascon and ACORN) are not completely nonce-misuse resistant [14]. Similarly, a study shows that many of the proposed AE algorithms, including Ascon and ACORN, have significant Information leakage in the 186 partan-6 FPGA and are likely vulnerable to DPA [13].

There are two approaches to the countermeasure of these passive attacks. First is the conventional method of hiding or masking the implementation of the cryptographic algorithm [28]. But the problem of these methods is high overhead, area, and cost of protection [29], which are critical in satellite application. Although, there are schemes such as COFB whose objective was to reduce the size of the mask without compromising security [30]. The alternate solution is fresh re-keying [31], which means obtaining a new session key every time. This approach can provide inherent protection against the DPA attack. This concept is similar to the idea of Forward Secrecy (FS), which uses a

new key for every new message. This FS concept is used in many applications nowadays because of its benefits such as with re-keying or FS, in case of a compromise of a key, only a message could be leaked while all the communication stays at risk otherwise [32]. The new keys can be achieved either derivation from the master key or some lightweight protocols that can be used for the exchange of new keys [33]. Most of the existing and proposed AE schemes are not designed for this re-keying or FS approach.

From this brief review of related work, we can see that; there is a need for nonce misuse resistant as well as a side-channel attack resistant AE algorithm for satellite application, which is also fast and capable of supporting high data rate support for satellite application. In this paper, we will show that incorporating re-keying or FS concept inside the algorithm design can produce an efficient AE algorithm, which is inherently side-channel attack resistant, as well as nonce misuse resistant. Furthermore, the re-keying approach allows simplifying the computation of block cipher used in the AE algorithm. Therefore, implementation results and performance of the proposed AE algorithm validates the computational superiority over the existing AE algorithm, such as AES-GCM.

3. Proposed Algorithm

Before the description of our algorithm, it is helpful to explain the design concept of the proposed algorithm.

3.1 Proposed approach

We can note from the AES-CTR part of AES-GCM that the ciphertext is generated by simple XOR operation of plaintext with the 128-bit output of AES block (say Temporary variable Ti). Here Ti is widely different from Ti-1, even though the same key and same algorithm are used, and the counter value of both is different by only 1 (increment operation). This is because of AES round functions, which ensure the widely different output even if the input is changed only by 1 bit, and the same key is used. But to note here is that in essence, the encryption process of AES-CTR is similar to One-time Pad (OTP), where the plaintext is simply XOR with the same size key, which is never used again.

Our design principle is that instead of XORing the plaintext Pi with Ti for ciphertext generation, we propose to XOR two new keys called key1, key2, with Ti for the ciphertext generation. This approach can provide some unique benefits as follows. Firstly, the total reliance on AES block cipher for the generation of unique Ti to eventually ensure unique ciphertext for the same plaintext can be significantly reduced. Because, even if using the same plaintext & corresponding Ti, the new keys key1 and key2, can provide the uniqueness in ciphertext for the same plaintext & Ti. Hence, the block cipher function can be simplified or reduced. Secondly, the use of new keys (key1 and key2) for every message block will inherently prevent our algorithm from side-channel attacks such a DPA. Thirdly, in the conventional encryption or AE algorithm approach, a single key is used with a different nonce to ensure unique ciphertext in case of the same plaintext. If we can ensure that apart from plaintext (or Ti), both the XORing numbers (i.e., key1 and key2) are always new, unique, and random, then this will ensure a unique ciphertext for the same plaintext and same Ti. Hence, the proposed algorithm SEAA will not be vulnerable to nonce misuse attack. Finally, it will provide security similar to the provably secure OTP. Because, due to our design of three variables for ciphertext generation (Ti, key1 and key2), compromise of any one variable (e.g., key1 or key2) will still maintain the confidentiality of the plaintext because of the other two variables. Mathematically, if N numbers of n bits each are XORed with each other, then there exist 2(N-1) n possible unique permutations, which can produce the same ciphertext. So, if the length of three numbers is 128 bits each, then there will be 2(3-1)128 = 2256 permutations of three numbers that can produce the same ciphertext. So, even if one number is known to the attacker, there still exist 2128 possible combinations of the other two numbers. As the attacker has no further information, he can never be sure which combination among 2128 is correct, which is a similar situation to OTP.

Based on the above description of the design principle, we can see that one of the main task of the proposed algorithm is to ensure new keys (key1 and key2) for every plaintext block, such that keys (key1 and key2) will never be used again, at least together in case of the same plaintext.

Now, we will explain the proposed algorithm SEAA in detail, which is shown in figure 1. As a pre-requirement, we suggested that the sender and receiver need to have shared a secret random key of 256 bits, whose first 128 bits will be called master key (K0) and the last 128 bits will be called initialization vector (IV), while the message is divided into 128-bit blocks. In figure 1, we can see that plaintext blocks are processed in the modified AES block to provide Ti. Then these Ti are XORed with corresponding Ki and IV_i to produce Ci. While P1 to Pn and XORed together. The resultant is processed in the modified AES block to generate the authentication tag by XORing with Kn+1 finally.

There are four main stages of SEAA; Key Expansion Algorithm, Modified AES for Ti generation, the Encryption process, and finally, the Tag generation.

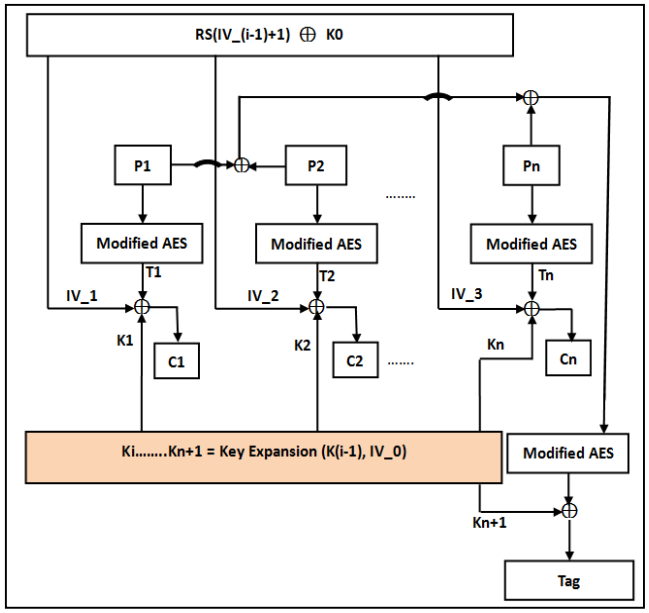

Fig. 1 Proposed AE Algorithm

3.2. Encryption Process

The encryption process of the SEAA is as simple as XOR of two keys IV_i and Ki with Ti, as shown in equation one below.

$$Ci = Ti \oplus IV\_i \oplus Ki \quad (1)$$

Where 'i' is block number, C is ciphertext, P is plaintext. Here Ti is generated through modified AES block step, Ki is generated through Key expansion algorithm, and IV_i is obtained by increment and right shift operation on IV_i-1 (as proposed for AES-GCM by A. S. Bader et al. [32]) and then XORing it with K0 as shown in equation 2 below.

$$IV\_i = RS((IV\_(i-1)) + 1) \oplus K0 \quad (2)$$

The right shift and increment operation on IV_(i-1) provides a new key every time. Hence, the generated ciphertext is always largely changed. Also, after the generation of IV_i, this IV_i becomes IV_(i-1) for the generation of next IV_i. Through this method, IV is continuously updated and never used again. Similarly, the last IV, i.e., IV_n of one message, is considered as IV_0 for the next message, this will provide the desired, never used again characteristic and hence side-channel attack resistance.

3.3. Key Expansion Algorithm

As we know that, we need to ensure the availability of new key for every message block. Besides, we also need one new key for tag generation. Therefore, if there are n message blocks, there is a need for n+1 keys (K1 to Kn+1).

In our other recent work [33], we have proposed a modification in AES key expansion algorithm, which can produce 128 distinct keys from 1 main key. We can use a slightly modified version of that key expansion algorithm for the generation of unlimited unique keys. The flow diagram of the modified key expansion algorithm is shown in figure 2.

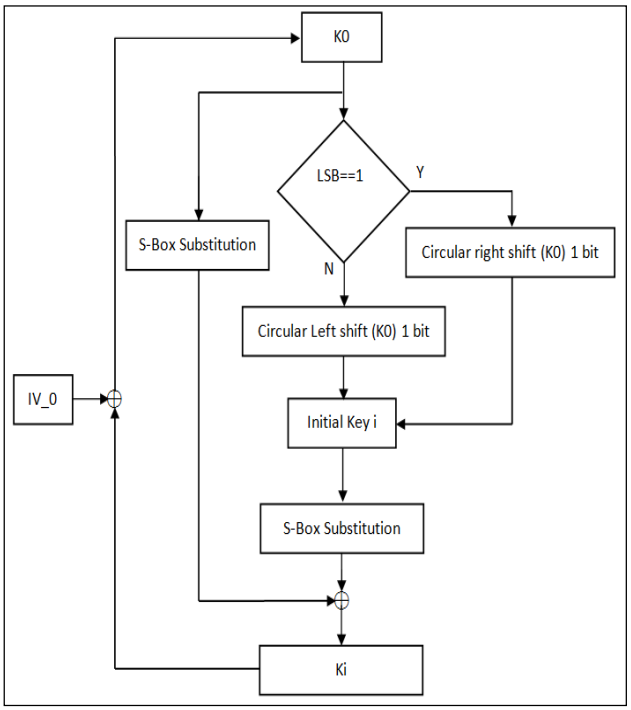

Fig. 2 SEAA Key Expansion Algorithm

The keys generated through the above algorithm are entirely different and unpredictable with the previous keys. Hence we can have unlimited unique and random keys through this algorithm. There are two possibilities for the execution of the above algorithm; firstly, it is possible that both the sender and receiver pre-generate and store a large number of keys (K1 to Kn) in a lookup table, to be used during the encryption process, and periodically update the table. This will save time during encryption. Alternately, the keys can be generated at the time of encryption, as the time required to generate one key in the above key expansion is very short. The keys generated from the above mechanism should never be used again. For example, if the number of blocks in 1st message is five, then keys K1 to K5 will be used for encryption and K6 for the tag generation. Now, for the 2nd message, K6 will become K0 for the derivation of new K1 to Kn for the second message, or if pre-calculated, K7 onward will be used for the following messages.

3.4. Modified AES

To generate Ti, we proposed using a modified AES block cipher with plaintext Pi and key K0 as input. The modified AES is the same as standard AES, with the only difference is reduced no of rounds from ten to four in the modified version, as shown in figure 3. The reason for this modification is that there are several algorithms which use four rounds of AES as building block such as stream cipher LEX [34], the Pelican message authentication code function [35], the AE algorithm POET [36], and provably secure MACs [37]. Also, this has been shown that 256 message blocks can be encrypted under the same key, given all round keys are independent [38] in 4 round AES. While our approach is to use new keys for a new message, hence the 4 round AES is sufficient to provide the desired high security against differential cryptanalysis.

Moreover, in the conventional algorithm such as AES-CTR, the input (counter) to the block cipher is changed by only 1 (increment) for different input blocks. As a result, the dependence on block cipher to produce unique Ti is enormous. Because, if Ti is not largely changed for minor input change (counter), then the same plaintext will produce similar ciphertext. However, in the proposed approach, the uniqueness of ciphertext does not only depend on Ti; instead, it also depends on the new Ki and IV_i. Hence, in the SEAA, the block cipher computations can be simplified as proposed. Also, unlike the traditional AE approach, the purpose of the block cipher in SEAA is mainly to provide strength to the authentication part, where the change in a single bit of input is desired to be reflected by a complete change in output through avalanche effect. For this, as experimental results show, four rounds are adequate to completely change the Ti by a change in a single bit in the input. Hence, reducing the complexity of AES rounds AES does not compromise security. Therefore, only 4 AES rounds are used in SEAA.

Also note that with this approach, we can use AES round keys in alternate blocks, such as, for odd number blocks, key 1 to key five can be used, and for even number message blocks, key 6 to key 11 can be used).

3.5. Tag Generation

Finally, the tag can be generated by giving the resultant XOR of all the plaintexts as input to the modified AES algorithm, and XORing the result of the block cipher with key Kn+1 as shown in fig.1. This tag of 128 bits will be attached with the ciphertext and will serve the authentication purpose similar to that in most cipher based MAC such as CMAC or PCMAC [34].

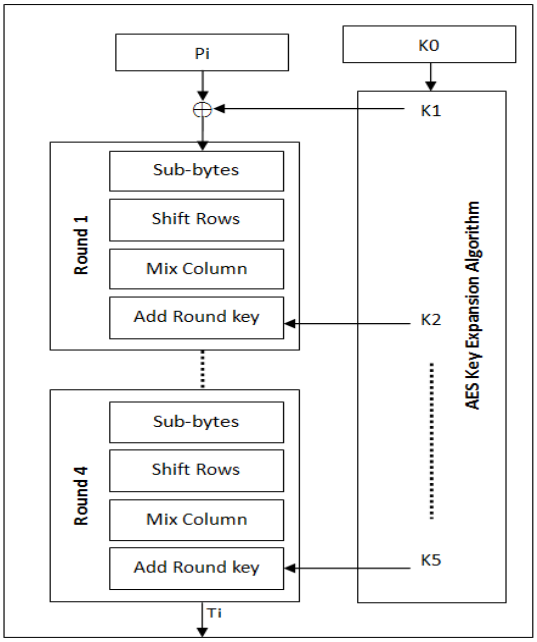

Fig. 3 Modified AES Algorithm

3.6. Decryption and Authentication Process

The decryption process of SEAA consists of XORing the ciphertext with key Ki and IV_i to get Ti, as shown below.

$$Ti = Ci \oplus IV\_i \oplus Ki \quad (3)$$

After receiving the Ti, the receiver can process Ti in the modified AES decryption process of the same four rounds to get the corresponding plaintext Pi. The order of the sub-process of the modified AES decryption process will be reversed than that in the encryption process for the recovery of plaintext.

After decryption and obtaining the plaintexts, the receiver can generate the tag by the same method through which generated by the sender, and compare the received and generated tags to ensure authentication. If the tags are the same, the receiver can be sure about the authenticity and integrity of the message. Otherwise, the receiver can discard the message by considering it forged during the path.

4. Experimental results

4.1 Assumptions and pre-computations

In the implementation stage of the algorithm, it is assumed that in SEAA, both the keys (Ki and IV_i) are pre-computed and stored (as recommended in the algorithm description section). After the sharing of secret keys K0 and IV_0, any desired number of the key can be generated through key expansion algorithm and stored in the memory, to be

utilized later in SEAA for encryption and authentication tag generation.

Table 1 shows the result of the proposed key expansion algorithm for the generation of K1 to K8 from K0. Table 1 also shows the results of the derivation of IV_1 to IV_8 from IV_0. From the generated keys (both Ki and IV_i), it can be seen clearly that these are entirely changed from previous keys and the master keys (K0 and IV_0).

Table 1: Key Expansion and IV Expansion Experimental Results

K0 = 2B7E151628AED2A6ABF7158809CF4F3C IV_0 = 09CF4F3CABF7158828AED2A62B7E1516	
K1=4043BC36E5A8B3C73940A80E7C818F57	IV_1=AF99B2887D555862BFA07CDB1C7045B7
K2=7B0839DF1C70D8FA4AE3C912200A090	IV_2=7CB2CC5216047E97F4272BE587F76DE0
K3=AEC4A51E33D4A9B3C91FAE05C8A7AA93	IV_3=9527733F23ACEDED51E4807ACA34F9CC
K4=DBCFFD4286D917D66289AAD758DABC	IV_4=E1EDAC89B978A450030555B56CD533DA
K5=655FB194ABB2A5AEEF0D772AF114518C	IV_5=DB88C352F412808EAA75BF52BFA5D6D1
K6=59DEF9100533714B5AAB5B0DA4ABCA0	IV_6=46BA74BF52A792E1FECDCA21561DA455
K7=310663AC391BFDCCDEC20E9F39E13E1E	IV_7=08232F4981FD1BD65491F098A2C19D17
K8=A492BA9D70FBEBBC8C65EA52F66940FA	IV_8=2F6F82B2E8505F4D81BFEDC458AF81B0

Furthermore, the NIST's statistical tests for randomness [39], such as Frequency test and Runs test are also performed on the generated keys to check the randomness of generated keys and the resultant p values are shown in table 2 below. The frequency test checks if the number of ones and zeros are almost equal or not in the binary key, while runs test determines whether the oscillation between different runs of ones and zeros of various lengths is too fast or too slow [39]. Since the results shows that all the generated keys pass the frequency test and runs test (p value > 0.01) hence all keys can be considered as random (however, all the applicable test in NIST statistical test suite for randomness from the total 17 test maybe performed for further endorsement).

Table 2: Experimental Results of NIST Randomness Tests on Generated Keys

Key no.	**Frequency Test P-value**		**Runs Test P-value**	
I	**Ki**	**IV_i**	**Ki**	**IV_i**
1	0.3768 (Pass)	0.5959 (Pass)	2.0000 (Pass)	0.8399 (Pass)
2	0.1573 (Pass)	0.4795 (Pass)	1.8940 (Pass)	1.6028 (Pass)
3	0.8597 (Pass)	0.4795 (Pass)	0.1109 (Pass)	0.6897 (Pass)
4	0.1573 (Pass)	0.7237 (Pass)	0.1508 (Pass)	0.0750 (Pass)
5	0.5959 (Pass)	0.7237 (Pass)	0.0199 (Pass)	0.1088 (Pass)
6	0.7237 (Pass)	0.5959 (Pass)	0.0503 (Pass)	0.0724 (Pass)
7	0.59595 (Pass)	0.2888 (Pass)	1.9773 (Pass)	1.2023 (Pass)
8	0.3768 (Pass)	0.8597 (Pass)	0.1356 (Pass)	1.6219 (Pass)

From the speed of the key expansion algorithm viewpoint, under the pre-computed assumption, the time of Key expansion algorithm can be excluded from the processing time of SEAA. However, for a reference, figure 4 shows the time required for the generation (expansion) of different no of keys from given K0 and IV_0. From figure 4, we can see that the algorithm can generate 3000 keys Ki or 2000 keys IV_i if the key expansion algorithm is run for only one second. Accordingly, any number of keys can be pre-computed in a short time duration and stored.

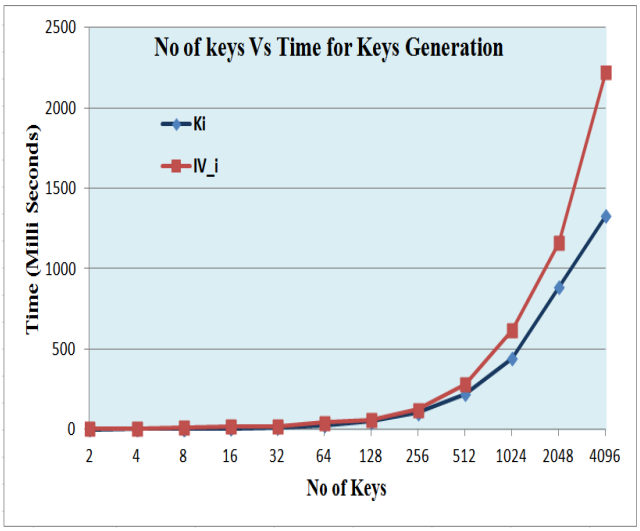

Fig. 4 Time for expansion of n no of keys through Key Expansion Algorithm

4.2 Software Implementation

The proposed algorithm has been implemented on MATLAB software on a normal Laptop with 4 GB RAM, a Core i5 processor without any code optimization for time. After having required encryption keys (IV_i and Ki) from the key expansion algorithm, the next stage of SEAA is the Ti generation using a modified AES block. For this, plaintext P1 to Pn are input to the modified AES algorithm (4 rounds only) to produce the T1 to Tn, respectively. To demonstrate the effectiveness of the proposed re-keying approach, we have used the same plaintext Pi in 8 input blocks for the same T1 to T8 generation. Then for the encryption, the generated Ti is XORed with corresponding Ki and IV_i to generate C1 to C8 as per equation 1. The results of the encryption process are shown in figure 5. From the results, it can be seen that having the same Pi and corresponding Ti as well, eight different ciphertexts (C1 to C8) are received, because of different Ki and IV_i. Hence the experimental results validate the algorithm's working with the desired output. Here IV_i and Ki are the same as shown in table 1.

K0= 2B7E151628AED2A6ABF7158809CF4F3C					
IV_0= 09CF4F3CABF7158828AED2A62B7E1516					
i	Plaintext	Ti	Ki	IV_i	Ci
1	F0F1F2F3F4F5F6F7 F8F9FAFB00000000	3E82CCABFBDFB57E6 CD694C34ABC1CAA	K1	IV_1	D158C21563225EDBEA3640162A4DD64A
2			K2	IV_2	A58083641C1CC6663C5F83B7EF4BD1DA
3			K3	IV_3	05611A8AEBA7F120F42DBABC482F4FF5
4			K4	IV_4	ACB4AFDF0021C839B9B148DCF131F5CC
5			K5	IV_5	8055BE6DA47F905E29AE5CBB040D9BF7
6			K6	IV_6	1DA55785A92B108B27B1EB52C6EB045F
7			K7	IV_7	07A7804E43395364E6856AC4D19CBFA3
8			K8	IV_8	B57FF4846374018F610C9355E47ADDE0

Fig. 5 Encryption Results of the Same Plaintext in 8 input blocks

Finally, the authentication tag is generated by the steps shown in figure 1 and explained in section 3.5. The results of the tag generation are shown in table 3 below. To verify the authentication process of our algorithm, a simple hex digit (a half byte) is changed in first byte of ciphertext block C2 from 5 to 6, as highlighted in the table below. As a result of this forgery, Ti extracted from Ci as per equation 3 is also be changed in the corresponding byte in the corresponding block. When the receiver uses this Ti to obtain the plaintext, the plaintext is completely changed, as highlighted in the table below. Finally, the resultant tag generated using recovered plaintext is completely changed from the original tag received together with the ciphertext, as shown in table 3 below. Therefore, the receiver can easily know that the integrity of the message has been compromised.

Table 3: Results of Tag Generation process of SEAA

OriginalP1= F0F1F2F3F4F5F6F7F8F9FA FB00000000	Original P2= F0F1F2F3F4F5F6F7F8F9FAF B00000000
Resultant C1= D158C21563225EDBEA364 0162A4DD64A	Resultant C2 = A58083641C1CC6663C5F83B 7EF4BD1DA
Tag (from original P1 & P2) = F20A5AE5EA35572B55C8713CB3204234	
Original C1= D158C21563225EDBEA364 0162A4DD64A	Modified C2 = A68083641C1CC6663C5F83B 7EF4BD1DA
Recovered P1= F0F1F2F3F4F5F6F7F8F9FA FB00000000	Recovered P2= 169B26FA5CFB7D115EBCE D6D6847C991
Tag (from recovered p1 and P2) = 0BC628A1D0D024D538A85C3FD6B5A1AD	

Figure 6 shows the comparison of block processing time for Ti generation in both standard AES algorithm (10 rounds) and modified AES algorithm (only four rounds). The results show that the processing time of 4 round AES is significantly less than the standard algorithm.

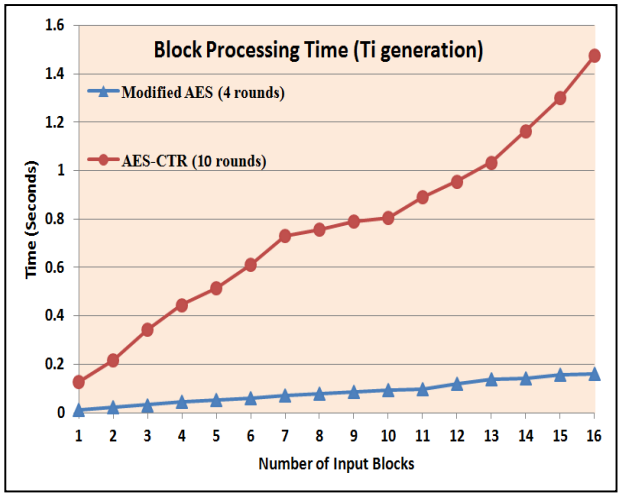

Fig. 6 Comparison of processing time of Modified and standard AES algorithm

Finally, figure 7 shows the comparison of total time for ciphertext and tag generation of the proposed algorithm with AES-GCM for up to 16 input message blocks. Here the total encryption time of the proposed algorithm includes time for keys expansion (Ki and IV both), modified AES Block processing time, encryption time (which consists of XORing the Ki, Ti, and IV_i together), and tag generation. From the results, it can be seen that the speed performance proposed SEAA algorithm is significantly better than the widely used AES-GCM on MATLAB software. Even though no code optimization has been performed and AES initialization time (such as key expansion, s-boxes generation, etc.) and tag generation times are not included in the AES-GCM results.

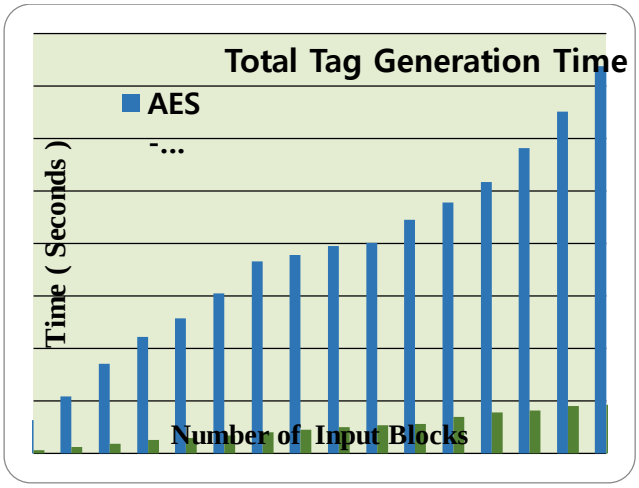

Fig. 7 Total Processing Times of the AES-GCM and SEAA

4.3 Hardware Implementation

It is important to note here that in our software implementation, the algorithm followed a serial flow. But on the hardware capable of parallelization, many of the processes can be parallelized for significant performance improvement. In particular, if the keys Ki and IV_i are pre-calculated, then the actual processing of the algorithm at runtime will only consist of modified block processing, the encryption process, and the tag generation. Here, due to the architecture of the algorithm, all the plaintext blocks can be processed in parallel to produce Ti simultaneously. Therefore, the corresponding encryption process (which consists of XORing of IV_i and Ki with Ti) can also be performed in parallel so that all the ciphertext can be generated in parallel in just one clock cycle after having Ti, Ki, and IV_i. Similarly, the tag can also be generated in parallel to ciphertext generation, so, we can see that apart from the key expansion algorithm, all the other processes can be parallelized. Therefore, the SEAA is fully parallelized (provided the keys are pre-computed and stored in memory), and hence suitable for high throughput applications.

The experiments were performed on the proposed algorithm on hardware using Xilinx Vertex-6 FPGA and Modelsim simulation software. The simulation results are shown in figure 8. The experiment for proposed encryption and authentication tag generation is performed using a single core of the proposed algorithm. The key generation takes two clock cycles for the generation of each expanded key. The modified ciphertext generation takes four clock cycles for Ti generation from Pi, and one clock cycle for XORing the Ti with Ki and IV_i. Therefore, a total of five clock cycles are required for the generation of ciphertext generation or encryption. Similarly, the tag generation used a separate algorithm core for implementation. Therefore, five clock cycles are used for encryption and authentication tag generation in parallel. The FPGA simulations results are the same as obtained through MATLAB implementation, as can be seen from fig. 8.

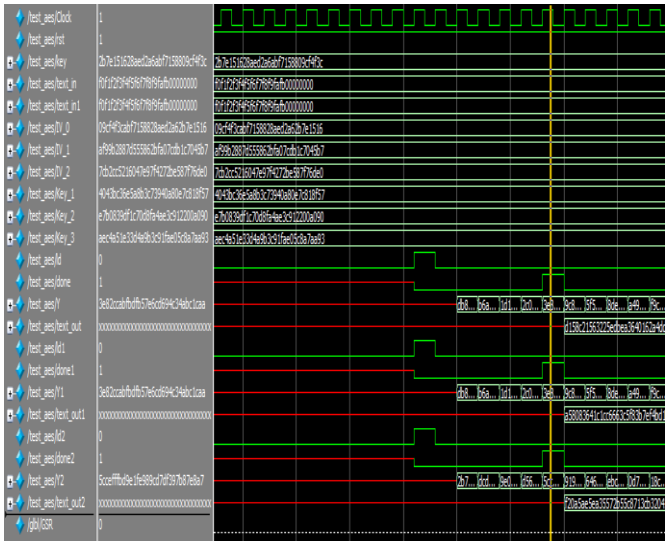

Figure 8 Simulation Results of FPGA Implementation of proposed AE algorithm

Table 4: Resource utilization of FPGA

Input blocks	*Resources*			
	CLB Slices	*Clock Freq. (MHz)*	*BRAM*	*Throughput (Gbps)*
1	956	305.64	20	7.84
2	1912	305.64	30	15.64
3	2868	305.64	40	23.47
4	3824	305.64	50	31.29
5	4780	305.64	60	39.12
6	5736	305.64	70	46.94
7	6692	305.64	80	54.77
8	7648	305.64	90	62.59
9	8604	305.64	100	70.41
10	9560	305.64	110	78.24
11	10516	305.64	120	86.06
12	11472	305.64	130	93.89

Table 4 shows the resource utilization of the proposed algorithm on FPGA. The throughput has been calculated by the following formula.

Throughput = number of bits *freq / clock cycles

There have been different implementations of AES-GCM on FPGA. Table 5 shows the results of the throughput comparison results of AES-GCM implementation on the FPGA and proposed algorithm. The proposed implementation results show high throughput implementation with the recent optimization results of the AES-GCM algorithm.

Table 5: Comparison of proposed and AES-GCM algorithms on FPGA hardware

S. No	*Implementation*	*Resources*		
		CLB Slices	*Clock Freq. (MHz)*	*Throughput (Gbps)*
1	Yang et al. [19]	463328	271.00	34.69
2	Lemsitzer et al. [20]	13200	110.00	14.1
3	Y. Zhang et al. [21]	6482	381.60	48.8
4	Proposed SEAA	11472	305.64	93.89

5. Discussion on Security Algorithms

5.1. Mathematical Analysis

As explained in section 2, the encryption process of our algorithm consists of XORing the Ti with two unique keys Ki and IV_i. Therefore, for the attacker, it is impossible to get to correct Ti without knowing the correct values of Ki, IV_i. This is because there exist 2256 possible combinations of Ki, IV_i, and Ti that can produce the same Ci. As, both Ki and IV_i will not be used again, so the attacker has no further information to guess the correct combination of three numbers among 2256 values.

The key expansion algorithm is designed to produce a new key every time, so assuming the Ki, IV_i will never be used again, the algorithm is as secure as OTP. Even if we consider the key expansion algorithm generates a key Ki that has already been used to encrypt any plaintext block in the past together with IV_i, even then it is unlikely that at the same instant the corresponding key IV_i will also be generated same as used earlier with that Ki. This is because both the keys Ki and IV_i are expanded through a different mechanism. So, we can see that even if a key Ki is repeated, the chances of the same corresponding IV_i is negligible. Hence, the repetition of the key will not make any compromise in security because of freshness in the corresponding key.

Similarly, knowing Ti somehow may also not help attacker because K0, which is used in the modified AES block, is not known to the attacker so he cannot reach to the plaintext easily.

5.2. Protection Against Compromised Key

Suppose the attacker can get the correct value of any one of the two secrets keys, K0 or IV. By knowing the key expansion algorithm, he can get all the corresponding Ki or IV_i as the sender and receiver know. Now if the attacker tries to decrypt the ciphertext, he will face two challenges; Firstly, which values of Ki or IV_i are to be used is a challenge for the attacker, especially if he does not have information about the block lengths of all the messages from the sender to the attacker till this message. If the attacker missed any single message and did not know the block length of that message, he cannot be sure about the correct value of Ki to be used for decryption.

Secondly, we know that even if the attacker can identify the correct value of Ki, he still cannot get the correct plaintext because he does not know the other two variables, Ti or IV_i. For an attacker, there still exist 2128 possible combinations of Ti and IV_i, which can produce the same Ci, despite knowing Ki. So, we can see. As explained in section 2, the advantage of our approach of using three variables (including two unique keys) is that knowing anyone key will not help the attacker, and the data protected with the proposed algorithm will remain secure.

Furthermore, due to proposed key expansion design, where K0 is used in IV_i expansion and IV_0 is used in Ki expansion, knowing any Ki will not help the attacker to reach any other Ki or K0. Hence, unlike standard AES key expansion in which knowing one round key could lead to even disclosure of master key [36], the master key in our algorithm will remain secure even after knowing the correct Ki.

We can see that this is a unique kind of protection, which our algorithm provides where the compromise of key does not compromise security. This does not exist in any other existing or proposed algorithm to the best of author's knowledge, because all of them uses one key and not designed to be used with forward secrecy or re-keying. So, if the attacker can get the secret key somehow, he can easily decrypt all future ciphertext produced by the help of that secret key.

5.3. Chosen Plaintext Attack

Generally, by the chosen-plaintext attack, an attacker tries to get the secret key used for encryption, so that later he can use the same key to decrypt ciphertext correctly. Suppose an attacker can get temporary access to the system/device using our encryption algorithm. The attacker will try to get some useful information such that when he does not have access to the system, he could get decrypt the ciphertext correctly. Now by having access to such a system, the attacker can get the ciphertext against his plaintext of choice. Even this cannot help the attacker because the ciphertext is generated using Ki and IV_i, which the attacker does not know either. It is possible that by applying some cryptanalysis technique on the modified AES block, the attacker gets success in obtaining the correct K0. But still, he does not know which value of Ki is to be used when, and also, he does not know the IV_i as discussed above. Hence the proposed algorithm is secure against the Chosen plaintext attack, which means it is also secure against known-plaintext attack, and ciphertext only attacks as well. Furthermore, even in a worst-case scenario, where the attacker can fetch IV_i and Ki also, even then knowing Ki and IV_i for other blocks will not be easy for attacker as

well knowing the correct combination of Ki and IV_i are two difficult tasks remains for the attacker.

5.4. Side-Channel Attack Protection

AS the design goal of the algorithm is to ensure protection against side-channel attacks, therefore, because of using new Ki and IV_i for every new message block protects from providing any useful information to the attacker through passive side-channel attacks such as DPA. As re-keying is incorporated in the proposed algorithm, hence it inherently protects the algorithm from a Side-channel attack, and no separate re-keying or fresh keying mechanism or exchange is required.

Also, many of the existing block ciphers use one key for a longer duration, which makes the key a target of an attacker, and they use different methods such as brute force, etc. But our algorithm uses new keys every time, which makes brute force type attack naturally not applicable because spending a significant amount of time and resources in exhaustive key search (brute force) makes sense in traditional block ciphers. Still, it is of no use in our use case. Because knowing anyone of Ki or IV_i will not help the attacker.

5.5. Nonce Misuse Protection

Again as the desired goal of the proposed algorithm is to also ensure its protection against nonce misuse attacks such as on AES-GCM. So, we can see that the proposed algorithm does not use any nonce. Hence the nonce misuse attack is not applicable in our algorithm in that simple context. Also, this reduces the nonce overhead and processing costs. Secondly, whenever the new key is generated (Ki or IV_i), it becomes the basis for the generation of new keys (Ki and IV_i). So, the flaw which exists in the counter-based nonce (whereby injecting faults the counter may be overflowed to repeat the nonce) is also not applicable to our proposed algorithm.

Finally, the use of two new keys every time ensures that even if the attacker can make repetition of any one of the keys (Ki or IV_i) somehow, then again, it is extremely unlikely that the same pair of IV_i and Ki is used again. This ensures that the confidentiality will remain protected even if one or both keys start repeating because the probability of repetition of the same used pair is still very low in that case.

5.6. Protection against Active Attacks

As explained in the Experimental results section, if an attacker can modify the ciphertext, without any change to the tag. Then the receiver can easily be intimated about this modification because the generated and received tags will not match. This same is true if the attacker can change the tag without any change to the ciphertext.

Also because for tag generation both K0 (in block process) and Ki (in last XOR for tag) are used, which are not known to the attacker, so for him, it is extremely difficult to forge both, the ciphertext and corresponding tag such that the receiver can be fooled about forgery because the attacker does not have any useful information to do this.

Lastly, if an attacker tries to use his key (IV_i and/or Ki) to make the receiver fool, the receiver cannot be a fool, because the correct plaintext will be recovered only using the correct keys of the intended sender. So, the proposed algorithm will ensure authenticity, integrity, and confidentiality of data against all known famous active/passive attacks.

6. Conclusion

To address the need for nonce misuse attack and side-channel attack resistant AE algorithm, which is suitable for the satellite application environment, this paper presents a new AE algorithm that is designed based on the re-keying or forward secrecy principle. The concept of re-keying is incorporated in the proposed algorithm such that it provides the inherent resistance against both, the side-channel attacks such as DPA (which is applicable to the most block cipher) as well as the nonce misuse attack (which applies to AES-GCM). Furthermore, by incorporating the re-keying technique in the proposed algorithm, the computational complexity of the block cipher has been reduced than the famous AES-GCM algorithm. Therefore, the experimental results show a significant increase in the computational speed &

Acknowledgments

This work was supported by the 2018 Industrial Internet Innovation and Development Project of the Ministry of Industry and Information Technology (no. IIIDP-9.1-2018 of MIIT).

References

[1] CCSDS, "The Application Of Security To CCSDS Protocols," Informational Rep. (Green Book), vol. CCSDS 350., no. March 2019.

[2] L. Jianwei, L. Weiran, W. Qianhong, L. Dawei, and C. Shigang, “Survey on key Security technologies for space information networks,” J. Comm. and Info. Netw., vol. 1, no. 1. pp. 72–85, 2016.

[3] National Institute of Standards and Technology (NIST) and Special publication 800-38D, "Recommendation for Block Cipher Modes of OperationS: Galois/Counter Mode (GCM) and GMAC,” 2007.

[4] ISO/IEC19772:2009, “Information technology-Security techniques-Authenticated Encryption,” 2009.

[5] CCSDS Recommended Standards, "CCSDS Cryptographic Algorithms," CCSDS 352.0-B-1 (Blue Book), no. November 2012.

[6] V. Arun, K. Vanisree, D. L. Reddy, “Implementation of AES-GCM encryption algorithm for high performance and low power architecture using FPGA,” Int. J. Res. Appl., vol. 1, no. 3, pp. 120–131.
[7] J. Su, N. Gu, Q. Bai, and C. Lin, “Parallel Implementation of AES-GCM with High Throughput and Energy Efficiency,” 2018 Int. Conf. Netw. Netw. Appl., pp. 251–256, 2018.
[8] N. Ahmed, L. M. Wei, and M. H. Jabbar, “Advanced Encryption Standard with Galois Counter Mode using Field Programmable Gate Array,” J. Phys. Conf. Ser., vol. 1019, no. 012008, pp. 1–7, 2018.
[9] H. Böck, A. Zauner, S. Devlin, J. Somorovsky, and P. Jovanovic, “Nonce-Disrespecting Adversaries : Practical Forgery Attacks on GCM in TLS,” in USENIX WOOT, 2016, pp. 1–11.
[10] P. Kocher, J. Jaffe, B. Jun, and P. Rohatgi, “Introduction to differential power analysis,” J. Cryptogr. Eng., vol. 1, no. 1, pp. 5–27, 2011.
[11] E. Brier, C. Clavier, and F. Olivier, “Correlation power analysis with a leakage model,” LNCS, vol. 3156, pp. 16–29, 2004.
[12] E. Ronen, C. O. Flynn, A. Shamir, and A. Weingarten, “IoT Goes Nuclear : Creating a ZigBee Chain Reaction,” Cryptol. ePrint Arch., vol. Report 201, p. http://eprint.iacr.org/2016/1047.
[13] A. Moradi and T. Schneider, "Improved Side-Channel Analysis Attacks on Xilinx Bitstream Encryption of 5, 6, and 7 Series," LNCS, vol. 9689, pp. 71–87, 2016.
[14] S. Koteshwara and A. Das, “Comparative Study of Authenticated Encryption Targeting Lightweight IoT Applications,” IEEE Des. Test, vol. 34, no. 4, pp. 26–33, 2017.
[15] W. Diehl, A. Abdulgadir, F. Farahmand, et al. “Comparison of Cost of Protection against Differential Power Analysis of Selected Authenticated Ciphers,” Cryptography, vol. 2, no. 26, pp. 1–32, 2018.
[16] “PRIMATEs v1.02. (Sep. 2014). CAESAR submission.,” 2014.
[17] S. Koteshwara, A. Das, and K. K. Parhi, “Architecture Optimization and Performance Comparison of Nonce-Misuse-Resistant Authenticated Encryption Algorithms,” IEEE Trans. Very Large Scale Integr. Syst., vol. 27, no. 5, pp. 1053–1066, 2019.
[18] H. Weiss, “CCSDS Standardization of Security Algorithms for Civil Space Missions,” pp. 1–11, 2012.
[19] B. Yang, S. Mishra, and R. Karri, “High Speed Architecture for Galois / Counter Mode of Operation (GCM),” in IACR Cryptology EPrint, 2005, p. 146.
[20] M. B. S. Lemsitzer, J.Wolkerstorfer, N. Felber, “Multi-gigabit GCM-AES architecture optimized for FPGASs,” in International workshop on cryptographic hardware and embedded systems CHES, 2007, pp. 227–238.
[21] Y. Zhang, N. Wu, F. Zhou, X. Zhang, and Z. Jinbao, “High performance AES-GCM implementation based on efficient AES and FR-KOA multiplier,” IEICE Electron. Express, vol. 15, no. 14, pp. 1–9, 2018.
[22] S. Gueron, A. Langley, and Y. Lindell, “AES-GCM-SIV: Specification and Analysis.,” IACR Cryptol. ePrint Arch., vol. 2017, no. July 2017, p. 168, 2017.
[23] S. Koteshwara, A. Das, and K. K. Parhi, “Performance comparison of AES-GCM-SIV and AES-GCM algorithms for authenticated encryption on FPGA platforms,” Conf. Rec. 51st Asilomar Conf. Signals, Syst. Comput. ACSSC 2017, vol. 2017-October, pp. 1331–1336, 2018.
[24] C. Dobraunig, M. Eichlseder, A. F. Mendel, and M. Schlaffer., “Ascon v1.2. Submission to the CAESAR Competition,” 2016.
[25] H. Wu, “ACORN: A Lightweight Authenticated Cipher (v3),” 2016.
[26] E. B. Kavun, H. Mihajloska, and T. Yalcin, “A Survey on Authenticated Encryption — ASIC Designer ’ s Perspective,” ACM Comput. Surv., vol. 50, no. 6, p. Article 88 (21 pages), 2017.
[27] W. Diehl, F. Farahmand, A. Abdulgadir, J.-P. and Kaps, and K. Gaj, “Face-off between the CAESAR Lightweight Finalists : ACORN vs. Ascon," in 2018 International Conference on Field-Programmable Technology (FPT), 2018, pp. 333–336.
[28] E. Prouff and M. Rivain, “Masking against Side-Channel Attacks : a Formal Security Proof,” LNCS, vol. 7881, pp. 142–159, 2013.
[29] C. Dobraunig, M. Eichlseder, S. Mangard, et al.," ISAP – Towards Side-Channel Secure Authenticated Encryption," IACR Trans. Symmetric Cryptol., vol. 2017, no 1, no. 80–105, 2017.
[30] A. Chakraborti, T. Iwata, K. Minematsu, and M. Nandi, “Blockcipher-Based Authenticated Encryption: How Small Can We Go?,” J. Cryptol., vol. https://do, 2019.
[31] A. Christoph Dobraunig, F. Koeune, S. Mangard, et al., “Towards fresh and hybrid re-keying schemes with beyond birthday security,” LNCS, vol. 9514, pp. 225–241, 2015.
[32] A. Murtaza, S. Jahanzeb, H. Pirzada, and M. N. Hasan, “An Efficient Encryption Algorithm for Perfect Forward Secrecy in Satellite Communication,” Commun. Comput. Inf. Sci., vol. in press, 2019.
[33] A. Murtaza, T. Xu, S. Jahanzeb, H. Pirzada, and L. Jianwei, “A Lightweight Authentication and Key Sharing Protocol for Satellite Communication,” Int. J. Comput. Commun. Engin, vol. 9, issue 1, pp 47-53, 2020.
[34] A. Biryukov, “The Design of a Stream cipher LEX,” LNCS, vol. 4356, pp. 67–75, 2007.
[35] J. Daemen and V. Rijmen, “The Pelican MAC Function.,” IACR Cryptol. ePrint Arch., vol. 088, 2005.
[36] F. Abed et al., “Pipelineable On-Line Encryption,” LNCS, vol. 8540, pp. 1–20, 2014.
[37] K. Minematsu and Y. Tsunoo, “Provably secure MACs from differentially-uniform permutations and AES-based implementations,” LNCS, vol. 4047, pp. 226–241, 2006.
[38] J. Daemen, M. Lamberger, N. Pramstaller, et al., “Computational aspects of the expected differential probability of 4-round AES and AES-like ciphers,” Computing, vol. 85, no. 1–2, pp. 85–104, 2009.
[39] A. Rukhin, J. Soto, and J. Nechvatal, "A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications," NIST Spec. Publ. 800-22, no. April 2010.

Abid Murtaza was born at Karachi, Pakistan. In 2010, he has received the M.Sc. Electronics degree from the University of Karachi, Karachi, Pakistan. He has been working in Satellite Ground Control Station at Space and Upper Atmospheric research Commission (SUPARCO), the National Space Agency of Paksitan, since 2010. Currently, he is pursuing his Ph.D. degree in Information security in Space Technology at Beihang University (Beijing, China). So far in his Ph.D. he has published more than 20 research papers at international conferences and journals. His research interests include Satellite Communication, Space Information networks, cryptography, Information Security for Satellite communication, etc.

Syed Jahanzeb Hussain Pirzada was born at Attock, Pakistan. In 2007, he received his BE degree in electronics engineering from NED University of engineering and technology, Karachi, Pakistan. In 2012, he received MS degree in electrical, electronics, control and instrumentation engineering from Hanyang University, Seoul, South Korea. He has more then 10 year's experience in working for the National Space Agency of Pakistan (SUPARCO). Since 2018 he is enrolled for PhD degree in Space Technology Applications at Beihang University, Beijing, China. His research interest is information security and cryptography for satellite applications.

Tongge Xu graduated from Beijing University of Aeronautics and Astronautics in 1993 with a master's degree in engineering. He is now Associate Professor of school of cyber science and technology at Beihang university, Beijing, China. His research areas are network management and flow / protocol analysis technology, UNIX / Linux system development, large information system design and development technology, public opinion big data analysis and mining

Liu Jianwei was born at Shandong, China. He received BS and MS degrees in electronics and information engineering from Shandong University, Shandong, China in 1985 and 1988 respectively. He received his Ph.D. degree in electronics and communication systems from Xidian University, Shaanxi, China in 1998. He is now professor and dean of school of cyber science and technology at Beihang University, Beijing, China. His current research interests include wireless communication networks, cryptography, and information and network security.