Preserving Data Security in Distributed Fog Computing

Hassan Noura[1], Ola Salman[1], Ali Chehab[1], and Raphaël Couturier[2]

[1]Electrical and Computer Engineering, American University of Beirut (AUB), Beirut, Lebanon
[2]Univ. Bourgogne Franche-Comté (UBFC), FEMTO-ST Institute, CNRS, Belfort, France

***Abstract*—In this paper, a novel cryptographic solution is proposed to secure data in fog computing. The solution combines the AES-GMAC operation mode with information dispersal over GF(2^w) to provide data confidentiality, integrity, and availability along with source authentication. The value of w is flexible (8, 16, 32 or 64) and it could be configured according to the fog device features. Moreover, the proposed cryptographic solution is based on the dynamic key-dependent approach, which allows for a good compromise between the security level and computational complexity. In the proposed solution, the collected data at one fog node is encrypted, authenticated and dispersed in a pseudo-random manner to its n neighbor fog nodes. For data recovery, any k of the n fragments along with the corresponding dynamic key are required to retrieve the original data. This complicates the attacker's task who needs to compromise at least k fog nodes to disclose the encrypted data. Additionally, attackers should seek the dynamic key, which is different for each input data. On the other hand, redundant fragments protect the stored data against up to $(n-k)$ fog nodes' failure or unavailability. The security and performance analysis tests show that the proposed security scheme exhibits a high level of efficiency and robustness.**

I. Introduction

With the emergence of the Internet of Things (IoT), billions of devices will be connected to the Internet, ranging from low power devices (e.g. sensors) to more capable ones (e.g. cars), and the heterogeneous set of connected devices exhibits different capabilities in terms of power, processing, storage, and utilizes different communication protocols. Additionally, different applications will run on top of these devices consuming and generating various kinds of data with different sizes, semantic, frequency, and privacy levels. According to the International Data Corporation (IDC), 40 trillion GB of data will be generated by 2020. In this era of big data, novel IT and data technologies have emerged to manage this enormous amount of data. Pushing all the data to the cloud presents limitations in terms of the required bandwidth and latency. Fog computing has been introduced to overcome these limitations, bringing the data to the network edge. It consists of deploying data processing, management, and storage "close" to the users' devices to enable time-critical applications. Note that fog computing is not an alternative to cloud computing, but rather a complementary one. Thus, the connection between fog nodes and the cloud is mandatory. The collected data is analyzed and pushed to the cloud for backup and the short-term data is temporarily stored at fog nodes. This sort-term data can be of critical type such as user credentials. The fog node can be a network device (e.g. gateway, router, etc.), a user device (e.g. laptop, mobile phone, etc.), or a small data center. Thus, given the constrained capabilities of fog nodes in terms of power, storage, and processing, data security faces a major challenge. Since traditional cryptographic operations are energy or memory consuming, these security solutions are not appropriate for fog computing, and on the other hand, a poorly designed lightweight solution would lead to compromising the constrained fog nodes. Accordingly, it is essential to have an efficient scheme for secure data storage to ensure data confidentiality, availability, and integrity with source authentication, while having low latency, energy and/or memory consumption to meet the fog node constraints.

A. Contributions

This paper introduces a novel scheme for data protection within fog computing. It combines secret sharing over GF(2^w) with the use of a dynamic key-dependent approach to achieve data protection and availability, where w is flexible and it can be set to 8, 16, 32 or 64. The proposed method consists of dispersing the collected data into n encrypted fragments using a dynamic key that changes at a periodic interval. For data recovery, any k of the n fragments, along with the corresponding dynamic key are required. The encrypted fragments are then distributed over the fog node's neighboring nodes. As such, an attacker has to compromise at least k fog nodes to obtain the useful analyzed information. On the other hand, redundant fragments protect the stored data against the failure or unavailability up to $(n-k)$ fog nodes.

B. Organization

This paper is organized as follows. Section II describes briefly relevant related works. In Section II, a review of the different secret sharing schemes is included. Section IV presents the network and threat models, summarizes the notations used in this paper, and defines the design goals and the evaluation metrics. In Section V, the proposed key derivation function and cipher primitives construction methods are presented. Then, Section VI describes in details the proposed cryptographic solution. Section VIII presents the experimental security analysis. Moreover, a cryptanalysis discussion is introduced in Section VIII. Section IX evaluates the performance of the scheme. Finally, an insight into possible future work concludes the paper.

II. Related Work

Securing IoT networks is key to enable their wide adoption. While previous works consider the protection of IoT devices [1], the focus of our proposed solution is about data protection. Data security is one of the key challenges in the big data era [2]. In this context, securing the data in cloud computing invoked the efforts of crypt-analysts, network security experts, software security engineers, and many others, and data breaches are still occurring within cloud computing [3]. In fact, the data security issue is aggravated in the case of fog computing [4]. Delivering Security as-a-Service (SECaaS) was proposed to ensure end-to-end system security including fog nodes, network, and data security [5]. In this paper, we focus on cryptographic solutions for data security. Such solutions have been widely used in cloud computing to ensure data confidentiality, privacy, and integrity [6], [7]. However, the application of these solutions in the fog domain is not a straightforward task. To preserve the confidentiality of data, lightweight symmetric and asymmetric encryption algorithms were proposed in the literature to cope with energy-constrained devices [8], [9]. Within this context, we aim at guaranteeing data integrity, availability, source authentication in addition to data confidentiality of fog devices by proposing a symmetric cryptographic solution that requires fewer computations and resources when compared to the asymmetric solutions [8]. The proposed solution targets fog devices that are computationally constrained and thus, not capable of preforming intense computations; they are capable of performing very basic operations and lightweight encryption. In this respect, few recent works proposed data security solutions for the fog computing domain, including the data privacy issue. In [10]–[12], the vehicular network case was considered, where one of the most private information, the users' location, is exposed to malicious threats. In [10], the authors present a solution based on Ciphertext-Policy Attribute Encryption (CP-ABE) to preserve data privacy and confidentiality. The solution minimizes the access latency by predicting user's mobility and pushing useful data to the nearest fog node. In [11], a reliable and privacy-preserving task re-composition (REPTAR) for vehicular crowd-sensing data is presented. The solution is based on a modified homomorphic Paillier encryption and super-increasing sequence to preserve crowd-sensed data privacy, confidentiality, and integrity. In [12], the authors proposed a solution based on certificate-less aggregate signcryption (CLASC) for securing road surface condition crowd-sensing. The solution preserves data confidentiality, integrity, mutual authentication, privacy, and anonymity. Differential privacy preserving query model is proposed in [13], and Laplacian noise is added to ensure the robustness of the proposed solution. In [14], a secure fog orchestrator is presented based on secure-by-design protocols. The solution is meant to deliver IoT services while preserving privacy in a compromised network. In [15], data de-duplication with dynamic ownership management is proposed for privacy preserving of multi-party owned data. A proof of ownership process with an asymmetrical key system is employed to manage the data access control. Securing aggregated data at the fog level is considered in [16], [17]. In [16], homomorphic Paillier encryption, Chinese Remainder Theorem, and one-way hash chain techniques are combined to ensure the privacy of the aggregated data communicated from the fog nodes to the cloud. In [17], Domingo-Ferrer additive privacy scheme is applied to preserve privacy of the aggregated data at the fog level while minimizing storage and communication overhead. Being a distributed security solution, the blockchain technology has enabled peer-to-peer authentication without the need of a centralized authority. Thus, applying the blockchain concept in the fog computing domain has been proposed to provide data integrity and user authentication [18]. However, this is associated with a high cost in terms of power consumption and storage resources, which requires powerful fog nodes. The main limitation of the reviewed cryptographic solutions is that they try to enhance the performance by reducing the computational overhead, but this comes at the expense of a reduced security level, which is not to be compromised. For example, the majority of recent solutions uses systematic IDA, which includes an identity matrix that reduces the mixing level.

In order to address the limitations of the previous works, some modifications are to be introduced to maintain the highest possible security level. For example, in this paper, we modified IDA by applying it at the sub-matrix level to enhance performance by enabling parallel computation. Moreover, look up tables are used instead of performing multiplication operations. On the other hand, to maintain the highest possible level of security, we use dynamic IDA matrices for each input data.

Moreover, as indicated in Table 1, existing solutions use a keyed hash function such as HMAC to ensure data integrity and source authentication. To reduce the message authentication overhead, we proposed to use AES for message authentication (GMAC) in addition to message confidentiality. The justification of this choice is that AES is well optimized for performance when compared to HMAC. We have selected GMAC for message authentication and integrity since it requires fewer computations compared to the CMAC operation mode (Galois multiplication instead of AES). Our proposed solution can be considered as a GCM operation mode, but it is adapted with the IDA concept (message authentication should be realized after the IDA step and for each fragment independently). We compared AES-128 to HMAC with SHA-512, and we found out that AES-128 is faster 7 times at least compared to HMAC with SHA-512. This allows us to reduce the message authentication overhead in the proposed solution when compared to the previous one. This solution is proposed towards preserving fog data with at a high level of security and the minimum possible overhead. Thus, to the best of our knowledge, there is no existing work that employs the dispersal information concept to ensure data confidentiality, integrity, availability, with source authentication in the fog computing domain.

III. Background

Secret Sharing is a distributed secret protection scheme. It consists of distributing a secret over several entities to overcome the centralized scheme disadvantages such as single point of failure, secret key disclosure, etc.

Secret sharing consists of fragmenting the secret into several fragments called **shares**. The secret can be recovered by the inverse process applied to a set of the distributed shares. As such, the secret can be recovered even if some of the shares are lost. However, a minimum number of shares needs to be collected in order to recover the secret, which is referred to as **the threshold**. In the literature, two well-known schemes are proposed in the secret sharing domain, the Shamir Secret Sharing Scheme (SSSS), and the Rabin Information Dispersal Algorithm (IDA).

A. Shamir Secret Sharing Scheme, SSSS

In 1979, Shamir proposed a polynomial-based secret sharing scheme [19]. The proposed solution consists of hiding a secret inside a polynomial whereby, given partial information of the polynomial, one can recover the secret.

SSSS stems from the fact that a polynomial of a fixed degree can be determined if we know the values that this polynomial takes at (n+1) points, where n is equal to the polynomial degree. The process of calculating a polynomial from its (n+1) points is known as the Lagrange interpolation method.

The fragments in SSSS have the same size as the secret, which makes it ideal from an information security perspective. However, from a memory space consumption perspective, SSSS is not that efficient.

The cost for space consumption is the factor n for a (m, n) scheme, where n stands for the number of fragments and m stands for the number of fragments needed for recovering the secret. As such, for a secret of length $|S|$, the total space needed after fragmentation is $n \times |S|$.

B. Rabin Information Dispersal Algorithm, IDA [20]

The objective of the Rabin Information Dispersal Algorithm is to overcome the SSSS storage overhead. IDA presents an efficient memory consumption scheme where a file F of size L is divided into n pieces, each with a size of $\frac{L}{m}$, where m is the number of shares needed to recover the original file F. IDA is space efficient because $\frac{n}{m}$ can be chosen to be close to 1. In fact, IDA is a diffusion transformation scheme that uses an invertible matrix for encoding/decoding data. Practically, linear algebra $modulus\ p$ is employed, where p is a prime. This is due to the convenience of dealing with data encoded in the finite field Z_p in the computer security domain.

The dispersal algorithm uses an n by m matrix, A, where n and m are two integers. An important condition is that any m rows of the matrix are linearly independent. For an input denoted by in, the output, denoted by out, is a sequence of integers of size n. In fact, F is stored as a sequence of f bytes, then each byte can be represented by an integer between 0 and 255. Thus, the input of size f is fragmented into a sequence of row vectors: $in = (c_0, c_1, c_2, ..., c_{\frac{f}{m}-1})$, where each c_i has a length m. In case f is not a multiple of m, F is padded with $(f \mod m)$ zeros. As a result, each c_i is considered as a sequence of m integers, such that $c_i = (c_{i,0}, c_{i,1}, ..., c_{i,m-1})$. Having $\frac{f}{m}c_i$, with c_i a vector of length m, these vectors result in a matrix C of $m \times \frac{f}{m}$. Finally, the IDA fragments are obtained by applying the following equation: $F = A \times C$; where F is the fragments matrix and $f_i = (f_{i,0}, f_{i,1}, ..., f_{i,\frac{f}{m}-1})$.

From a mathematical perspective, based on the information ratio, a perfect secrecy is achieved when the size of the shares is equal to the size of the secret at least [21]. Thus, Shamir's scheme presents better security level than the IDA one. In fact, even though the IDA's scheme shares the disclosure risk over n fragments, but it does not restrict unauthorized access to any of them. Consequently, if one fragment of information was leaked, the opponent can guess the number of participants, the threshold, the length of the data, and many other patterns. This makes IDA a non-perfect scheme from a security perspective. In this context, we return to the fact that there is always a compromise between performance and security level. In the following, we present the proposed IDA variants to enhance security level of the IDA scheme.

C. IDA Variants

1) Krawczyk (1998): Krawczyk's scheme [22] is a combination of Shamir SSSS, IDA, and symmetric encryption. First, a random key is chosen and used to encrypt the secret. Then, the encrypted secret is partitioned by applying IDA. Additionally, the key is partitioned with SSSS and sent along with the secret partitions.

2) AONT-IDA (2008) [23]: AONT-IDA consists of applying first AONT (All or Nothing Transform) [24], followed by IDA. In AONT-IDA, the message sequences are encrypted first and then, the pseudo sequences are fed into the IDA diffusion matrix.

AONT, proposed by Rivest [24], is a block cipher scheme that serves initially as a solution for small key spaces. All the blocks need to be decrypted first to recover the original message, which explains the designation of "ALL or NOTHING".

As per the simulations performed in [23], the AONT-IDA, IDA, and Krawczyk schemes have their execution time increasing linearly with the secret size. IDA is the fastest, Krawczyk is the second fastest, and AONT-IDA is the slowest. From a theoretical perspective, AONT-IDA adds the AONT execution time compared to the IDA time, which explains the simulations results.

3) AONT-RS 2011 *[25]:* AONT-RS adds integrity check to the encoded data. Additionally, it employs a systematic IDA scheme such as Reed Salomon

(RS). AONT-RS differs from AONT-IDA by adding the hash of the encoded sequences to the fragments distributed over the n participants. Thus, IDA can be seen as a special case of the Reed-Solomon error correcting codes when using the Galois Field arithmetic [26], [27].

4) CAONT-RS 2014 *[28]:* CAONT-RS modifies the existing AONT-RS scheme by replacing the random information employed in AONT with cryptographic hashes from the secret. Thus, data encryption is performed by using a cryptographic hash key derived from the data itself. Consequently, the dispersed information preserves the content similarity by including a data-dependent random sequence.

5) Salted IDA (2011) [29]: The Salted IDA scheme consists of two stages: salting the data followed by IDA. Salting data consists of adding pseudo randomness using a secret seed and a deterministic function fs to produce a random sequence. Then, the data is encrypted and dispersed. The deterministic function can be a hash function applied to the secret seed.

6) IDA-Over Encryption 2016 *[30]:* This variant uses multi-layer encryption with IDA. The scheme aims at reducing the re-encryption overhead when authority revocations occur such as the case with key leakage. Thus, in case of authority revocation, $\frac{1}{4}$ of the data needs to be re-encrypted, which reduces the computation overhead.

7) IDA-XOR 2016 *[31]:* This variant is based on "error code correction". It is a lightweight IDA, without using general RS codes, combining the generation of random bits with the systematic IDA. It presents better performance than SSSS and an ideal information ratio, but it is not space efficient.

To compare the different schemes, several metrics have to be considered, according to [32], and these include the storage factor, the key management scheme, the data de-duplication option, the revoking authority, the provided security options, the computational overhead, and performance optimization. In table I, we present the comparison of the different secret sharing schemes. The storage factor represents the number that multiplies the secret size K. It is expressed in terms of the number of shares n, the threshold m, the secret size K, the size of data cipher hash $H(C)$, and ϵ. We can see that IDA and IDA with encryption exhibit the lowest storage overhead.

IV. Problem formulation

This section presents assumptions about the considered IoT network and the anticipated adversaries, as well as a definition of the problem scope. The notations used are listed in Table II.

A. Network Model

Our system model consists of a network of IoT networks. Each IoT network consists of M devices, where each device is denoted by Dev_i, such that $i \in [1, M]$. These M devices are connected to a fog node. Therefore, each fog node, denoted as Fo_l such as $l \in [1, m]$, is responsible for gathering data from a set of devices $\mathcal{S}_j$, where $j \in [1, m]$. Generally, for each application session s, an IoT device Dev_i generates data denoted by d_i^s. Apart from collecting and processing data, a fog node is able to communicate with at least $(m - 1)$ other fog nodes that are in its neighborhood. Each fog node possesses a secret key SK and obtains a new nonce after each new session, s. Several fog key management schemes were presented recently in the literature such as the trusted certification solution or the symmetric Key Distribution Center (KDC). For our proposed solution, KDC is more suitable since it involves low overhead in terms of communication and computations. Moreover, readers can refer to [33] for more details about the possible key management solutions within fog-IoT systems. Even though a blockchain-based key management scheme could be applied to a fog-IoT system [34]; However, the scalability of such a scheme would be limited to the storage capacity of the fog nodes. After each s, a fog node produces a new dynamic key DK_i from its static key SK and the obtained nonce (i.e. Nonce is generated based on a Secure Pseudo Random Number Generation function (SPRNG)). Each fog node has the ability to perform a stream cipher and a one-way hash function. After each session s, a fog node Fo_l transforms collected data d_i^s into q fragments $f_i^{s1}, ..., f_i^{sq}$. The fog distributes the $(q-1)$ fragments over $(q - 1)$ of its neighbors, relying on a selected replication policy, keeping one of the fragments for

TABLE I: Variants comparison

	Fragment Size	**Key Management**	**Operation Base**	**Revoking Authority Optimized**	**Data Deduplication**	**Security Services**	**Computational Overhead**	**Additional Performance Optimization**
Secret Sharing Shamir	$\|D\|$	Keyless	Polynomial	No	No	Data Availability	None	
IDA	$(\frac{\|D\|}{k})$	Keyless	Polynomial	No	Yes	Data Availability+ Weak DC	-	-
Krawczyk	$\frac{n}{m} + Kn$	SSSS	Polynomial	No	No	Availability	Matrix Diffusion + Encryption + PRNG(1) + Polynomial Diffusion	None
AONT+IDA	$(\frac{n}{m} + \frac{H(C)+K}{m})$	$(\oplus)$with data	Polynomial	No	No	Data Availability	Matrix Diffusion + Encryption + PRNG(1) + Hash	None
AONT-RS	$(\frac{n}{m} + \frac{H(C)+K}{m} + \frac{H(c)}{m})$	$(\oplus)$with data	Polynomial	No	No	Availability	Matrix Diffusion + Encryption + PRNG(1) + Hash	Systematic IDA
CAONT-RS	$(\frac{n}{m-1})$	Keyless	Polynomial	No	Yes	Data Availability	Matrix Diffusion + Hash + Hash	Systematic IDA
IDA + Over Encryption	$(\frac{n}{m})$	NA	Polynomial	Yes	No	Availability	Matrix Diffusion + Encryption + Encryption	Partial Encryption
Salted IDA	$(\frac{n}{m} + \frac{S}{m})$	NA	Polynomial	No	No	Availability	Matrix Diffusion + Encryption + PRNG(i)	None
IDA-XOR	n	Keyless	Boolean	No	No	Availability	Matrix Diffusion + PRNG(t-1) + XOR	None

itself. Note that each fog node Fo_l can be associated to one or more clouds C_k, where $k \in [1, z]$, such that Fo_l assigns collected and processed data to related authorized clouds.

B. Threat Model

For designing suitable security mechanisms for fog networks, we consider two kinds of adversaries, namely *honest-but-curious*, and *malicious* fog nodes, defined as follows:

- *Honest-but-curious* adversary: it provides proper inputs or outputs at each step of the security scheme, and properly performs any expected calculations, but it may attempt to gain extra information about the system. As such, we consider the *honest-but-curious* threat model against the privacy and confidentiality requirements.
- *Malicious* adversary: malicious users may attempt to deviate from the security scheme, to provide invalid data or to delete valid data. In addition, a malicious party can also influence other parties to act maliciously by substituting their local inputs. Moreover, a malicious user can impersonate other legitimate users to get access to the system as a legal entity. Note that a malicious insider has the ability to read, modify or destroy the data by employing compromised fog nodes. However, during an application session s, it cannot destroy or get access to more than $(n-k)$ fog nodes and cannot intercept more than $(k-1)$ communications between the fog nodes. As such, we consider the malicious user threat model mainly against the confidentiality, availability, integrity and message authentication

TABLE II: Summary of notations

Notation	**Definition**
Dev_i	i^{th} IoT device
Fo_j	j^{th} Fog node
s	an application session during which an IoT device generates data
d_i^s	data to be fragmented, collected by Dev_i in session s
SK	static key stored inside an IoT device
DK	dynamic key derived from the static key and Nonce in a nonlinear manner
DK_C	dynamic key used inside the fragmentation procedure
DK_A	dynamic key used for authentication
f_i^{rj}	jth fragment of data collected by Dev_i during session s
n	total number of fragments
k	Threshold (number of fragments required for recovery)
SK	Secret Key
N_o	Nonce
DK	Dynamic Key
K_{GM}	sub-key used for the generation of matrices
K_{SRM}	Selection matrices sub-key used to construct the selection matrix table π_{SRM}.
π_{SRM}	selection table with nr elements length. It is used to select which IDA matrix will be used during each input sub-matrix.
K_{USRM}	sub-key used to construct the update (permute) the selection permutation table $u\pi_{SRM}$.
$u\pi_{SRM}$	Update selection table and it has nr elements length. It is used to update the selection table π_{SRM} for each new input data
IV_S	Initial Vector sub-key
IV_{IA}	Initial Vector for Integrity-Authentication
IV_E	Initial Vector sub-key for encryption
K_{SRF}	Selection sub-key used to construct the update selection table π_{SRF}.
π_{SRF}	Controls the distribution of the encrypted sub-fragments over the n fog nodes.
K_{USRF}	Update selection sub-key used to construct the update selection table π_{USRF}.
π_{USRF}	Update selection table and it has n elements length. It is used to update the selection table π_{SRF} for each new input data.
K_{IA}	Sub-key used for the message Integrity-Authentication
K_S	Sub-key used for generation of the selection tables
K_E	Sub-key used for the encryption of the original data
X	the produced filtered keystream
x_i	the i^{th} block of X and it is used to construct the G_i IDA vandermode matrix
G	A set of m dynamic IDA matrices
$G(i)$	The i^{th} dynamic IDA matrix
$G_k(i)$ and $G_k^{-1}(i)$	a $k \times k$ of the i^{th} dynamic IDA matrix ($G(i)$) and its corresponding inverse matrix.
D	Original application data
$\|D\|$	size of the original data I
data chunk	k consecutive bytes of permuted data
data share	an encoded data chunk with length n bytes
fragment	a final data fragment, which represents the same column of all the data shares and is stored in one location storage entity
k	number of fragments required for data recovery
nr	number of data chunks (original blocks) inside initial data
$n, n \geq k$	total number of fragments
DC_i	i^{th} data chunk set (original block), a set of k bytes of data chunks
DS_i	i^{th} data share set (encoded block), a set of n bytes of data shares

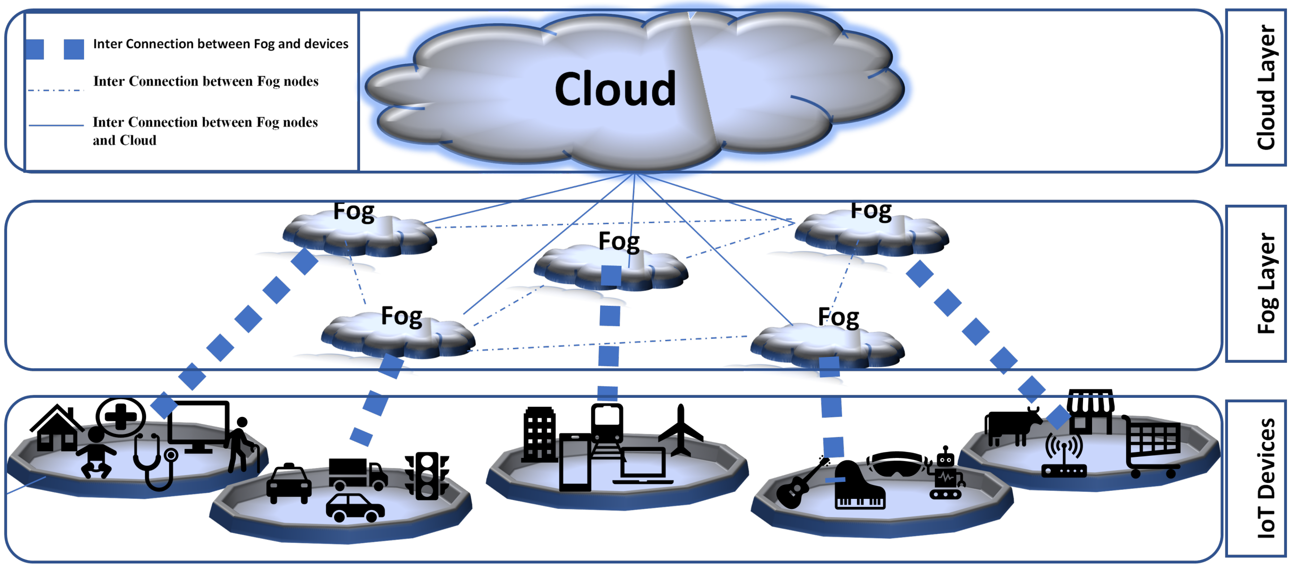

Inter Connection between Fog and devices
Inter Connection between Fog nodes
Inter Connection between Fog nodes and Cloud
Cloud
Fog
Fog
Fog
Fog
Fog
Cloud Layer
Fog Layer
IoT Devices

Fig. 1: Network Model

properties.

C. Design Goals and Evaluation Metrics

Our aim is to provide a fog architecture that is able to secure the received data from IoT devices. The fog layer introduces major enhancement in terms of performance, but suffers from security and privacy issues. The process of securing data should minimize the data processing cost, storage and transmission overheads, as well as maximize data survival. Several metrics are taken into consideration while evaluating the scheme characteristics such as storage overhead (ST), transmission cost (TC), and data resilience (DR); these help to compare the proposed scheme against the existing ones. The storage overhead describes the data overhead stored in addition to the collected data. The transmission cost measures the amount of data that has to be transmitted among fog nodes. Data resilience measures the ability to recover the initial data after destroying/altering parts of the processed data.

V. Proposed Key Derivation Scheme

In this section, we describe the derivation scheme of the dynamic key and the sub-keys. The generation process is illustrated in Fig. 2, and the function inputs are described as follows:

- **Secret key** SK**:** the secret key is shared among the fog nodes. This secret is changed for each application session. The renewal of this key can be performed in different ways, one of which is the Binary Elliptic Curve Diffie Hellman protocol (ECDH) [35]. The length of SK can be equal to 128, 256 or 512 bits. In fact, different possible solutions can be used to distribute secret keys among the nodes such as through a Key Distribution Center (KDC), which requires that each fog has a shared master secret key with the KDC. However, the proposed solution is based on the concept that each fog has its own secret key that it is not shared with any other fog node. The data about the secret key is protected and stored at each fog node, and it cannot be recovered from the node. As such,

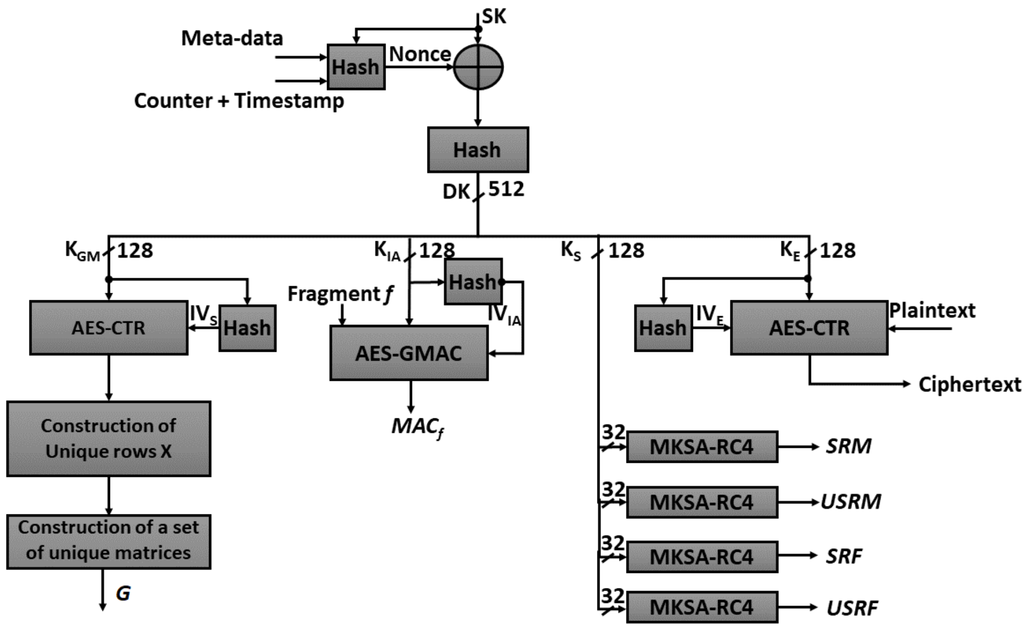


Fig. 2: Proposed key derivation function and generation process of the cipher primitives and their updates

any attack on a fog node will not lead to any security issue since data is protected and well distributed. Moreover, and to further enhance the security, a dynamic key-dependent cryptographic solution is used and it employs the fog's secret key and another variable unique meta-data to produce different dynamic keys in a lightweight manner. We might need to introduce one of the existing key recovery mechanisms to avoid any key management issue.

- **Nonce** N_o**:** the nonce is generated using a pseudo-random number generator. For each application session, a new N_o is generated, and it is associated with the application data at the fog node and thus, for the reverse process, the fog node can retrieve it.

To generate the dynamic key, the secret key SK and nonce N_o are XOR-ed and the output is hashed. This results in the dynamic key, $DK = h(SK \oplus N_o)$, which has a size of 64 bytes, and h represents a cryptographic hash function such as SHA-512.

Next, the dynamic key is divided into four sub-keys to be used in the different cipher primitives. This enhances the security of the cipher scheme since different keys are used at different stages. In our proposed solution, we choose to apply SHA-512 since it is resistant against collision attacks. The dynamic key and its corresponding sub-keys are randomly generated at every application session; a change of one bit in the $Nonce$ would result in a key that is completely different from the previous key. This dynamic scheme enhances the key security and makes the cipher immune against attacks.

A. Dynamic Key & Sub-keys Derivation

In this sub-section, we describe the generation of the dynamic key and its corresponding sub-keys (see Fig. 2). The meta-data represents certain parameters that describe the message such as length and its corresponding source. This factor is introduced to increase the security level of the proposed dynamic key derivation. The frequency of the key generation is determined by the application's session time. We employed SHA-512 for generating the dynamic key, which is then divided into four sub-keys. DK is of size 512 bits and each of the sub-keys

$\{K_{GM},\ K_{IA},\ K_S,\ K_E\}$ is of size 128 bits, and these are constructed as follows:

- **Generation of Matrices Sub-key** K_{GM}: this sub-key is used to generate the fragmentation matrices; it consists of the most significant 16 bytes of DK.
- **Integrity and Authentication Sub-key** K_{IA}: this sub-key is used for integrity and authentication checking; it consists of the next most significant 16 bytes of DK.
- **Selection of Matrices Sub-key** K_S: this sub-key is used for generating the selection/update matrices for fragments distribution over the fog nodes; it consists of the third most significant 16 bytes.
- **Encryption Sub-key** K_E: this sub-key is used for data encryption; it consists of the least significant 16 bytes of DK.

Similarly, the selection key K_S is divided into four sub-keys, as shown below:

- **Selection Matrices Sub-key** K_{SRM}: this sub-key is used for generating the IDA matrices selection table; it consists of the most significant 32 bits of K_S.
- **Update Selection Matrices Sub-key** K_{USRM}: this sub-key is used for updating the IDA matrices table; it consists of the second most significant 32 bits of K_S.
- **Selection Fog Nodes Sub-key** K_{SRF}: this sub-key is used for generating the fog nodes selection table; it consists of the third most significant 32 bits of S_K
- **Update Selection Fog Nodes Sub-key** K_{USRF}: this sub-key is used for updating the fog nodes selection table; it consists of the least significant 32 bits of K_S

Table II shows all the notations used in this paper. Next, we describe the construction of the proposed cipher primitives that are based on these four sub-keys.

B. Construction of Cipher Primitives

In this part, we detail the proposed techniques to generate the required selection permutation tables and the invertible IDA matrices.

1) Dynamic Permutation Primitives: KSA-RC4 represents the key setup algorithm of RC4. It is used to produce a substitution table S with 256 elements in RC4. It has been modified in [8] to produce dynamic flexible permutation tables in addition to substitution ones. In this paper, it is used to construct selection tables based on the produced permutation tables.. AS illustrated in Algorithm 1, for an input of L bytes, this algorithm outputs a dynamic permutation table, P, of len elements.

According to [8], for $L \geq 4$, the obtained permutation table P exhibits robust security properties. Moreover, as P is bijective, the inverse P^{-1} can be computed by the equation P^{-1}[P(i)]=i; where $P(i)$ is the i^{th} element of P.

2) Dynamic Selection Sub-matrices: As shown in Fig. 2, the K_S sub-key is used to generate the selection/update tables, which is based on the proposed $MKSA-RC4$ algorithm. Specifically, π_{SRM} is generated using the K_{SRM} sub-key. In this case, $\pi_{SRM}(i)^{th}$ serves in selecting the i^{th} IDA matrix for the i^{th} encrypted data block. π_{SRM} consists of nr elements, with $1 \leq \pi_{SRM}(i) \leq m$, where m represents the produced IDA matrices and $m \leq nr$. At the decryption phase, π_{SRM} serves in selecting the inverse IDA matrix $G_k^{-1}(\pi_{SRM}(i))$. In fact, π_{SRM} controls both the modified and the inverse IDA processes. Moreover, another permutation table π_{USRM} is generated using k_{USRM} sub-key to update the permutation table at each application session.

3) Dynamic Fragments Distribution : On the other hand, another permutation table is needed for the data fragments distribution over the corresponding fog nodes. π_{SRF} is generated using the k_{SRF} sub-key. This table consists of n elements and controls the distribution of the encrypted sub-fragments over the n fog nodes. Similarly to π_{USRM}, an updated table for the fragments distribution table π_{USRF} is generated using k_{USRM}. Consequently, the order in which the fragments are distributed over the neighbor nodes is changed at every application session.

4) Dynamic Key-Dependent Pseudo Random IDA Matrices: As mentioned before, to generate the IDA matrices, the k_{GM} sub-key is used. The proposed IDA-based scheme requires m matrices, where each matrix consists of a row with n distinctive and

Algorithm 1 Proposed modified KSA for RC4

```
1: procedure MKSA($K = \{k_1,\ k_2,\ \dots,\ k_L\}, L, len$)
2:     for $i \leftarrow 0$ to 255 do
3:         $P\ [i] \leftarrow i$
4:     $j \leftarrow 0$
5:     for $i \leftarrow 0$ to $len$ do
6:         $j \leftarrow (j + S[i] + k[j \bmod L]) \mod len$
7:         $swap(S[i], S[j])$
8:     return $P$
```

non-zero bytes. To generate these matrices, AES in counter mode (AES-CTR) is applied to k_{GM} and the initial vector IV, where IV is obtained by hashing K_{GM}. The produced key-stream is used to build m invertible matrices. Thus, the n bytes of each row are checked; if redundant or zero values are encountered, the cipher process is repeated until having n distinctive non-zero bytes. Then, the obtained row is used to construct an $n \times k$ IDA Vandermonde matrix (using Vandermonde matrix form), which is used during the fragmentation process. Based on the Vandermode matrix properties, any k rows of any IDA Vandermonde matrix form an invertible $k \times k$ matrix. In fact, avoiding repeated and zero values in the n-element rows is a key requirement to construct the IDA invertible matrices.

VI. Proposed Cryptographic Solution

In this section, we present our proposed cryptographic solution consisting of encrypting the data, fragmenting it, and distributing it over n fog nodes. Figure 3 illustrates an outline of the proposed data confidentiality-Availability-Integrity solution. Furthermore, the inverse process is not detailed since it consists of the same operations with minor changes (e.g. the use of the inverse matrices in the multiplication operations). For consistency, the used notations are included in Table II.
Essentially, the encryption is based on the symmetric scheme employing a secret key SK. This key is the base for generating the dynamic key with all the corresponding sub-keys. The values of n and k depend on the required availability level, topology of the fog system, and application requirements. Therefore, $(n-k)$ can be adjusted according to these parameters. In general, a higher availability level requires the increase of $(n-k)$, but at the expense of increased storage and communication overhead. The values of n and k should be selected to achieve the required balance between availability and performance.

A. Encryption Process

The encryption process is based on AES with Galois/Counter Mode (GCM), which is an authenticated encryption operation mode. GCM uses the counter (CTR) operation mode for encryption/decryption, which can be considered as a stream cipher. In addition, GCM uses the GMAC algorithm for the message authentication process. GCM is selected since it exhibits higher performance and efficiency compared to the CCM mode. Moreover, the advantage of using GCM in the context of fog is that it can ensure a high throughput with reasonable hardware resources.

B. Proposed Data Availability Process

The proposed IDA-based algorithm (see Algorithm 2) takes as input an encrypted vector of data D. This vector is fragmented into a set of nr ($\lceil \frac{|D|}{k} \rceil$) blocks $C = C_1\ , C_2,\ \dots\ , C_{nr}$, where each block has a length equals to k elements. A padding operation is introduced if necessary to complete the last block.

Then, the encoding process consists of a multiplication operation of a k block and a chosen IDA matrix (a set of key-dependent dynamic IDA matrices were generated).

As per its definition, the permutation table $\pi_{SRM} = [\pi_{SRM}(i)]_{1 \leq i \leq nr}$, which has nr elements is used to select the dynamic encoding IDA matrix.

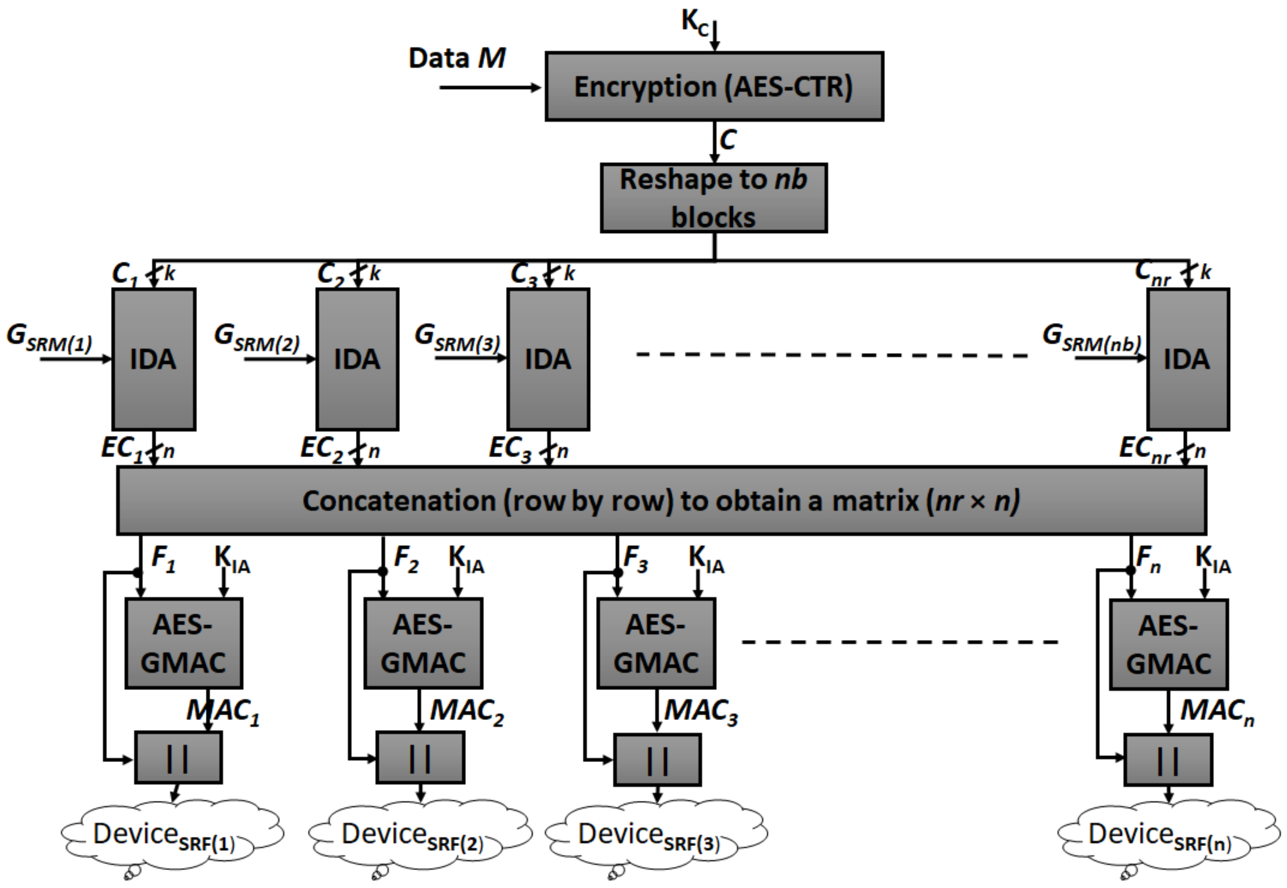


Fig. 3: Proposed cryptographic solution

Note that $\pi_{SRM}(i)$ is an integer value between 1 and m, having only m IDA matrices. Additionally, $m \leq nr$, so an IDA matrix can be used to encode more than one sub-matrix.

In the presented algorithm, $G = GS(\pi_{SRM}(i)$ means that the $\pi_{SRM}(i)$ IDA matrix is used. In fact, G consists of m invertible matrices. The obtained result for each encoded process is an encoding block that has a size of n elements. Then, all these encoded blocks are concatenated to form a matrix with a dimension equals to $n \times (nr)$. Each row of this matrix represents a fragment f_i, with $1 \leq i \leq n$. This means that n fragments are the output of the encoding process. Furthermore, each fragment should be authenticated and its corresponding message authentication code should be concatenated to it before forwarding it to the corresponding fog.

Note that the value k is related to the security level of the proposed IDA-based algorithm. When k increases, the security level increases but at the cost of increased complexity. Note that k and n can be changed according to the possible number of neighboring fog nodes.

C. Data Authentication and Integrity Scheme

In the proposed scheme, the data authentication and data integrity are ensured by means of the hashing process. After fragmenting and encoding the initial data, AES_GMAC is applied to each fragment using the K_{IA} sub-key. The initial vector

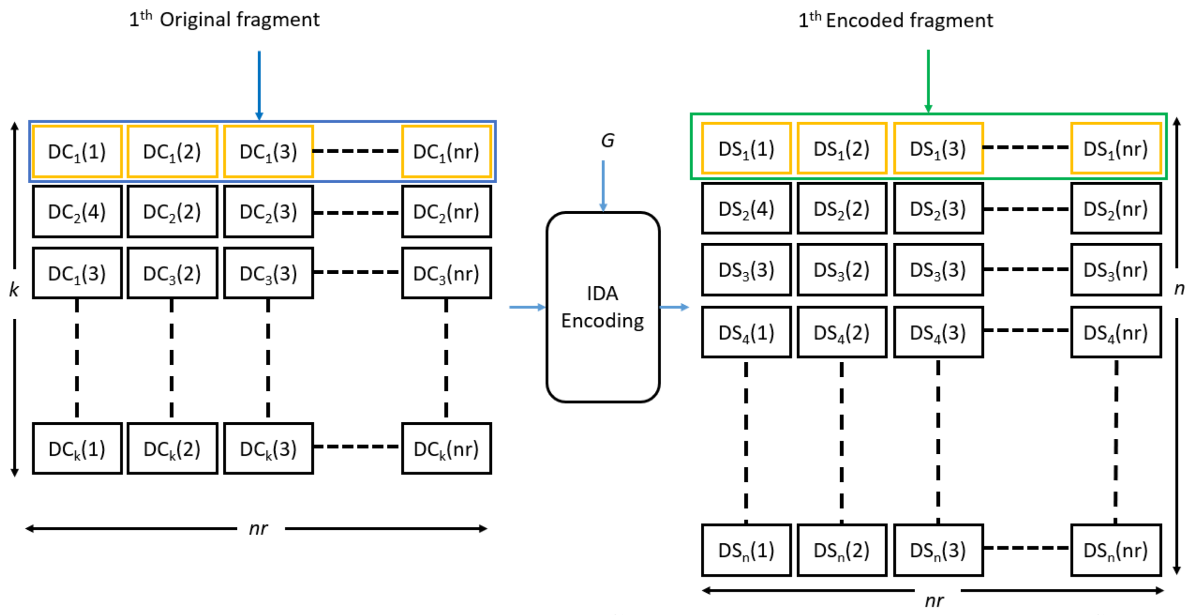

1th Original fragment
1th Encoded fragment
DC1(1) DC1(2) DC1(3) DC1(nr)
DC2(4) DC2(2) DC2(3) DC2(nr)
DC1(3) DC3(2) DC3(3) DC3(nr)
DCk(1) DCk(2) DCk(3) DCk(nr)
k
nr
G
IDA Encoding
DS1(1) DS1(2) DS1(3) DS1(nr)
DS2(4) DS2(2) DS2(3) DS2(nr)
DS3(3) DS3(2) DS3(3) DS3(nr)
DS4(1) DS4(2) DS4(3) DS4(nr)
DSn(1) DSn(2) DSn(3) DSn(nr)
n
nr

Fig. 4: Example for k: Encoding of the data chunks DC_i (left) is transformed into n data shares DS_i (right) and $i = 1,\ 2,\ \dots,\ nr$.

Algorithm 2 Proposed fragmentation algorithm outline.

```
1: Input: encrypted sub matrices($C = \{C_1\ , C_2,\ \dots\ , C_{nr}\}$), $G$, $\pi_{SRM}$
2: Output: $EC = \{EC_1,\ EC_2,\ \dots\ , EC_{nr}\}$
3: for $i = 1 \rightarrow nr$ do
4:     $G_i = G(\pi_{SRM}[i])$
5:     $EC_i = G_i \odot C_i$
6: $EC = EC_1||EC_2||\dots||EC_n$
7: for $i = 1 \rightarrow n$ do
8:     for $j = 1 \rightarrow nr$ do
9:         $f_ij = EC_{i,j}$
```

required by AES_GMAC is also derived from K_{IA} by hashing. Consequently, for each fragment f_i, a signature MAC_i is generated, which is concatenated with the corresponding fragment f_i, before dispersing the fragments over the n fog nodes.

In the proposed scheme, the authentication process is not applied directly after encryption, instead after the IDA encoding step. In fact, the proposed scheme uses the CTR mode to ensure data confidentiality and the Galois Message Authentication Code (GMAC, which is GCM authentication-only variant) to ensure data integrity and source authentication, in addition to IDA that ensures data availability.

Let us indicate that GCM, and consequently GMAC, accept arbitrary lengths for the initialization vectors. In addition, GCM can be implemented in parallel and it can be more efficient by using pipeline instructions or hardware pipeline.

D. Inverse Cryptographic Solution

At the stage where a fog node needs to recover the stored data, upon a request from a connected user, the inverse cryptographic process is performed. We do not detail the operations as we did for the encryption and information dispersal processes, given that only a few changes are introduced. In the following, we summarize the main points in the decryption process. Note that all the cryptographic and permutation primitives with their corresponding inverses can be regenerated at the fog node by using the dynamic key associated with the stored data fragment.

1) First, k fragments are collected from k different fog nodes based on the inverse of the π_{SRF} permutation table.
2) Then, data integrity and source authentication are verified for each fragment. The verification consists of generating the MAC of each fragment and comparing it to the one associated with the fragment. The fragment hash is computed by applying AES_GCMA with the K_{IA} sub-key.
3) Next, the verified fragments are concatenated in a matrix, where each row of the obtained matrix represents k bytes of the data share. Then, the data share is multiplied by the inverse IDA matrix, which is selected based on the inverse of the π_{SRM} permutation table. After de-fragmenting all the encoded blocks, a matrix of $nr \times k$ is obtained. The rows of this matrix are concatenated to form a vector representing the initial data.

VII. Security Analysis

In this section, we describe the security evaluation of the proposed scheme. To this end, we apply it with a set of $10,000$ data instances of size $15,000$ bytes each. The security tests include the randomness evaluation of the obtained data fragments, and the generated dynamic keys. We consider different statistical properties including randomness, correlation, independence, and uniformity.

A. Randomness of Dynamic Keys

A randomness test is performed on the generated dynamic keys. The purpose of using dynamic keys is to avoid key disclosure, given that the time needed to perform a brute force attack would be greater than the session time. Thus, at the next session, the key is changed, and knowing the previous key, the attacker can disclose only the previous session's data. Note that in this case, the attacker needs to know all the fragmentation scheme primitives with the nodes locations to have access to all the nodes and to know the fragments order.

B. Randomness of Fragments

The chosen plain-text attack considers the outputs of different inputs, and thus the similarity test enables the disclosure of the encryption key. A key requirement for a secure encryption scheme is the randomness of the output. Also, given that the fragmentation scheme adds redundancy to ensure data availability, the redundant data should be distributed randomly among the different fragments. In this context, different statistical tests have been applied to the obtained fragments including recurrence, independence and uniformity tests. These tests with the obtained results are detailed next.

1) Recurrence: This test consists of measuring the variation among the values of a sequence. After fragmentation, a session input data can be written as a sequence: $x_i = x_{i,1},\ x_{i,2},\ \dots,\ x_{i,m}$. In Fig. 5, the variation between the original and encrypted fragments (first fragment) is shown. The results prove that the proposed scheme hides any clear pattern and hence, it is very difficult to get useful information regarding the original message.

2) Independence: The correlation analysis aims at measuring the dependence between two variables. In our case, the attacker tries to infer any relation between the encrypted and original data [8]. Thus, the independence of the fragments and the original data is a key requirement for a secure encryption scheme. To this end, the correlation coefficients between the original fragments and their corresponding encrypted ones are measured, and the correlation coefficient between the different encrypted fragments is presented. Fig. 6-a) shows the results of the Empirical Cumulative Distribution Function (ECDF) of the correlation test between the obtained fragments and the original ones, considering the fragmentation of $10,000$ data instances with $n = 8$ and $k = 4$. The obtained correlation results are very close to the average value, which is close to 0. This means that the fragments are independent of the original data. In parallel, we measure the percentage of the difference between the original message and a set of k encoded fragments (encrypted) at the bit level. The results, presented in Fig. 6-b), show that the ECDF of the difference percentage is very close to the ideal value (50%) with a mean value equals to 49.99% and a low standard deviation of 0.144. These results confirm that the proposed scheme ensures the independence between the original and the encoded fragments at the bit

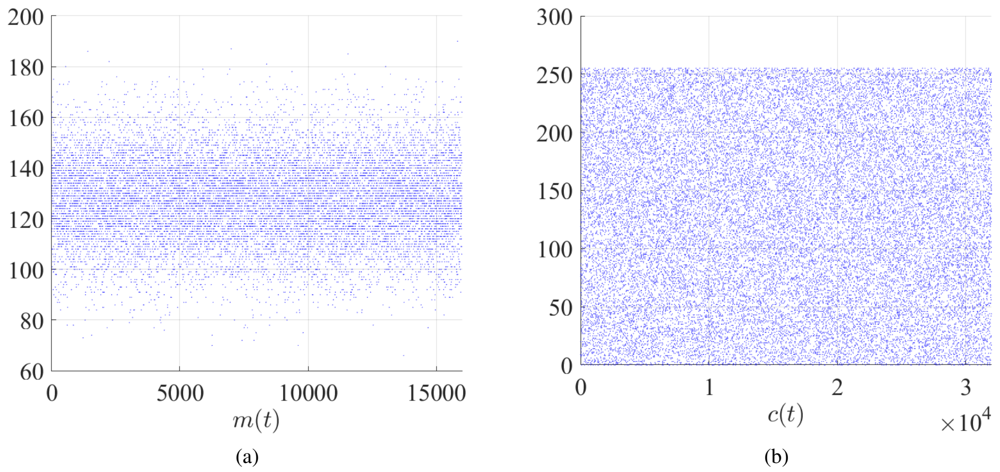


Fig. 5: Plots of original data and one of its corresponding encrypted fragmented ones, (k=4 and n=8).

level. In addition, a numerical example of the inter-correlation and bit difference percentage in a matrix form between the original and obtained fragments is presented in TABLE IV and TABLE III for a random dynamic key, respectively. The results show that the inter-correlation is always close to 0 for the different original and obtained fragments, which means that the encrypted fragments are independent of the original ones.

3) Uniformity: Another statistical property, which ensures the security of the proposed scheme, is the uniformity of the obtained fragments. In other terms, statistical attacks try to extract any pattern based on the distribution/frequency of the fragments. Thus, having a uniform distribution of the produced fragments prevents the attacker from acquiring any partial information about the proposed scheme. In Fig. 7, the distributions of the original data and its corresponding fragments are presented. This distribution validates visually that the produced fragments satisfy the uniformity property.

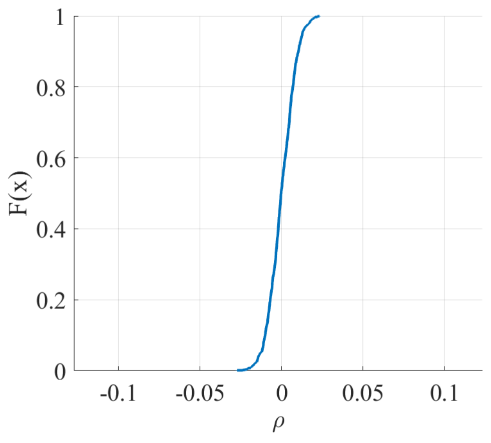
C. Sensitivity to Dynamic Key

In a keyed cryptographic scheme, the key security is vital to avoid key-related attacks. In this context, the scheme sensitivity to the dynamic secret key, DK, is required. In other terms, a small change in the key should result in two completely different encrypted messages. In our case, the dynamic keys used for two different sessions should give different fragments. Accordingly, we consider two dynamic secret keys: DK_w and DK'_w that differ in only one random bit to be applied to the same message, where $w = 1, \ \ldots, \ 1,000$. The sensitivity test for the w^{th} dynamic key ($DK^{'}_{w}$) is calculated as follows:

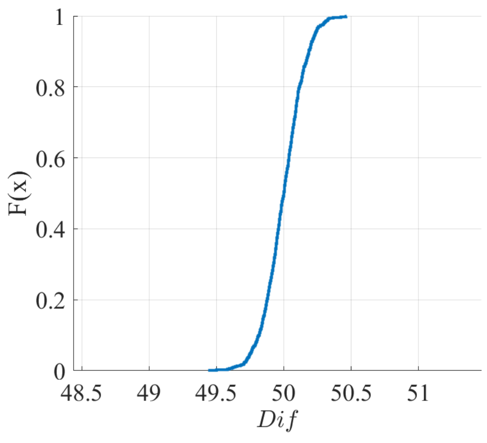
$$KS_w = \frac{\sum_{it=1}^{T} FE_{DK_w} \oplus FE_{DK'_w}}{l} \times 100\% \quad (1)$$

Where FE represents the proposed cryptographic scheme (encryption+modified IDA), T is the length of the input data in bits.

We run the sensitivity test for 1,000 iterations. At each iteration, different messages and different keys are used, with the condition to have one bit changed between the two considered keys. As shown in Fig. 8, the PDF of the obtained key sensitivity percentages indicates that the obtained values are very close to the mean value, which is 50.58%, and the standard deviation is very low and equals to 0.141. The results show that the proposed scheme exhibits a high dynamic key sensitivity. In addition, for a random key, the internal percentage difference among encoded fragments is shown in TABLE V, which shows that



(a)



(b)

Fig. 6: Correlation coefficients between original and fragmented data(a). In addition, the difference between original and encrypted fragmented data (% of changed bits) (b) for k=4 and n=8.

TABLE III: Correlation coefficient between the original and the encoded/encrypted segments for a random dynamic key with k =4 and n =8

	ED_1	ED_2	ED_3	ED_4	ED_5	ED_6	ED_7	ED_8
O_1	0.0185	-0.0139	-0.0398	0.0047	-0.0034	0.0129	0.0067	0.0006
O_2	0.0226	-0.0167	0.0207	-0.0177	-0.0084	-0.0209	0.0017	-0.0229
O_3	-0.0169	0.0248	-0.0404	-0.0024	0.0094	0.0276	-0.0302	0.0210
O_4	-0.0133	-0.0090	0.0322	0.0026	-0.0136	-0.0059	-0.0101	-0.0075

TABLE IV: The percentage difference between the original and the encoded/encrypted fragments for a random dynamic key with k =4 and n =8

	ED_1	ED_2	ED_3	ED_4	ED_5	ED_6	ED_7	ED_8
O_1	0.4995	0.4953	0.5003	0.5037	0.4997	0.5031	0.4922	0.5052
O_2	0.4992	0.4964	0.4977	0.4985	0.5053	0.5042	0.4938	0.5017
O_3	0.4959	0.5042	0.4962	0.4982	0.4963	0.5013	0.5035	0.5013
O_4	0.5027	0.4994	0.4970	0.5014	0.4999	0.4978	0.5031	0.5016

TABLE V: Key sensitivity test between two fragmentation results obtained for the same data but with two slightly different keys (Dk and DK') with k =4 and n =8

	ED_1	ED_2	ED_3	ED_4	ED_5	ED_6	ED_7	ED_8
ED'_1	50,17	49,83	49,82	50	49,41	50,23	49,77	49,69
ED'_2	50.05	49,90	50,19	50	50,04	49,91	49,84	50,07
ED'_3	50,12	49,83	50,09	50,08	49,75	50,30	49,67	50,02
ED'_4	50.02	50,22	50,05	50,50	50,09	50,64	49,61	49,64
ED'_5	50,11	50,57	49,93	50,19	49,65	49,58	50,44	49,91
ED'_6	50,41	49,70	49,76	49,69	49,96	49,8	49,54	50,54
ED'_7	50,34	49,90	50,08	49,64	50,25	50,64	50,23	49,88
ED'_8	50,15	49,55	49,72	49,80	49,78	49,58	50,31	49,92

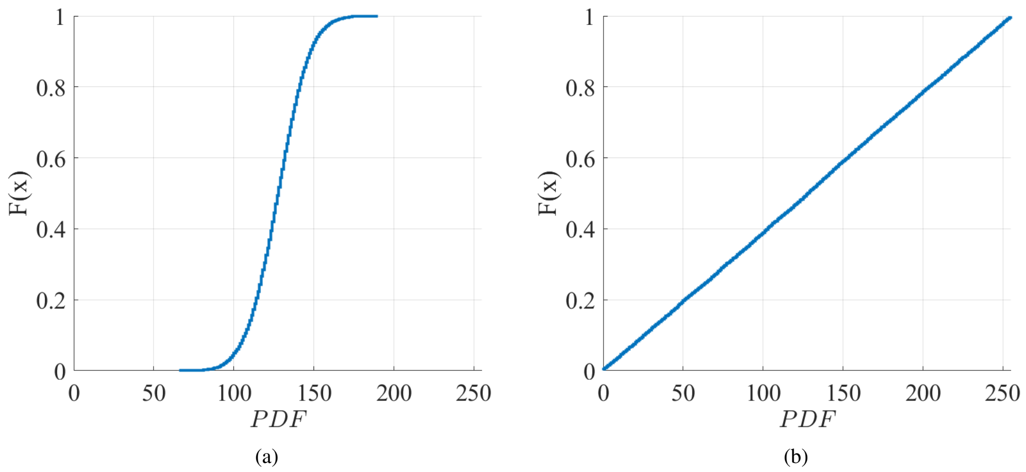


Fig. 7: The distribution of an original message (a) and its correspondent encrypting encoded ones (k=4 and $n = 8$)

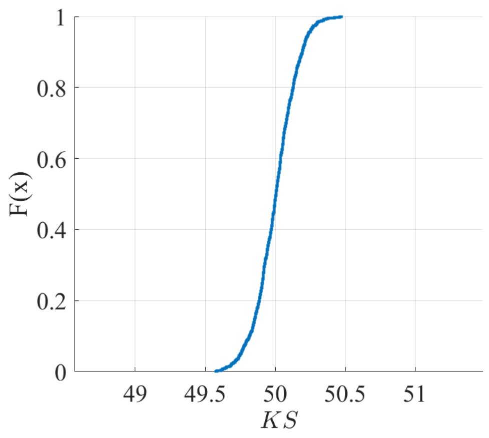


Fig. 8: Key sensitivity test measuring the bit difference between two fragmentation results obtained for the same data but with two slightly different keys for 1000 iterations

a high difference is always reached between any pair of encoded fragments. Also, the correlation test is applied on each pair of encoded fragments and the obtained results confirm the independence among encoded fragments as shown in TABLE VI.

VIII. Cryptanalysis Discussion

The main components that ensure the robustness of the proposed scheme are 1) the use of the dynamic key approach, 2) the use of the (AES) block cipher with (GCM) mode, and 3) the use of dynamic key-dependent secret invertible coding matrices. In fact, the use of the dynamic key approach ensures the backward and forward secrecy. In other terms, if the attacker discloses the data of one session by compromising the used key, he cannot disclose neither the previous nor the next data. Moreover, the dynamic key approach is used to generate all the cipher and fragmentation primitives, in addition to the hashing process. In summary, the proposed scheme presents a complete data security framework for fog systems.

In fact, the proposed scheme ensures data security via the dispersal of data over n different fogs (or clouds). The difficulty of collecting t fragments from the n dispersed ones is a challenging task that the security of the proposed scheme relies on. Moreover, the attacker should know the right order of the fragments, the secret de-fragmentation matrices and the used dynamic key.

TABLE VI: Correlation coefficient among encoded/encrypted fragments for a random message dynamic key with k =4 and n =8

	ED_1	ED_2	ED_3	ED_4	ED_5	ED_6	ED_7	ED_8
ED_1	-							
ED_2	0.0093	-						
ED_3	-0.0037	-0.0074	-					
ED_4	0.0249	0.0270	0.0018	-				
ED_5	0.0208	-0.0393	0.0048	-0.0262	-			
ED_6	-0.0099	-0.0118	-0.0062	0.0070	-0.0074	-		
ED_7	0.0097	-0.0278	0.0092	0.0086	-0.0267	0.0330	-	
ED_8	-0.0075	0.0037	0.0086	-0.0244	0.0015	0.0088	0.0018	-

Next, we describe the possible attacks that can be performed when the attacker gets access to k fragments with $n \leq t$.

A. Frequency Analysis Attacks

Frequency analysis or ciphertext-only attacks consist of analyzing the ciphertext to find the recurrence of certain patterns. This type of attacks relies on the fact that each type of data (text, image, video) might present some statistical properties that might be conserved by the cipher scheme. Thus, knowing this pattern, the attacker can reveal partial information about the cipher scheme. However, the performed uniformity test shows that the output of the proposed scheme is uniform and thus, the frequency analysis cannot reveal any information. Consequently, the proposed scheme is robust against frequency analysis attacks.

B. Brute-force Attacks

Brute-force attacks are mainly performed in order to disclose the secret key. This can be done by trying all the key possibilities to get the correct result based on known/chosen plain/cipher texts pairs. In this context, the length of the key plays an essential role in hardening the attacker task. In fact, the time needed to crack a key increases exponentially with the key length. However, in our case, using a dynamic key makes the attacker task much harder. The attacker has to know the key during a session to be able to disclose the original data, in addition to compromising t channels and having the knowledge of all the scheme primitives. However, knowing the key for one session, will not give the attacker the privilege to disclose neither the previous processed data nor the future ones.

C. Linear and Differential Attacks

Linear cryptanalysis aims to find a relation between the input and the output of a cipher scheme. The differential cryptanalysis aims at revealing a pattern between the difference of two plaintexts and the difference between their corresponding ciphertexts. However, the security tests have shown that the input and output of the proposed scheme are independent and that the output is random with uniform distribution. In addition, AES is immune to statistical attacks, which makes the proposed scheme robust against linear and differential attacks. On the other hand, the use of dynamic keys in each session makes the linear and differential attacks much harder. At each session, a new key is used, and as shown in the key sensitivity test, a change of one bit in the key results in very different cryptographic primitives and consequently, different encoded fragments for the same plaintext.

D. Sybil Attacks

A Sybil attack is basically an authentication attack, and it can be prevented by using a strong authentication scheme. In the proposed solution, we assume that a strong authentication scheme should be adopted among fog nodes in a periodic manner and before exchanging data fragments. The scheme can be based on multi-factor authentication and recently, several solutions were presented such as the one in [33]. In this paper, the main focus is about the preservation of fog data and not the authentication of entities.

In fact, different cryptographic protocols that can validate that entities have shared knowledge (key). Such protocols can be based on trusted certification solution (asymmetric) or key distribution center (symmetric) to ensure that each fog entity is assigned exactly one identity. As such, a Sybil attacker will not be able to use multiple identities for authentication using the same shared key. Moreover, other authentication factors such as device fingerprint would make Sybil attacks even more complex to be conducted.

In summary, the cryptanalytic analysis confirms the robustness of the proposed scheme towards well-known cryptanalytic attacks.

IX. Performance Analysis

The security tests show that the proposed scheme that ensures the integrity, authentication, and confidentiality security services is robust against several security attacks. However, providing high level of security often comes at the cost of computational complexity, execution time overhead, and resource requirements. Achieving the appropriate trade-off between the security level and complexity is a main challenge in a fog system, which presents constraints in terms of storage and computational resources. In fact, the proposed solution is designed to reach a good balance between system performance and security level.

About the implementation, the proposed modified IDA solution (IDA) was realized in C, compiled with -O3 optimization option, while the optimized AES OpenSSL implementation is used with the GCM operation mode for message authentication and confidentiality. This operation mode exhibits relatively low computational complexity with respect to the provided security level (see Fig. 10).

In the following, we include the discussion and results of several performance elements including the cost of the proposed key derivation function and cryptographic primitives generation, execution time, storage/communication overhead, efficiency, and error propagation.

A. Computational Overhead

The proposed solution is based on the dynamic key-dependent approach, which is the main distinction when compared to the previous variants. Table 1 illustrates the required cost overhead of the previous information dispersal variants. Next, we describe the cost overhead of the proposed solution in terms of the required computations. To achieve a high security level with the minimum possible computational complexity, we are relying on the dynamic key approach.

The required key derivation function requires only one iteration, for one input block, of a secure hash function in addition to an xor operation to produce the dynamic key for each new session. Then, the obtained dynamic key is divided into a set of dynamic sub-keys that are used to generate the different sets of cipher primitives. The techniques used to produce the primary and update cipher primitives are lightweight and satisfy the required cryptographic properties. Moreover, for each new input message, the selection cipher primitives are updated in a lightweight manner by using update cipher permutation tables ($USRM$ and $USRF$). Accordingly, for each input message, the SRM and SRF tables are permuted (updated) based on the update permutation tables ($USRM$ and $USRF$) and consequently, different fragments are produced for the same message. Therefore, the proposed solution reduces the cost overhead of the proposed KDF and cipher primitives construction techniques. On the other hand, towards reducing the IDA multiplication overhead, we use a look-up table for the multiplication operation.

Moreover, in this paper, AES with GCM (authentication and confidentiality) operation mode is iterated with dynamic keys. In fact, AES is optimized at the majority of hosted devices [36]. Also, AES-GCM can be performed in parallel at the fragment level (t fragments for confidentiality and n fragments for message authentication). Moreover, the data availability is ensured by using the proposed modified AONT-RS fragmentation process. In general, the information dispersal algorithm produces n fragments, but in the proposed approach, it can also be performed in parallel at the sub-matrix level. This leads to a high security level with the minimum possible delay overhead. Indeed, these multiplication operations can be done in parallel and thus, the computational complexity, execution time and associated delays of the fragmentation process can be minimized. On the other hand, This leads also to a high security level with the minimum possible overhead since a set of m variable IDA matrices are used, for each input message, and they are unknown to attackers .

Therefore, all operations of the proposed cryptographic solution have been selected for low power consumption and delay. It can be considered as a lightweight solution and it can be adapted according to the fog limitations in terms of power, storage, and computations. The performance of the proposed solution may be optimized using parallel computations and optimized AES and IDA operations. While the high security levelis achieved using dynamic selection IDA primitives (SRM and SRF) for each new input message. Consequently, different cryptographic primitives are selected for each new input message.

As indicated previously, the proposed dynamic key derivation function, cipher primitives, and update cipher primitives construction techniques were designed to achieve a good balance between the security level and performance. In fact, the proposed solution is flexible; for the first variant, we can update the dynamic key for each input message, while the second variant uses update cipher primitives to renew cipher primitives for each new input message (during one session). Therefore, the second variant of the key generation technique does not require dynamic key and cipher primitives re-initialization. This means that the second variant requires low overhead compared to the first one.

B. Execution Time

This section is introduced to evaluate the execution time of 1) the proposed cryptographic scheme, 2) the dynamic key generation scheme and 3) the data confidentiality and message integrity procedure, where "OpenSSL" is used. The Computational delays of both, the key derivation function and the cipher primitives construction are described below. In addition, they are also quantified in order to assess the total associated delay as follows:

1) T_H denotes the required hash time for a block of h bytes.
2) $T_{MKSA}(x)$ denotes the required execution time of the modified KSA of RC4 for a table with h elements. It exhibits a low computational delay and it is used to construct the permutation tables.
3) T_{CIDA} denotes the required time to construct m IDA matrices, each with a size of $n \times k$.

$$CD_{KDF} = T_{xor} + T_H + 4 \times T_{MKSA} + T_{CIDA} \quad (2)$$

The Computational delays of the proposed cryptographic solution are described as follows:

$$CD_{proposed} = k \times T_{Encr} + n \times T_{Auth} + T_{IDA}(k, n) \quad (3)$$

where:

1) T_{Encr} denotes the required encryption time (AES-CTR) for a fragment of $\frac{|D|}{k}$ bytes length.
2) T_{Auth} denotes the required execution time of the message authentication algorithm (AES-GMAC) for a fragment of $\frac{|D|}{k}$ bytes length.
3) T_{IDA} denotes the required IDA diffusion time for a set of k input fragments that produce n output fragments.

In fact, the encryption and message authentication processes are applied in parallel at the fragment level. The length of each fragment is $\frac{|D|}{k}$ bytes.

Accordingly, the proposed solution is an efficient one based on the overall reduced overhead. Experiments have been performed on different Raspberry Pi devices that might play the role of a fog node. In the following, three different devices have been used, Raspberry Pi 0, 2 and 3, respectively called RPI0, RPI2 and RPI3. RPI0 has a 1GHz mono-core micro-controller with ARMv6 instructions and 512MB RAM. RPI2 has a 900MHz quad-core micro-controller with ARMv7 instructions and 1GB RAM. RPI3 has a 1.2GHz quad-core micro-processor with ARMv7 instructions and 1GB RAM. OpenSSL is a very popular tool and is widely used since it is considered as one of the most efficient cryptographic libraries that provide robust, commercial-grade, and full-featured toolkit for Transport Layer Security (TLS) and Secure Sockets Layer (SSL) protocols. Finally, AES OpenSSL is used to compare two authentication-encryption operation modes, CCM and GCM, which ensure data confidentiality and message integrity in addition to source authentication. In general, CCM (Counter with CBC-MAC) and GCM (Galois/Counter Mode) modes are preferred. Fig. 9 illustrates the numerical values of the data rate of CCM and GCM cryptographic operation modes using AES OpenSSL implementation with RPI0, RPI2 and RPI3.

The obtained results in Fig. 10 prove that the GCM mode is always more efficient than the CCM mode since it requires low execution time for the different AES (Advanced Encryption Standard) symmetric key sizes and RPI devices. As such, it is more suitable for fogs applications. It should be noted that as the message size increases, the time required to perform encryption and message authentication also increases.

The obtained data rate results clearly show that GCM can achieve a high throughput compared to CCM. Therefore, the GCM operation mode is used in the proposed solution instead of CCM. Consequently, the proposed solution will not degrade the main functionality of fog devices.

On the other hand, the efficiency of the proposed modified IDA scheme is also analyzed. The proposed IDA solution relies on an optimized Galois Field Arithmetic library scheme. TABLES VII, VIII, IX and Fig. 11 show the required average execution time for the proposed modified IDA in function of n and k, in addition to the input data size. According to the obtained results, increasing the file size leads to an increase of the required execution time. In addition, increasing n will also increase the required execution time for a fixed file size. Similarly, for fixed file size and n, increasing k also increases the required computation complexity and consequently the execution time. Fig. 11 indicates clearly that the variation of the execution time is linear ($O(filesize)$), which is desirable for constrained fog devices.

C. Storage/Communication Overhead

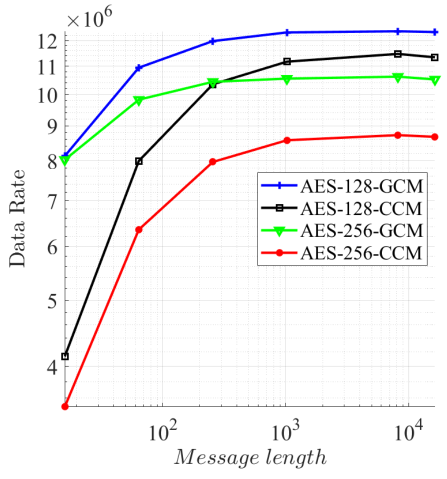
The storage and communication overhead are related to size of the redundant data and the number of fragments that need to be sent over the network. In other terms, this overhead is related to the values of n and k, where n is the number of fragments that should be sent to the neighboring nodes, and k the minimal number of fragmented needed to recover the original data. Having in total $n \times \frac{|M|}{k}$ fragments, where —M— is the original data size, the storage overhead is ($\frac{(n-k)\times|M|}{k}$). Moreover, the communication overhead is proportional to n. In case where n is close to k, the communication overhead decreases, however the data availability also decreases. In contrast, when k is less than n, the provided data availability is high at the cost of a higher communication overhead. Consequently, n should be chosen in a way to meet the compromise between data availability and communication overhead.

D. Efficiency and Error Propagation

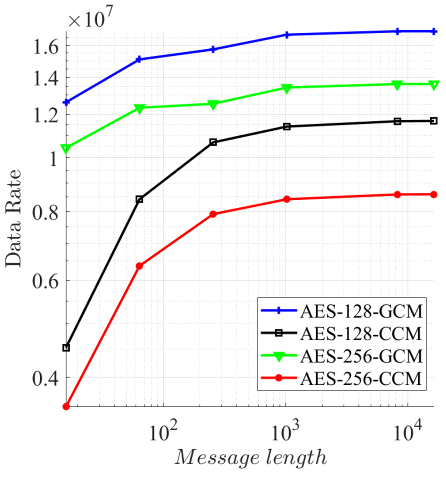
Another important requirement is to ensure low error propagation. If an error occurs at one fragment, the error should not result in the deterioration of the session data. Thus, in our proposed scheme, given that the decryption consists of multiplying each fragment by a decryption sub-matrix, the results of the correct fragments will not be affected and the error will be limited to the erroneous row/segment. In addition, requiring any t fragments to recover the original data, the erroneous can be replaced by another fragment and thus, the probability of recovering the original data correctly increases when $n > k$.

X. Conclusion

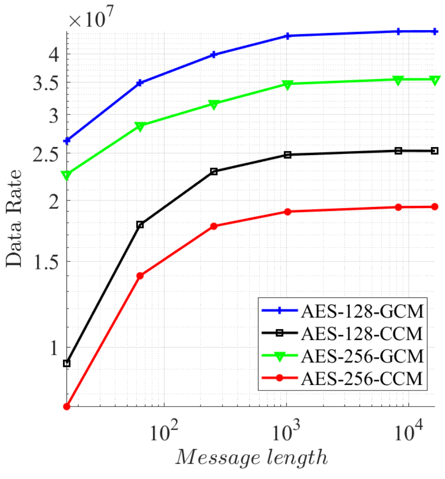
In this paper, a cryptographic scheme that jointly encrypts, authenticate and fragment input data is proposed for fog systems. To the best of our knowledge, this is the first work in this direction. The novelty of the proposed scheme is that it benefits from the dynamic key-dependent cryptographic scheme to enhance the security of fog systems, where a random nonce is produced and combined with a pe-shared secret key to generate a dynamic key. The produced dynamic key is used to derive the needed cryptographic primitives, which include encoding matrices and selection/update tables. Consequently, the encoding matrices become dynamic and secure and can lead to different fragments for the same input message during the same session. In addition, source authentication and message integrity and confidentiality are ensured by using AES with the GCM operation mode. Finally, security tests and performance analysis were presented to validate the safe and efficient deployment of the proposed scheme for fog systems.



(a) RPI-0

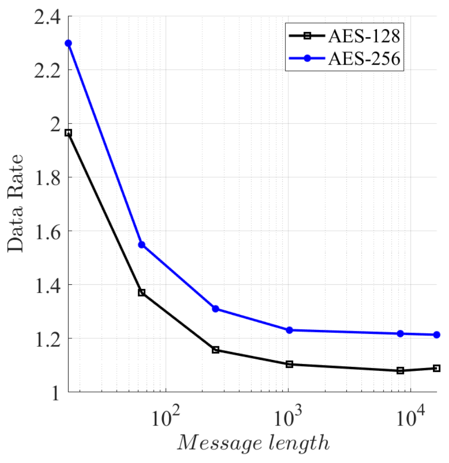


(b) RPI-2

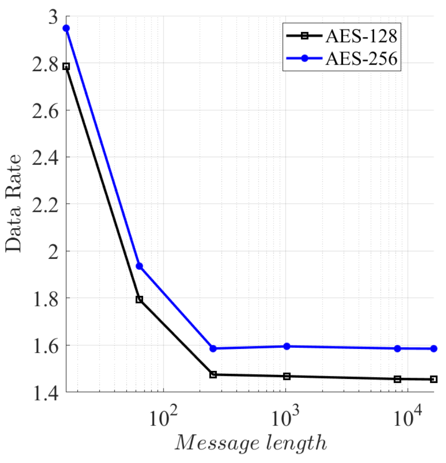


(c) RPI-3

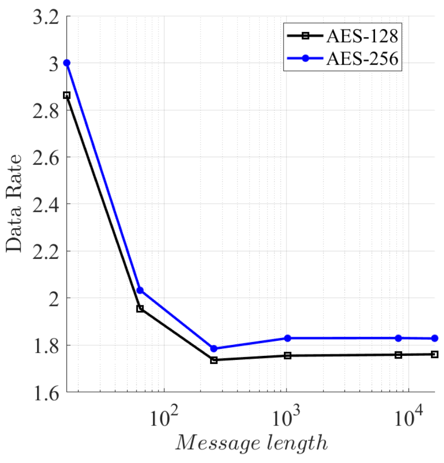
Fig. 9: Variation of data rate versus message length for the CCM and GCM operation modes with RPI-0 (a), RPI-2 (b) and RPI-3 (c)



(a) RPI-0



(b) RPI-2



(c) RPI-3

Fig. 10: The corresponding data rate ratio between GCM and CCM operation modes versus message length with RPI-0 (a), RPI-2 (b) and RPI-3(c)

Acknowledgement

This paper is partially supported with funds from the Maroun Semaan Faculty of Engineering and Architecture at the American University of Beirut and also from the EIPHI Graduate School (contract ”ANR-17-EURE-0002”). We also thank the supercomputer facilities of the Mésocentre de calcul de Franche-Comté.

TABLE VII: Execution times of optimized IDA approach with different sizes of messages and different values of fragments for recovery and fragments on RPI0.

Size	k	n	Exec. time (s)	k	n	Exec. time (s)
16KB	4	16	0.066	3	8	0.037
64KB	4	16	0.27	3	8	0.15
256KB	4	16	1.06	3	8	0.59
1MB	4	16	4.28	3	8	2.34
4MB	4	16	17.13	3	8	9.87
16MB	4	16	68.68	3	8	37.57
16KB	8	16	0.08	5	16	0.071
64KB	8	16	0.32	5	16	0.28
256KB	8	16	1.3	5	16	1.11
1MB	8	16	5.43	5	16	4.44
4MB	8	16	22.04	5	16	17.78
16MB	8	16	87.78	5	16	71.12
16KB	12	16	0.091	7	24	0.10
64KB	12	16	0.37	7	24	0.41
256KB	12	16	1.5	7	24	1.65
1MB	12	16	6.37	7	24	6.63
4MB	12	16	25.36	7	24	26.47
16MB	12	16	101.6	7	24	105.89

TABLE VIII: Execution times of our approach with different sizes of messages and different values of fragments for recovery and fragments on RPI2.

Size	k	n	Exec. time (s)	k	n	Exec. time (s)
16KB	4	16	0.058	3	8	0.032
64KB	4	16	0.26	3	8	0.13
256KB	4	16	0.94	3	8	0.51
1MB	4	16	3.76	3	8	2.03
4MB	4	16	14.78	3	8	8.15
16MB	4	16	59.66	3	8	32.6
16KB	8	16	0.073	5	16	0.06
64KB	8	16	0.28	5	16	0.28
256KB	8	16	1.11	5	16	0.96
1MB	8	16	4.51	5	16	3.93
4MB	8	16	18.17	5	16	15.49
16MB	8	16	72.98	5	16	61.98
16KB	12	16	0.08	7	24	0.09
64KB	12	16	0.33	7	24	0.36
256KB	12	16	1.31	7	24	1.43
1MB	12	16	5.24	7	24	5.76
4MB	12	16	20.82	7	24	22.93
16MB	12	16	83.32	7	24	91.71

References

[1] S. Tu, M. Waqas, S. U. Rehman, M. Aamir, O. U. Rehman, Z. Jianbiao, and C. Chang, "Security in fog computing: A novel technique to tackle an impersonation attack," *IEEE Access*, vol. 6, pp. 74 993–75 001, 2018.

[2] O. Salman, I. Elhajj, A. Chehab, and A. Kayssi, "Iot survey: An sdn and fog computing perspective," *Computer Networks*, vol. 143, pp. 221 – 246, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1389128618305395

[3] C. B. Tan, M. H. A. Hijazi, Y. Lim, and A. Gani, "A survey on proof of retrievability for cloud data integrity and availability: Cloud storage state-of-the-art, issues, solutions and future trends," *Journal of Network and Computer Applications*, vol. 110, pp. 75 – 86, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1084804518301048

[4] Y. Guan, J. Shao, G. Wei, and M. Xie, "Data security and privacy in fog computing," *IEEE Network*, vol. 32, no. 5, pp. 106–111, September 2018.

[5] B. A. Martin, F. Michaud, D. Banks, A. Mosenia, R. Zolfonoon, S. Irwan, S. Schrecker, and J. K. Zao, "Openfog security requirements and approaches," in *2017 IEEE Fog World Congress (FWC)*, Oct 2017, pp. 1–6.

[6] N. Kaaniche and M. Laurent, "Data security and privacy

TABLE IX: Execution times of our approach with different sizes of messages and different values of fragments for recovery and fragments on RPI3.

Size	k	n	Exec. time (s)	t	n	Exec. time (s)
16KB	4	16	0.034	3	8	0.019
64KB	4	16	0.14	3	8	0.075
256KB	4	16	0.55	3	8	0.30
1MB	4	16	2.19	3	8	1.21
4MB	4	16	8.83	3	8	4.86
16MB	4	16	37.37	3	8	21.12
16KB	8	16	0.041	5	16	0.036
64KB	8	16	0.16	5	16	0.14
256KB	8	16	0.66	5	16	0.58
1MB	8	16	2.66	5	16	2.3
4MB	8	16	10.7	5	16	9.2
16MB	8	16	44.97	5	16	37.01
16KB	12	16	0.048	7	24	0.053
64KB	12	16	0.23	7	24	0.26
256KB	12	16	0.77	7	24	0.85
1MB	12	16	3.09	7	24	3.44
4MB	12	16	12.36	7	24	13.8
16MB	12	16	49.86	7	24	55.18

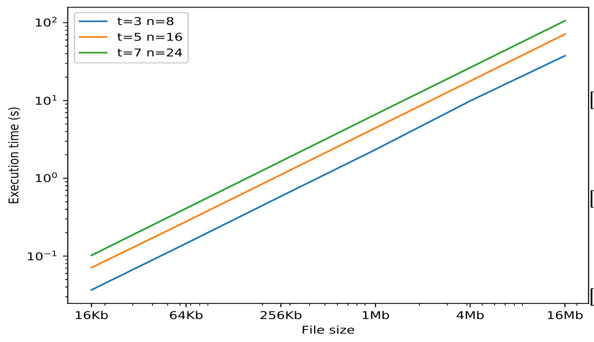


Fig. 11: Mean average of the required execution time (10000 times) of the proposed modified IDA algorithm with variable length message in addition to n and $k = t$.

preservation in cloud storage environments based on cryptographic mechanisms," *Computer Communications*, vol. 111, pp. 120 – 141, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S014036641730796X

[7] M. Ali, R. Dhamotharan, E. Khan, S. U. Khan, A. V. Vasilakos, K. Li, and A. Y. Zomaya, "Sedasc: Secure data sharing in clouds," *IEEE Systems Journal*, vol. 11, no. 2, pp. 395–404, June 2017.

[8] H. Noura, A. Chehab, L. Sleem, M. Noura, R. Couturier, and M. M. Mansour, "One round cipher algorithm for multimedia iot devices," *Multimedia tools and applications*, vol. 77, no. 14, pp. 18 383–18 413, 2018.

[9] H. Marzouqi, M. Al-Qutayri, K. Salah, D. Schinianakis, and T. Stouraitis, "A high-speed fpga implementation of an rsd-based ecc processor," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 1, pp. 151–164, Jan 2016.

[10] K. Xue, J. Hong, Y. Ma, D. S. L. Wei, P. Hong, and N. Yu, "Fog-aided verifiable privacy preserving access control for latency-sensitive data sharing in vehicular cloud computing," *IEEE Network*, vol. 32, no. 3, pp. 7–13, May 2018.

[11] S. Basudan, X. Lin, and K. Sankaranarayanan, "A privacy-preserving vehicular crowdsensing-based road surface condition monitoring system using fog computing," *IEEE Internet of Things Journal*, vol. 4, no. 3, pp. 772–782, June 2017.

[12] B. Wang, Z. Chang, Z. Zhou, and T. Ristaniemi, "Reliable and privacy-preserving task recomposition for crowdsensing in vehicular fog computing," in *2018 IEEE 87th Vehicular Technology Conference (VTC Spring)*, June 2018, pp. 1–6.

[13] M. Du, K. Wang, X. Liu, S. Guo, and Y. Zhang, "A differential privacy-based query model for sustainable fog data centers," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2018.

[14] A. Viejo and D. Sánchez, "Secure and privacy-preserving orchestration and delivery of fog-enabled iot services," *Ad Hoc Networks*, vol. 82, pp. 113 – 125, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1570870518305493

[15] D. Koo and J. Hur, "Privacy-preserving deduplication of encrypted data with dynamic ownership management in fog computing," *Future Generation Computer Systems*, vol. 78, pp. 739 – 752, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X17301309

[16] R. Lu, K. Heung, A. H. Lashkari, and A. A. Ghorbani, "A lightweight privacy-preserving data aggregation scheme for fog computing-enhanced iot," *IEEE Access*, vol. 5, pp. 3302–3312, 2017.

[17] F. Y. Okay and S. Ozdemir, "A secure data aggregation

protocol for fog computing based smart grids," in *2018 IEEE 12th International Conference on Compatibility, Power Electronics and Power Engineering (CPE-POWERENG 2018)*, April 2018, pp. 1–6.

[18] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, and K. Salah, "A user authentication scheme of iot devices using blockchain-enabled fog nodes," in *2018 IEEE/ACS 15th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, 2018, pp. 1–8.

[19] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[20] M. O. Rabin, "Efficient dispersal of information for security, load balancing, and fault tolerance," *Journal of the ACM (JACM)*, vol. 36, no. 2, pp. 335–348, 1989.

[21] R. M. Capocelli, A. De Santis, L. Gargano, and U. Vaccaro, "On the size of shares for secret sharing schemes," in *Annual International Cryptology Conference*. Springer, 1991, pp. 101–113.

[22] H. Krawczyk, "Secret sharing made short." in *Crypto*, vol. 93. Springer, 1993, pp. 136–146.

[23] I.-P. Katrinebjerg and R. W. Lauritsen, "Backups with computational secret sharing."

[24] R. L. Rivest, "All-or-nothing encryption and the package transform," in *International Workshop on Fast Software Encryption*. Springer, 1997, pp. 210–218.

[25] J. K. Resch, "Development cleversafe, inc. 222 s. riverside plaza, suite 1700 chicago, il 60606," 2011.

[26] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *Journal of the society for industrial and applied mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

[27] F. J. MacWilliams and N. J. A. Sloane, *The theory of error-correcting codes*. Elsevier, 1977.

[28] M. Li, C. Qin, P. P. Lee, and J. Li, "Convergent dispersal: Toward storage-efficient security in a cloud-of-clouds." in *HotCloud*, 2014.

[29] S. Wang, D. Agrawal, and A. El Abbadi, "A comprehensive framework for secure query processing on relational data in the cloud," *Secure Data Management*, pp. 52–69, 2011.

[30] Z.-t. Yu, Q. Qian, R. Zhang, and C.-L. Hung, "Sida: An information dispersal based encryption algorithm," in *Frontier Computing*. Springer, 2016, pp. 239–249.

[31] L. Chen, T. M. Laing, and K. M. Martin, "Efficient, xor-based, ideal (t, n)-threshold schemes," in *International Conference on Cryptology and Network Security*. Springer, 2016, pp. 467–483.

[32] W. J. Buchanan, D. Lanc, E. Ukwandu, L. Fan, G. Russell, and O. Lo, "The future internet: A world of secret shares," *Future Internet*, vol. 7, no. 4, pp. 445–464, 2015.

[33] M. Wazid, A. K. Das, N. Kumar, and A. V. Vasilakos, "Design of secure key management and user authentication scheme for fog computing services," *Future Generation Computer Systems*, vol. 91, pp. 475 – 492, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18303959

[34] M. A. Khan and K. Salah, "Iot security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018.

[35] W. Stallings, *Cryptography and network security: principles and practice*. Pearson Upper Saddle River, 2017.

[36] K. Chida, K. Hamada, D. Ikarashi, R. Kikuchi, and B. Pinkas, "High-throughput secure aes computation," in *Proceedings of the 6th Workshop on Encrypted Computing & Applied Homomorphic Cryptography*. ACM, 2018, pp. 13–24.