An Efficient Data Integrity Verification Scheme For Distributed Fog Computing Architecture

Youssef Sellami, Youcef Imine, Antoine Gallais
LAMIH-UMR CNRS 8201, Universités Polytechnique Hauts-de-France Valenciennes, France
Email: {Youssef.Sellami, Youcef.Imine, Antoine.Gallais}@uphf.fr

***Abstract*—Fog computing is a promising computing paradigm that provides computing services close to end-users at edge of the network. Therefore, it supports large-scale, geographically distributed and latency-sensitive applications. However, due to its untrusty nature, numerous security challenges must be overcome. The integrity verification of the data in fog computing context is one of the main challenges to tackle. Indeed, most of existing solutions consider a centralized data storage context, and usually rely on a trusted third party to check data integrity. However, these approaches are not suitable for fog computing architecture, since the data is stored in a dynamic and a completely distributed manner.**

Motivated by these challenges, we propose in this paper, a new efficient public verification scheme that protects the integrity of the data in fog computing architecture. Our scheme secures data integrity and authenticity based on the short integer solution SIS problem and the identity-based signatures. In addition, it allows to efficiently verify the integrity of data, even when it is separately shared on multiple servers. This verification can be performed by any end-user on the architecture, and without relying on any trusted third party. Finally, we show through extensive simulation that our solution is highly effective and outperforms existing solutions.

***Index Terms*—Data integrity, Fog computing, Security, Lattice-based cryptography, Distributed networks**

I. Introduction

Fog computing is a new paradigm that aims to extend cloud services to the edge of the network. While the cloud is known as a centralized system, fog is a distributed and decentralized infrastructure [4]. In fact, this infrastructure has the ability to support and manage a large amount of shared information in distributed manner. In addition, with the significant processing power of fog servers, it is also known as an intelligent gateway that offloads to the cloud. Therefore, it enables more efficient data storage, analysis, processing and improves quality-of-services (QoS) for streaming and real time applications. As fog computing is still in its infant stage, it faces many security challenges due to its distinct characteristics [12], such as large scale geo-distributed nodes and low latency.

One of the most serious concerns in the fog is that users are unsure of the the integrity of their data. Indeed, according to the current fog computing paradigm, the fog servers layer is mainly constituted of equipment that are close to end-users, but that do not necessarily implement robust security mechanisms [2]. Therefore, it is very difficult to have trust on these servers. Moreover, given the huge flows of communications and exchanged fog architecture, it is very difficult to permanently verify the integrity of one's whole data. Therefore, it is more suitable to have an integrity verification mechanism that allows to verify the integrity of the whole data by randomly checking few of its blocks.

Academic contributions allowing data integrity verification are generally divided into two categories [11]. In the first one, only the data owners can perform the integrity verification (private verification), while in the second one, other entities may challenge storage servers and verify data integrity (public verification).

Usually, Merkle Hash Tree (MHT) is the most used method to protect the integrity due to its simple design and efficiency [18]. This design is based on the tree structure, which is built from the leaves to the root. Thus, the user is able to verify the authenticity and integrity of the data block based on the root signature. This solution becomes however time and memory consuming, if we manage large amounts of data fragmented into multiple data blocks. Furthermore, traditional methods such as bloom filter [6] have mainly focused on data integrity verification of static data. However, a limited attention has been given to the importance of considering the possibility of dynamic data updates in integrity verification.

In addition, in a public integrity verification system, data owners are not always available to respond to data consumer verification queries. Therefore, for practical reasons, owners can delegate this task to a trusted third-party auditor (TPA). However, the design of TPA model usually considers a centralized service, which does not meet the requirements of high frequency distributed data verification in fog computing. In fact, this would result in a significant loss of time to properly satisfy verification requests within the fog computing architecture.

To overcome these problems, we propose a new public integrity verification scheme for data shared in a distributed fog computing architecture. Our scheme is secure and reduces data integrity checking latency. In addition, it facilitates public data access, and does not require the

978-1-6654-6975-3/22/$31.00 ©2022 IEEE

intervention of the data owner or any Third party auditor. Moreover, it considers both static and dynamic data sharing cases. Our paper presents the following contributions:

- We propose a new data structure based on the SIS problem and identity-based signatures to enable efficient data fog integrity checking.
- We construct a new publicly verifiable scheme to check the integrity of the fog data that supports dynamic operation.
- We propose a scheme that reduce storage and communication overhead between network components compared to Public Key Infrastructure based solutions.

The remainder of this paper is organized as follows: related works are summarized in section II. In section III, we detail the Background. In section IV we present the proposed approach. In section V, we give a security analysis. In section VI, we evaluate the performance of our solution. Finally, in section VII, we conclude the paper.

II. Related Work

Most of the data integrity verification techniques proposed in the literature, focuses on the architecture that supports cloud computing, while few solutions that have been proposed for distributed architectures. To ensure integrity, authors [17] have based their solutions on the use of Counting Bloom Filter CBF to support dynamic operations on the data. However, this solution is limited because of the BF which is known for its false positive failure, where there is a chance of getting an erroneous response when searching a message. Authors [9] proposed a new publicly verifiable scheme namely Position-aware Merkle Tree (PMT) to resist integrity attacks by constructing some positional parameters in each node, they also proposed a rigorous security proof on the PMT. However, this design requires the CSP to transfer the Auxiliary Authentication Information AAI of each contested block to the auditor, which leads to high resource consumption. Authors [14] proposed an approach based on MHT, BLS signature, and bilinear mapping to verify the integrity of the stored data in the cloud. The user creates the MHT based on the block signatures, calculates and signs the root, and then delivers the signed root, file and block signatures to the server. Throughout the audit process, the user selects a random group of blocks and provides the block numbers to the server. The server responds with the hash values of the blocks to be validated, the signed root and the siblings hash values. It enables public auditability, remote data integrity, and dynamic operations support.
Authentication on peer-to-peer storage networks was addressed in [13], where a distributed hash table model based on the MHT was applied. However, they did not implement their solution and did not experimentally test its effectiveness. The blockchain has been used as a way to save the history of transactions made on cloud-based content. Since the blockchain is public, all nodes in the blockchain network can verify data using pseudonymous identities on the blockchain [8]. Indeed, the blockchain is known by its distributed aspect and it uses MHT to protect the integrity of a single block of data at a time. Therefore, we can notice that the MHT is not distributed but it is used in a distributed context like in blockchain. Whereas in the context of fog computing, we have the advantage of saving data blocks in a distributed way. Thus, this will pose a problem in terms of computation and memory management, including with MHT.
Due to the nature of fog computing, the data is constantly queried in the fog servers and is constantly modified, hence it is vulnerable to forgery attacks, for this purpose a lattice-based incremental signature scheme based on the SIS problem was proposed [15]. Thus, it provides an incremental authentication scheme of the updated fog data while maintaining the public key size and signature length within an acceptable range. They also minimized the computation time by using parallel computing. However, the use of these schemes is not efficient because they do not prevent the spread of the old signature when creating the new one. An interesting publication proposed in 2019 [16], which addresses the problem of data integrity verification in the cloud, where they proposed a framework for data sharing in cloud computing and proposed a public verification scheme preserves user identity traceability and privacy. However, their scheme is based on a TPA which is a source of limitation in terms of communication time loss when there will be multiple integrity verification requests in the network, as well as its vulnerability in the architecture as a central server for integrity checking.

III. Background

In this section, we present some security aspects that will be exploited in our scheme.

A. Lattices

To start with, we consider the shortest independent vector problem SIVP that is described as follows:

With a lattice L of dimension n, SIVP consists of finding the n linearly independent vectors $\{v_1, ..., v_n\} \in$ L and let $max_i \| v_i \| \leq \mu$ - λ_n(L), where $\mu \geq 1$ is a function of dimension n. However, this issue is known to be NP-hard for any approximation factor $\mu \leq$ O(1) [3].

B. Short integer solutions problem (SIS)

The SIS problem has been defined based on Ajtai's work [1] as follows :

Let l and q be integers, where l is the primary security parameter and usually $q = poly(l)$, and let $\beta > 0$. Given a uniformly random matrix $A \in Z_q^{l \times m}$ and $m > llogq$, it is hard to find a nonzero integer vector $z \in Z^m$, such that $Az = 0 \ mod \ q$ and $||z|| \leq \beta$ (where $||.||$ denotes Euclidean norm).

Based on the hardness of SIS, the authors in [10] has then proven that given the same parameters as SIS, $f_A(x) = Ax \ mod \ q$ is a collision resistant function.

C. Bilinear Maps

Let G_0 and G_1 be two multiplicative cyclic groups of prime order p. Let P be a generator of G_0 and e be a bilinear map, such that e : $G_0 \times G_0 \rightarrow G_1$
The bilinear map e has the following properties:

Bilinearity: for all U, V $\in G_0$ and a, b $\in \mathbb{Z}_p$, we have: e(a.U,b.V) = e(U, V)^{ab}

Non-degeneracy: e(P, P) $\neq 1$.

We say that G_0 is a bilinear group if the group operation in G_0 and the bilinear map e are both efficiently computable. Notice that the map e is symmetric if e(a.P,b.P) = e(P, P)^{ab} = e(b.P,a.P)

IV. Proposed approach

In this section, we present our proposed scheme that verifies the integrity of published data blocks in a distributed Fog computing architecture.

Our protocol benefits from the hardness of SIS problem to ensure the integrity of our data. Therefore, data owners will first structure their data blocks using a collision resistant function that we build according to SIS hard instances.The data blocks may then be separately shared on multiple fog servers. Furthermore, our scheme allows to authenticate any data shared in the architecture through a secure Identity Based Signature (IBS). Thus, the owner need to store a signature for the whole data in each fog server. On the other hand, any data consumer is able to verify the data block by comparing the result of running our collision resistant function on the received data and the one checked using IBS verification.

A. Architecture design overview

Our distributed architecture is divided over several zones. In each zone, we have the following components:

a) Identification servers: responsible for the management of security in a given zone. Therefore, all security parameters and cryptographic materials will be generated by these entities. In our protocol, we consider these servers to be fully trusted.

b) Fog servers: which provides the data storage service. Each server is then responsible for storing users data, as well as responding to data access requests. In our scheme, we consider fog servers to be curious but honest.

c) End users: we can distinguish two types of end users: data owners and consumers. Data owners can share their data blocks anywhere on the fog servers and in any zone, where they are connected and identified. On the other hand, consumers can navigate between zones and access data from the zone they are connected to. In our protocol, we consider that users may act maliciously.

B. Model

The proposed model consists of the following steps:

1) Setup phase:

- Let G_0 and G_1 be two multiplicative cyclic groups of prime order p and P be a generator of G_0.
- Let e be a bilinear map defined as follows e : $G_0 \times G_0 \rightarrow G_1$
- Let H_1, H_2 and H_3 be three hash functions defined as follows: $H_1 : \{0,1\}^* -> G_0$ that maps identities to elements in G_0, $H_2 : \{0,1\}^* -> Z_p^*$ and $H_3 : \{0,1\}^* -> \{0,1\}^\lambda$
- Let n, m, q be integers chosen according to SIS security parameters (see III-B)
- Let I be a uniform random number generator [7] defined as follow: I_{n+1} = (aI_n+C_n) mod q, C_{n+1} = (aI_n+C_n) div q. Where the unique integers a, I_n, C_n $\in \mathbf{Z_q^*}$, the starting values I_0 and C_0 are the seed, a is the multiplier, q is the base and C_n is the carry.

Each identification server IS_s generates some security parameters related to its own zone, as follows:

- Choose a master secret key $MSK_s = t$, where t randomly in Z_p^* and compute the public key of the zone as $PK_s = t.P$.
- Publish the zone's public key as PK_s.

2) Identification phase: in order to share and protect the integrity of data in zone$_i$, the owners must perform an identification with the server IS_s. The goal is to verify the owner's identity in order to provide him the keys used to sign shared data. To do so, we run the following steps:

- The owner O_w sends his identity to the server IS_s.
- Upon the verification of the owner O_w's identity, IS_s computes a secret key for O_w as $S_{ID_w} = t.H_1(ID_w)$.
- IS_s sends S_{ID_w} to O_w through secure channel.

Since our solution aims mainly to ensure integrity, we assume that identification servers have a secure method to verify data owners identities. However, how this verification is done is out of the scope of our paper.

3) Data sharing: in order to ensure the integrity of their data, data owners proceeds as follow:

a) Public parameters preparation:

- Choose randomly the parameters I_0,a,C_0 of the uniform random generator I (defined among the public parameters in the setup phase).
- Using I, generate a matrix $A \in Z_q^{l*m}$ (with l, m, q satisfying the Ajita SIS conditions (mentioned in III-B)), where $A = \{\{I_1, I_2..., I_m\}, ..., \{I_{(l-1)*m+1}, ..., I_{l*m}\}\}$.

b) Data matrix construction: Given the security parameters (l, m, q) and the matrix A defined in the previous section, the data D constituted of N blocks D=$\{D_1, D_2, ..., D_N\}$, and the hash function $H_3 : \{0,1\}^* -> \{0,1\}^\lambda$.

- Split the N blocks into Δ=m/λ groups one after the other. Therefore, the data will be ranged as $D = \{DG_1, ..., DG_{N/\Delta}\}$, where we have for instance $DG_1 = \{D_1, ..., D_\Delta\}$. For sake of simplicity for the readers, we consider in that N/δ is an integer.

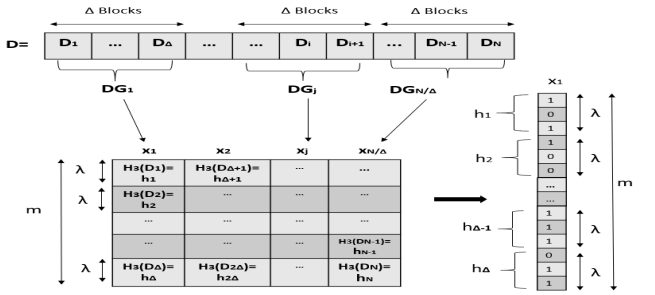

Fig. 1: Matrix X construction.

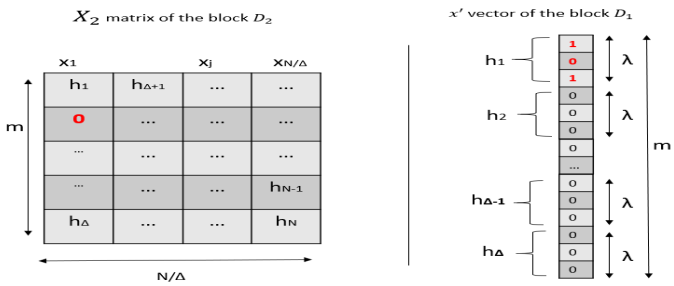

Fig. 2: Construction of the matrix X_2 and the vector x'.

- For each block D_i in each group $DG_j, j \in \{0, N/\Delta\}$, inject the bit sequences of $H_3(D_i)$=h_i into the vector x_j as shown in figure 1.
  In case the number of data blocks is small and we cannot fill the whole vector x_j, we just complete the remaining cells with zeros.
- Construct a matrix $\mathbf{X}$, such as X=[x_1, x_2, ..., $x_{N/\Delta}$].
- Compute V=A*X mod q, where V=[v_1, v_2, ...,$v_{N/\Delta}$] and q is a prime number.

The resulting matrix V ensures the integrity of the blocks injected into X according to SIS problem. Indeed, V is the result of the multiplication of the matrix A, that we generate using the uniform random generator I; and the data matrix X constructed of small entries. Thus, the matrix V is a unique result that is resistant to collisions, since it meets SIS security conditions.

As we can notice, the size of the matrix V may induce a considerable overhead in terms of storage and communication. In addition, it may lead to perform more computation to verify the integrity of data blocks.
Therefore, to decrease this overhead, we reduce the matrix V into a single vector v_u such that $v_u = V * u$, where u is a random vector having N/Δ entries in Z_q^*.

c) Signature generation: in order to securely share its data, the data owner O_w uses his identity ID_w to generate an identity based signature (based on Hess signature [5]) as follows:

- Choose a random $P_1 \in G_0$, and a random $k \in Z_p^*$
- Compute $r = e(P, P_1)^k$
- Compute $w_1 = H_2(v_u, param_A, u_j$,r). Note that, u_j is the cell corresponding to the group DG_j (where the block D_i has been ranged) in u, and $param_A$=(I_0, a, C_0).
- Compute $w_2 = w_1.S_{ID_w}$ + k.P_1
- The signature of the data is then σ = (w_1,w_2)
- For each block D_i in each group DG_j, keep a vector v^{Di}, such that v^{Di}=(A*X_i mod q)*u, where in X_i we copy the entries of X and replace by zeros the entries corresponding to the ith block. (see Figure 2)

The construction of our protocol allows the data owner O_w to securely share his data blocks in a completely distributed manner in different fog servers.

To do so, in each server where some data blocks are shared, O_w stores $param_A$, ID_w and the data signature σ. In addition, O_w shares for each data block D_i, the following information: (i, D_i, v^{Di}, u_j).

Note that in the original construction of SIS [1], the matrix A is stored with data. As a result, the storage overhead of **A** becomes high, since **A** has a large size due to the security parameters (l,m,q). However, in our solution the storage overhead is very low. Indeed, rather than storing all the values of A, in our protocol we only share $param_A$ that contains the uniform random generator I parameters. Consequently, we reduce the overhead while being able to reconstruct A when needed.

4) Integrity verification: in our protocol, end users are able to verify the integrity of any data block that is shared in the architecture. Thus, given the block owner identity ID_w, the public key of the identification server PK_s, the data signature σ and the requested block D_i along with its corresponding security parameters (v^{Di}, u_j, $param_A$), the verification is done as follows:

- Construct a vector $x' \in \{0,1\}^m$ that contains $H_3(D_i)$ in the positions corresponding to the block D_i and zeros everywhere else. (see figure 2). We recall that H_3 is defined as $H_3 : \{0,1\}^* -> \{0,1\}^\lambda$
- Reconstruct the matrix A using $param_A$.
- Compute $(A * x' \text{ mod } q) * u_j = v'$.
- Compute v", such that v" = $v' + v^{Di}$.
- Compute $r' = e(w_2, P).e(H_1(ID_w), -PK_s)^{w_1}$
- If $w_1 = H_2(v$", $param_A$, u_j,r') then the block is valid.

C. Dynamic operation

The owner has the right to delete or to modify the blocks D_i and its associated metadata.

1) Data deletion: to delete the block D_i, the owner must apply the following steps:

- Create a vector x' that contains zeros everywhere except in the cells that support $H_3(D_i)$.
- Compute the vector $v_i^{'}$=$(A * x' \ mod \ q) * u_j$, where u_j is the cell corresponding to D_i in the vector u.
- For each remaining block D_r, update the vector v^{D_r} as $v^{D_r} = v^{D_r}$-$v_i^{'}$.
- Compute a new $v_u^{'}$ as $v_u^{'}$=v_u-$v_i^{'}$.

2) Data modification: to update the block D_i by a D_i', the owner must apply the following steps:

- Run the data deletion phase.
- Construct a new vector x' using the new block D_i'.

- Compute the vector $v_i^{'}=(A * x' \ mod \ q) * u_j$.
- For each remaining block D_r, update the vector v^{D_r} as $v^{D_r} = v^{D_r}+v_i^{'}$.
- Compute a new $v_u^{'}$ as $v_u^{'}=v_u+v_i^{'}$.

3) Data insertion: to insert a new data block D_i', the owner must apply the following steps:

- Check if there are empty cells in the matrix X, where $i < \lceil N/\Delta \rceil$. If not, add a new cell $u_{(N+1)/\Delta}$ in u.
- Construct a new vector x' using the new block D_i.
- Compute the vector $v_i^{'}=(A * x' \ mod \ q) * u_j$.
- For each remaining block D_r, update the vector v^{D_r} as $v^{D_r} = v^{D_r}+v_i^{'}$.
- Add a new vector $v^{D_i'}=v_u$
- Compute a new $v_u^{'}$ as $v_u^{'}=v_u+v_i^{'}$.

Finally, for all the operations above, the data owner generates a new signature σ using the new vector $v_u^{'}$.

V. Security analysis

A. Impersonation attack

Our protocol uses an identity-based signature scheme to guarantee the identity of data owners who share or modify the data blocks in fog servers. If an attacker impersonates an owner's identity, it will need to recover the private key of that user. In that case, the attacker needs to be able to produce a new signature using the identity of the owner or to forge an existing one.

In the first case, that would mean that the attacker is able to recover the master secret key of the identification server (IS), and then use it to generate a private key with the owner's identity. Therefore, as long as the master secret key of the IS is kept secret, only one option remains to explore. This option consist of recovering the MSK from the public key of the IS. However, this will require to solve the discrete logarithm problem in cyclic groups, which is supposed to be hard. In the second case, the attacker need to forge an existing signature. However, in our scheme we a signature that has been proven to be secure in [5].

B. Attack on data integrity

The attacker may seek to alter data blocks or their associated parameters by deletion or modification. However, as discussed above, forging the signature associated with the data is a hard task. Therefore, an attacker may try to modify the data while keeping the same signature. Since the owner will generate a signature using the v_u (see section IV-B3c), the attacker best option is to alter the data in a way that he finds a matrix X' such that $f_A(X') = A * X'$ will still be equal to v_u. However, in our scheme, u_v is computed using a one way function f_A that is, according to SIS problem, resistant to collisions [10]. Therefore, finding such matrix X' means that SIS problem can be broken since a collision has been found.

VI. Performance evaluation

In this section, we evaluate and compare the performance of our protocol with MHT and BF. To do so, we have run a simulation in which we consider two type of processes, one for data owners and the other for data consumers. Both processes share and access data into a virtual fog server. Table I shows simulation settings.

Table I: Simulation parameters

Simulation parameters	Value
Number of rows l of **A**	64
Number of column m of **A** (m $\geq$ 2l lg q)	1536
Value of q $\approx l^2$	4099

A. Number of calculated hashes

Figure 3 shows a comparison between our scheme, MHT and BF in terms of the number of hashes calculated for each method. As we can notice, the number of hashes computed in the three schemes grows linearly according to the number of data blocks. Whereas, our solution generates less hashes since we generate only one hash per block. As for the BF, the number of hashes generated per block varies depending on the choice of security parameters. Increasing this number will minimize the probability of false positives, but it will have a direct impact on the performance of this solution. Note that, in our simulation we used three hashes per block. Finally, in MHT, we have to compute hashes for each block. These hashes constitute the leaves of a tree. After that, intermediate hashes are computed up to a root node, where each intermediate node up in the tree is the hash of its respective children. However, by increasing the number of blocks, the size of the tree grows, which complicates its management.

B. Integrity verification latency

Figure 4 shows a comparison between our scheme, MHT and BF in terms of the required time to apply the integrity check of a block. We notice that for some data blocks, other solutions have better results. In addition, We notice that our solution has an almost stable variation even if the number of data blocks increases, since we need to recover only the data block and its associated parameters. As for the MHT, some of the data blocks and their hash combination are required to reconstruct the tree, which results in a higher waste of time. Regarding the BF, we notice that we approach its performance when we reach a high number of blocks. Moreover, our solution is deterministic while it is approximate for the BF.

C. Block modification latency

Figure 5 shows a comparison between our scheme and MHT in terms of the required time to apply a change to a single block of data. We notice that our solution performs better than MHT, especially when the number of data blocks increases which implies the growth of the tree and complicates the modification. But in our solution, we only need one block of data and its corresponding metadata that we modify by a simple addition and subtraction computation. As for BF, One of its drawbacks is its inability to perform dynamic operations, for this reason we cannot apply the comparison.

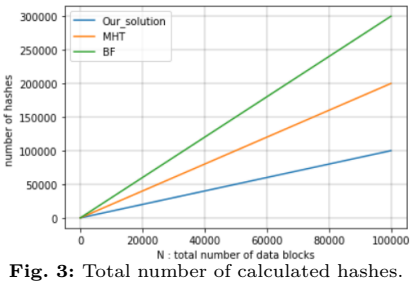

Fig. 3: Total number of calculated hashes.

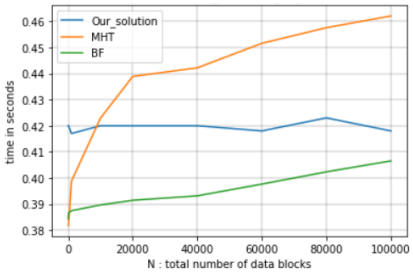

Fig. 4: Latency for block integrity checking.

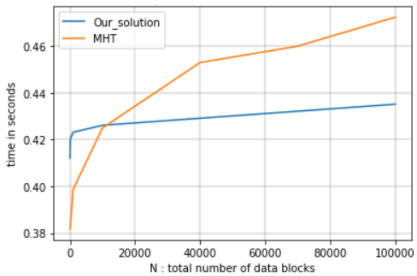

Fig. 5: Latency for block modification.

VII. Conclusion

In this paper, we proposed a new scheme for data integrity verification in a distributed fog computing architecture. We secure the integrity verification process using SIS problem. Moreover, our scheme allows data owners to distribute data blocks in multiple servers in the architecture. The authenticity of these blocks can still be ensured and verified using an identity based signature mechanism. Furthermore, our solution allows a fast data integrity verification and modification. This verification can be done by any end-user, without the relying on a trusted third party, and with a reduced storage, computation and communication overhead. Finally, we demonstrated through simulation results that our scheme achieve better results compared to existing solutions during data sharing, verification and modification.

In future work, we intend to evaluate our scheme in a real use case application, and investigate giving data consumers the possibility to modify the data while ensuring identity accountability.

References

[1] Ajtai and al. Generating hard instances of lattice problems. In *Proceedings of the twenty-eighth annual ACM symposium on Theory of computing*, pages 99–108, 1996.

[2] Aljumah and al. Fog computing and security issues: A review. In *2018 7th international conference on computers communications and control (ICCCC)*, pages 237–239. IEEE, 2018.

[3] Blömer and al. On the complexity of computing short linearly independent vectors and short bases in a lattice. In *Proceedings of the thirty-first annual ACM symposium on Theory of computing*, pages 711–720, 1999.

[4] De Donno and al. Foundations and evolution of modern computing paradigms: Cloud, iot, edge, and fog. *Ieee Access*, 7:150936–150948, 2019.

[5] Hess and al. Efficient identity based signature schemes based on pairings. In *International workshop on selected areas in cryptography*, pages 310–324. Springer, 2002.

[6] Jeong and al. Secure cloud storage service using bloom filters for the internet of things. *IEEE Access*, 7:60897–60907, 2019.

[7] L'Ecuyer and al. Uniform random number generation. *Annals of Operations Research*, 53(1):77–120, 1994.

[8] Liang and al. Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, pages 468–477. IEEE, 2017.

[9] Mao and al. A position-aware merkle tree for dynamic cloud data integrity verification. *Soft Computing*, 21(8):2151–2164, 2017.

[10] Micciancio and al. Hardness of sis and lwe with small parameters. In *Annual Cryptology Conference*, pages 21–39. Springer, 2013.

[11] Shacham and al. Compact proofs of retrievability. In *International conference on the theory and application of cryptology and information security*, pages 90–107. Springer, 2008.

[12] Stojmenovic and al. An overview of fog computing and its security issues. *Concurrency and Computation: Practice and Experience*, 28(10):2991–3005, 2016.

[13] Tamassia and al. Efficient content authentication over distributed hash tables. *Technical Report*, 2005.

[14] Wang and al. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE transactions on parallel and distributed systems*, 22(5):847–859, 2010.

[15] Wang and al. Efficient incremental authentication for the updated data in fog computing. *Future Generation Computer Systems*, 114:130–137, 2021.

[16] Yan and al. Integrity audit of shared cloud data with identity tracking. *Security and Communication Networks*, 2019, 2019.

[17] Zhang and al. A joint bloom filter and cross-encoding for data verification and recovery in cloud. In *2017 IEEE symposium on computers and communications (ISCC)*, pages 614–619. IEEE, 2017.

[18] Zhou and al. Data integrity verification of the outsourced big data in the cloud environment: A survey. *Journal of Network and Computer Applications*, 122:1–15, 2018.