# One Digit Checksum for Data Integrity Verification of Cloud-executed Homomorphic Encryption Operations

Mostefa Kara [1], Abdelkader Laouid [1], Omer Al dabbas [2],
Mohammad Hammoudeh [3], and Ahcène Bounceur [4]
[1] LIAP Laboratory, University of El Oued,
PO Box 789, El Oued 39000, El Oued, Algeria.
[2] Faculty of Engineering Technology,
Al-Balqa' Applied University,
Amman 15008, Jordan Amman, Jordan
[3] Information & Computer Science Department,
King Fahd University of Petroleum and Minerals,
Academic Belt Road, 31261, Dhahran, Saudi Arabia
[4] Lab-STICC UMR CNRS, University of Western Brittany UBO,
Brest 6285, France

February 20, 2023

**Abstract**

Homomorphic Encryption (HE) is used in many fields including information storage, data protection, privacy preservation, blockchain, and authentication. HE allows an untrusted third party to perform algebraic operations on encrypted data. Protecting the results of HE against accidental or malicious tampering attacks is still an open research challenge. In this paper, we introduce a lightweight technique that allows a data owner to verify the integrity of HE results performed in the cloud. The proposed method is quick, simple, and applicable, as it depends on adding a single digit to the encrypted message before storing it in the cloud. This digit represents verification proof and it is later used to ensure a verifiable HE. Our technique can be integrated with any HE scheme that uses encryption with non-isolated plaintext.

**keywords :** Data Integrity, Verifiable FHE, Homomorphic Encryption, Cloud Computing.

# 1 Introduction

Cloud computing is an infrastructure where storage and processing are managed by remote servers that are accessed through the Internet. Clients use the Internet to access cloud services such as data storage, processing, software, intelligence, and networking in a flexible on-demand approach. Cloud data access and management are characterized by their flexibility to meet the client needs [1]. Cloud computing is divided into several levels of services [2]. Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS). Through IaaS, the cloud gives client companies access to the IT infrastructure they need, calculations, storage, networks, security, etc. As part of a PaaS solution, the cloud provides the client company with a development and deployment environment in which it will be able to design its own web and mobile applications. The SaaS model amounts to providing access to software hosted in the cloud. This software is accessible from any point and not depends on the company's computer system.

Cloud services offer companies several considerable advantages, they save storage space and computing time. The benefits of cloud computing are faced with several security and privacy challenges that hinder their adoption. Integrity checking and encryption are crucial to protecting users' privacy and data. For example, the client is generally unaware of other clients that use the same server to perform various tasks. Therefore, the customer cannot control the processing of his data.

Security techniques [3], confidentiality [4, 5], and data encryption [6, 7] offer an effective solution to overcome the conflicting requirements of privacy and cloud storage. However, classic encryption does not provide complete protection of data confidentiality and integrity. Despite being in an encrypted form both during storage and transmission, the data must be decrypted in order to be processed by a third party. Inexpensive cloud computing and cloud storage have fundamentally changed how companies manage their data. Classic encryption techniques, such as AES, are extremely fast and store data in encrypted form. Nevertheless, to perform a simple operation on the encrypted data, either the cloud needs access to the secret key, which leads to confidentiality concerns, or the client needs to download, decrypt, and work on data locally, which can be costly and make a logistic challenge. Homomorphic Encryption (HE) offers a solution to processing data in its encrypted format to reduce integrity and confidentiality risks [8]. With HE, the cloud can directly perform on the encrypted data and return only an encrypted result to the client. However, HE may suffer from some limitations such as feasibility, the impossibility of running ad-hoc/discovery-based queries, the possibility to leak private information, keeping track of noise, and data integrity issue, which we will address in this paper.

## 2 Related work

Fully Homomorphic Encryption (FHE) is a common way to support data computation in the ciphertext form by third parties who are not trusted. Untrusted entities may return incorrect calculation results accidentally or intentionally. To guarantee data integrity, many works in the literature presented verifiable encryption-decryption schemes. An analysis of existing FHE integrity techniques has been presented in [9], and the authors proposed a new vision for VFHE. They analyzed a range of schemes and evaluated their performance in different settings.

Huang et al. [10] proposed a Verifiable FHE scheme (VFHE). The plaintext is transformed into the triangular matrix and the matrix blinding technology is exploited to protect privacy and ensure security. Data outsourcing is used to implement FHE based on the matrix operation principle. The verification proof is generated according to the specific properties of the triangular matrix, so clients can check the correctness of the results. The limitation of this technique is that the client must keep proof of verification for each operation to be performed. Also, it suffers from a high size and complexity (key size, ciphertext size, and computing complexity: $2 \times d^2$, $d^2$, and $O(d^2)$ respectively).

The authors of [11] proposed a verifiable noise-free FHE technique that uses the ring of Lipschitz's quaternions and allows computations over encrypted data under a symmetric key. This probabilistic cryptosystem allows verification of a calculation if it was performed in its correct form or not. To construct their VFHE scheme, the authors used properties of non-commutativity of Lipschitz integers. The drawback of this scheme is that it requires the user before encrypting a plaintext to transform it into a quaternion.

In [12], the authors studied the operation of verifiable delegation of computation on encrypted data. They tried to improve previous definitions in order to tolerate attacks that learn whether or not users accept the result of a delegated computation. Working in the amortized model of [13], they constructed a technique for arbitrary calculation. They developed a homomorphic hashing scheme that permits authenticating computation results. The problem generation algorithm in this scheme uses the secret key to encrypt a message to be sent to the cloud to be processed and a secret value that is kept private by the user, which takes extra storage space.

Jin et al. [14] provided a universal construction of FHE. Using the existing FHE schemes, they constructed a general VFHE. The objective of their scheme is the verifiability of an evaluate function $f$. The problem in this construction is that $f$ depends on the corresponding FHE scheme. Unlike our technique which uses a single method that works with all FHE schemes.

The authors of [15] proposed a verifiable decryption for the Brakerski-Gentry-Vaikuntanathan (BGV) scheme. They presented an interactive zero-knowledge proof protocol to certify the correctness decryption of BGV ciphertexts. The challenge in BGV is to provide an interactive proof system of three rounds, which takes more time. To enforce data integrity during an outsourced computation, Awadallah et al. [16] proposed a verification scheme based on the

modular residue to validate homomorphic encryption computation over an integer finite field. This scheme incurs 1.5% storage overhead. It depends on the ciphertext size as opposed to our scheme which uses a single digit regardless of the ciphertext size.

# 3 Verifiable encryption

Verifiable Encryption (VE) is an encryption technique where a party can prove some characteristics of an encrypted message $m$. When an encryption technique is secure, ciphertext $Enc(m)$ should reveal no information about $m$. This property may not be suitable in cases where checking some characteristics of the encrypted data is required before handling the encrypted message. The definition of VE can be generalized as follows. VE allows a verifier to check some properties of a message after performing operations in its encrypted form.

A general system model of a VE can be shown as follow.

- The client encrypts a message $m$ to get the ciphertext $c = Enc(m)$, also computes the related verification proof $VP$. Then the client must save $VP$ and sends the computing function $F$ to the third part for calculation (TPC).
- After TPC receives data, calculates the ciphertext $c$ according to $F$, and returns the results $r$ to client.
- The client decrypts $r$ where $m' = Dec(r)$ and then verifies the correctness of $m'$ according to $VP$. If they are valid, the computing results from TPC will be accepted. Otherwise, $r$ will be rejected.

Security threats in basic VE models originate from the TPC, which may reveal and misuse the private data, may dishonestly perform the computing operation due to other goals such as computing resource saving or software and hardware errors, and return incorrect results. Therefore, VE scheme must satisfy the following objectives:

- Correctness: If TPC honestly performs the computing operations; the results must be correct and can be accepted by the client.
- Privacy: TPC can not access any sensitive data from the input or output computing results.
- Verifiability: The client can check the correctness-incorrectness of results returned by TPC at an extremely high probability.

Considering implementation requirements and conditions, it is difficult to build a HE that is fully and verifiable at the same time. All the proposed VFHE schemes are difficult to use in practice. This is why we have developed this verification technique, which can be integrated with almost any FHE proposed in the literature. In our scheme, it is not necessary to save the related verification proof $VP$.

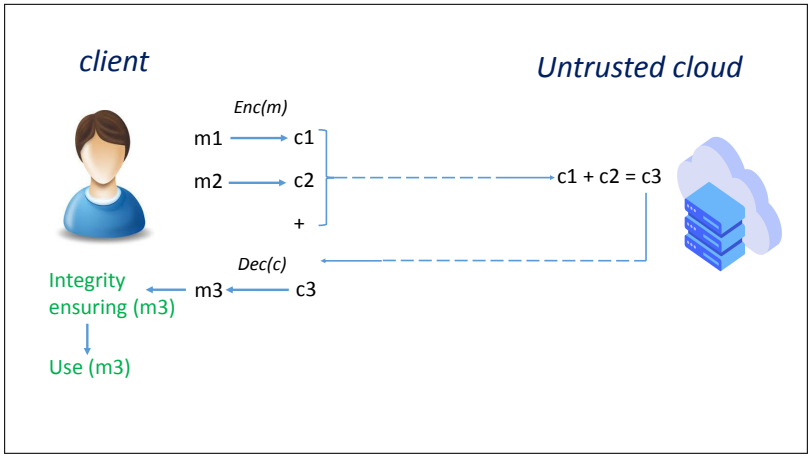


Figure 1: Homomorphic encryption and data integrity

# 4 HE and Data integrity issue

Data integrity is an essential and sensitive requirement in the design, implementation, and use of any data system. It can be defined as the validity, completeness, accuracy, and consistency of data as illustrated in Figure 1. After proper data validation and error checking, the data owner can ensure that sensitive data is not used, exploited, improperly classified, or stored incorrectly.

All unauthorized changes to the data during storage, computation, or retrieval operations, including unexpected hardware failure, malicious intent, or human errors, will inevitably lead to errors in the data use later. Data integrity can be guaranteed in local databases to prevent intentional data changes. But it will be much more difficult when using a third party, e.g., an untrusted cloud, to make operations on the encrypted data.

While validation of these homomorphic calculations is a prerequisite for data integrity, our proposal consists of adding one digit to the ciphertext to prevent any manipulations of homomorphic processing outcomes by a third party, whether these changes are accidental or intentional.

Suppose a data owner requests the cloud to excute a homomorphic addition operation of two ciphertexts. After obtaining $c_3$ ($c_3 = c_1 + c_2$), the data owner will utilize $c_3$ considering that it is a valid output, i.e., $c_3$ was not manipulated. Nevertheless, the untrusted cloud may have changed that calculated result, whether by error or deliberately. The data owner can not discover this change especially if many 'addition' operations are performed.

$User$: encrypts $x$ and $y$, sends them to the cloud.
$Cloud$: $c_x + c_y \rightarrow c_z$, changes it to $c'_z$ (where $c_z \neq c'_z$), sends it to the user.
$User$: decrypts and uses $c'_z$, but $Dec(c'_z) = z' \neq x + y$ !

# 5 Proposed Technique

The core idea is to set the verification proof $v$ equal to the sum of digits of the plaintext recursively until a single digit is obtained (Equation 1).

$$v = P(m) = \sum(\sum(..(\sum_{i=1}^{j} d_i))) \tag{1}$$

Where $P$ function denotes $Proof(m)$ and $m = d_1 d_2 \dots d_j$ with $d_i \in \{0, .., 9\}$. We have $v \in \{0, .., 9\}$; this calculation can be reduced and replaced by a single operation as shown in Equation 2.

$$v = P(m) = m \mod 9 \tag{2}$$

Noting that we can use any modulus other than 9 i.e., $v = m \mod k$ where $\forall k < m$.

After the client calculates the verification proof $v$, it encrypts the message $c = Enc(m)$ and sends the couple $(c, v)$ to the cloud. The client does not need to store $v$ locally. Finally, the cloud must store the ciphertext $c_i$ with the corresponding verification proof $v_i$. Note that the proposed technique does not respond to a specific FHE scheme but it is applicable to all HE schemes.

$$C = (c, v) \quad where\ c = Enc(m)\ and\ v = m \mod 9 \tag{3}$$

In each calculation operation $O$ performed on $c_i$ by the cloud, the cloud must do the same operation $O$ with its $v_i$; finally, it computes $v_i \mod 9$ to obtain a single digit as shown in Equation 4 with $O$ is an operation (or set of operations) of addition or multiplication.

$$(c_3, v_3) = (c_1 O c_2, (v_1 O v_2) \mod 9) \tag{4}$$

The verification consists of calculating the verification poof of result decryption provided by the cloud. So, the client decrypts $c_3$ where $Dec(c_3) = m_3$, calculates $P(m_3)$, and checks if $v_3 = P(m_3)$ or not. $v_3$ is the proof value returned by the cloud.

**Verf**: If $v_3 = P(dec(c_3))$ then accept computed result.

**Example**

Consider the following cryptosystems:

Additive scheme:

$Enc(m) : c = m \times k \mod n$ ; $Dec(c) : m = c \times k^{-1} \mod n$ ; where $k$ is the secret key. Let $k = 1980$ and $n = 2017$;

Multiplicative scheme:

RSA with $n = 3127$ , $e = 3$ , and $d = 2011$;

**Client**

Let plaintexts $m_1 = 25$ and $m_2 = 44$;

So $P(m_1) = 25 \mod 9 = 7$ and $P(m_2) = 44 \mod 9 = 8$;

$c_{+1} = 25 \times 1980 \mod 2017 = 1092$ ;

$c_{+2} = 44 \times 1980 \mod 2017 = 389$ ;
$c_{\times 1} = 25^3 \mod 3127 = 3117$ ;
$c_{\times 2} = 44^3 \mod 3127 = 755$.

**Cloud**

$Op_1$ : addition : $c_+ = c_{+1} + c_{+2} = (1092 + 389) \mod 2017 = 1481$;
$v_+ = (7 + 8) \mod 9 = 6$.
$Op_2$ : multiplication : $c_\times = c_{\times 1} \times c_{\times 2} = (3117 \times 755) \mod 3127 = 1831$ ;
$v_\times = (7 \times 8) \mod 9 = 2$.

**Client (verf)**

$Op_1$ : $m_+ = Dec(c+) = c_+ \times k^{-1} \mod n = 1481 \times 109 \mod 2017 = 69$ ; (69 = 25 + 44);
For verification: $P(m_+) = P(69) = 69 \mod 9 = 6$ it is strictly equal to $v_+$ which is calculated by the cloud ($v_+ = 6$). Therefore, the client accepts $m_+ = 69$.
$Op_2$ : $m_\times = Dec(c_\times) = c_\times^d \mod n = 1831^{2011} \mod 3127 = 1100$ ; (1100 = 25 $\times$ 44);
For verification: $P(m_\times) = P(1100) = 1100 \mod 9 = 2$ it is strictly equal to $v_\times$ which is calculated by the cloud ($v_\times = 2$). Therefore, the client accepts $m_\times = 1100$

# 6 Scheme analysis

To show the correctness of this technique, prove the following verification condition (Lemma 1).

**Lemma 1.** *if $P(Dec(c_1 O c_2)) = v_1 O v_2$, then the returned result is correct.*

*Proof.* We have both $c_1 O c_2$ and $v_1 O v_2$ calculated by the cloud, $Dec()$ function is used by the verifier (client). If $c_1 O c_2$ is done correctly, then $Dec(c_1 O c_2) = m_1 O m_2$ (Homomorphic system). Therefore, $P(Dec(c_1 O c_2)) = P(m_1 O m_2) = (m_1 O m_2) \mod 9 = m_1 \mod 9 O m_2 \mod 9$ because $(m_1 + m_2) \mod 9 = m_1 \mod 9 + m_2 \mod 9$ and $(m_1 \times m_2) \mod 9 = m_1 \mod 9 \times m_2 \mod 9$. So, $P(m_1 O m_2) = P(m_1) O P(m_2) = v_1 O v_2$. □

If we use $v = m \mod 9$, the only information the cloud knows is $\sum digit(m) = v$ and that gives the untrusted cloud no advantage for manipulating the calculated result; more security if we use $k \neq 9$ but with additional storage space; with $k = 9$, $\mid v \mid = 1$; with $k \neq 9$, $\mid v \mid = len(k)$. So, the technique is secure because it does not give any useful information about $m$ to the cloud. In order for the cloud to manipulate the calculation result and at the same time keep the correct corresponding proof, it must have the client's secret key (assuming the decryption scheme is known). The only way to generate $c' \neq c$ with their corresponding $v' \neq v$ is to know the client's secret key $sk$, $c'$ must verify $P(Dec_{sk}(c')) = v'$.

A safe use of our scheme is for example RSA; in above instance where $n$ = 3127 , $e$ = 3 , and $d$ = 2011; with $m$ = 25, $c$ = 3117 , and $v$ = 7; if the cloud want to change $v$ to $v' = 8$, it have to change $c$ to $c' = 3110$ or 2401, etc.

because $Dec(c' = 3110) = 1817$ and $P(1817) = 8 = v'$ or $Dec(c' = 2401) = 566$ and $P(566) = 8 = v'$, that is impossible without knowing the private key $d$. How the cloud can see that $P(Dec_d(c')) = 8$?

Unsafe use of our scheme considering the following trivial FHE scheme. $Enc(m) : c = m + r \times p \mod n$ ; $Dec(c) : m = c \mod p$ ; where $p$ is the secret key and $r$ is a random number. In this example, the target $m$ is isolated and an adversary can attack it; we note here that $Dec(c + 1) = m + 1$, so if the adversary makes $c' = c + 1$ and $v' = v + 1$, the verification condition is still correct; $P(Dec(c + 1)) = P(m + 1) = v + 1 = v'$, the client will accept the incorrect result $c'$. We can say that a scheme of encryption with isolated plaintext $m$ in the ciphertext means $Dec(cO\alpha) = mO\alpha$.

Because our scheme consists of a single elementary calculation regardless of input size in both proof generation and proof verification ($v = c \mod 9$) operations, it realizes a computing complexity equal to $O(1)$.

# 7 Conclusion

In this paper, we presented a new method to ensure data integrity, or rather to ensure homomorphic calculations validity performed on encrypted data by an untrusted third party, whether these errors are intentional or unintentional. The proposed technique is lightweight and applicable, especially in environments that need to low time and storage space such as the Internet of Things. Our method can be integrated with any FHE scheme (any encryption with non-isolated plaintext). It consists of adding one digit to the ciphertext which guarantees the detection of whether the results have been tampered with or not. After presenting some concepts about the cloud and homomorphic encryption, we explained the proposed scheme and then proved its validity, effectiveness, and efficiency in maintaining data integrity. We plan in the future to apply this technique in other fields that require data integrity checks.

# References


[1] J. Zou, D. He, S. Zeadally, N. Kumar, H. Wang, and K. R. Choo, "Integrated blockchain and cloud computing systems: A systematic survey, solutions, and challenges," *ACM Computing Surveys (CSUR)*, vol. 54, no. 8, pp. 1–36, 2021.

[2] M. Hammoudeh, G. Epiphaniou, S. Belguith, D. Unal, B. Adebisi, T. Baker, A. Kayes, and P. Watters, "A service-oriented approach for sensing in the internet of things: Intelligent transportation systems and privacy use cases," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 15 753–15 761, 2020.

[3] M. Kara, A. Laouid, and M. Hammoudeh, "An efficient multi-signature scheme for blockchain," *Cryptology ePrint Archive*, 2023.

[4] M. Kara, A. Laouid, A. Bounceur, F. Lalem, M. AlShaikh, R. Kebache, and Z. Sayah, "A novel delegated proof of work consensus protocol," in *2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP)*. IEEE, 2021, pp. 1–7.

[5] M. Kara, "The secure management of autonomous cloud entities in a distributed system," Ph.D. dissertation, University Of Eloued, 2022.

[6] M. Kara, A. Laouid, A. Bounceur, and M. Hammoudeh, "Secure clock synchronization protocol in wireless sensor networks," 2023.

[7] M. Kara, "A lightweight clock synchronization technique for wireless sensor networks: A randomization-based secure approach," 2023.

[8] S. Moffat, M. Hammoudeh, and R. Hegarty, "A survey on ciphertext-policy attribute-based encryption (cp-abe) approaches to data security on mobile devices and its application to iot," in *Proceedings of the International Conference on Future Networks and Distributed Systems*, 2017.

[9] A. Viand, C. Knabenhans, and A. Hithnawi, "Verifiable fully homomorphic encryption," *arXiv preprint arXiv:2301.07041*, 2023.

[10] R. Huang, Z. Li, and J. Zhao, "A verifiable fully homomorphic encryption scheme," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage: 12th International Conference, SpaCCS 2019, Atlanta, GA, USA, July 14–17, 2019, Proceedings 12*. Springer, 2019, pp. 412–426.

[11] A. El-Yahyaoui and M. D. ECH-CHERIF EL KETTANI, "A verifiable fully homomorphic encryption scheme for cloud computing security," *Technologies*, vol. 7, no. 1, p. 21, 2019.

[12] D. Fiore, R. Gennaro, and V. Pastro, "Efficiently verifiable computation on encrypted data," in *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, 2014, pp. 844–855.

[13] R. Gennaro, C. Gentry, B. Parno, and M. Raykova, "Quadratic span programs and succinct nizks without pcps," in *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*. Springer, 2013, pp. 626–645.

[14] F. Jin, Y. Zhu, and X. Luo, "Verifiable fully homomorphic encryption scheme," in *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*. IEEE, 2012, pp. 743–746.

[15] F. Luo and K. Wang, "Verifiable decryption for fully homomorphic encryption," in *Information Security: 21st International Conference, ISC 2018, Guildford, UK, September 9–12, 2018, Proceedings 21*. Springer, 2018, pp. 347–365.

[16] R. Awadallah, A. Samsudin, and M. Almazrooie, "Verifiable homomorphic encrypted computations for cloud computing," *International Journal of Advanced Computer Science and Applications*, vol. 12, no. 10, 2021.