





Article

A Multi-Key with Partially Homomorphic Encryption Scheme for Low-End Devices Ensuring Data Integrity[†]

Saci Medileh¹ , Abdelkader Laouid^{1,*} , Mohammad Hammoudeh², Mostefa Kara¹ , Tarek Bejaoui³, Amna Eleyan⁴  and Mohammed Al-Khalidi⁴ 

¹ LIAP Laboratory, University of El Oued, P.O. Box 789, El Oued 39000, Algeria

² Information and Computer Science Department, King Fahd University of Petroleum and Minerals (KFUPM), Academic Belt Road, Dhahran 31261, Saudi Arabia

³ Computer Engineering Department, University of Carthage, Amilcar 1054, Tunisia

⁴ Department of Computing and Mathematics, Manchester Metropolitan University, Manchester M1 5GD, UK

* Correspondence: abdelkader-laouid@univ-eloued.dz

[†] This article is a revised and expanded version of a paper entitled A Multi-Key Based Lightweight Additive Homomorphic Encryption Scheme, which was presented at the 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy, El Oued and 20–21 November 2021.

Abstract: In today's hyperconnected world, the Internet of Things and Cloud Computing complement each other in several areas. Cloud Computing provides IoT systems with an efficient and flexible environment that supports application requirements such as real-time control/monitoring, scalability, fault tolerance, and numerous security services. Hardware and software limitations of IoT devices can be mitigated using the massive on-demand cloud resources. However, IoT cloud-based solutions pose some security and privacy concerns, specifically when an untrusted cloud is used. This calls for strong encryption schemes that allow operations on data in an encrypted format without compromising the encryption. This paper presents an asymmetric multi-key and partially homomorphic encryption scheme. The scheme provides the addition operation by encrypting each decimal digit of the given integer number separately using a special key. In addition, data integrity processes are performed when an untrusted third party performs homomorphic operations on encrypted data. The proposed work considers the most widely known issues like the encrypted data size, slow operations at the hardware level, and high computing costs at the provider level. The size of generated ciphertext is almost equal to the size of the plaintext, and order-preserving is ensured using an asymmetrical encryption version.

Keywords: lightweight cryptography; homomorphic encryption; multi-key encryption; privacy-preserving; cloud data integrity



Citation: Medileh, S.; Laouid, A.; Hammoudeh, M.; Kara, M.; Bejaoui, T.; Eleyan, A.; Al-Khalidi, M. A Multi-Key with Partially Homomorphic Encryption Scheme for Low-End Devices Ensuring Data Integrity. *Information* **2023**, *14*, 263. <https://doi.org/10.3390/info14050263>

Academic Editors: Moutaz Alazab and Ammar Alazab

Received: 10 March 2023

Revised: 23 April 2023

Accepted: 26 April 2023

Published: 28 April 2023



Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

1. Introduction

In the current technology-driven world, such as smart cities, metaverse, 5G, and others, security is considered the crucial element that users, developers, and researchers are concerned about [1]. Homomorphic Encryption (HE) is a type of encryption that allows computations to be performed on ciphertexts without revealing their plaintext. The obtained results can be decrypted only by the owner of the secret key. HE can resolve many security and privacy issues in various technologies and applications. One of the common practical applications of HE is protecting data on the cloud. The power of HE allows users to profit from an untrusted cloud provider's huge computation and storage.

The homomorphism concept ensures secure data processing in regulated industries, such as financial services and healthcare, via the use of the data without access to its decrypted content. This aspect can also be exploited in other applications like the Internet of Medical Things (IoMT), where HE offers predictive analytics of medical data without compromising data privacy. In fact, HE may meet other services such as maintaining

customer privacy in personalized advertising, financial privacy of functions such as market forecasting and image recognition, and forensic investigations. Election transparency frequently uses HE, where additive encryption systems are suitable for voting applications. These systems allow voters to add different values impartially while keeping their private values and protecting data from manipulation. HE also has been used in cryptocurrency [2], a peer-to-peer currency issued without passing through a central bank. Cryptocurrency is used through a decentralized computer network and managed with a confidential ledger by blockchain users, which lists all transactions since the origin. There are numerous applications in fields where data privacy is of utmost importance. In this list, we explore some of the domains that have widely exploited homomorphic encryption in the literature:

- Cloud computing: Homomorphic encryption has been used to enable secure computation of sensitive data in the cloud without revealing the data to the cloud provider.
- Machine learning: Homomorphic encryption can be used to perform secure and private machine learning on encrypted data without the need for decryption.
- Blockchain: Homomorphic encryption can be used to enable secure and private transactions on a blockchain without revealing the transaction details to third parties.
- Privacy-preserving data analysis: Homomorphic encryption can be used to perform privacy-preserving data analysis on encrypted data without the need for decryption or data sharing.
- Internet of Things: Homomorphic encryption can be used to enable secure and private computation on IoT devices without the need for decryption or data sharing.

For instance, integrating the Internet of Things with blockchain has gained significant attention in industry and academia. These two technologies can provide trusted, secure decentralized data storage and reliable communication in various domains, such as healthcare, finance, and industrial systems. However, there is a risk of privacy leakage of sensitive information in the centralized IoT system because the centralized servers can access the plain text data from the IoT devices. Homomorphic encryption (HE) has been integrated with blockchain-based IoT systems to provide high privacy and security. Recently, HE has become particularly relevant in healthcare, where sensitive medical data are collected from various IoT devices. With the integration of HE, the collected data can be encrypted before being sent to the blockchain network, and computations can be performed on the encrypted data without decrypting it. This integration ensures that sensitive patient data remains private and secure, even when it is analyzed or processed by third-party applications. The reason for integrating blockchain-based IoT with HE is to provide a decentralized access model, i.e. in that the data will not be stored in a centralized server and that the owner of the private key controls access to the data. Additionally, integrating HE with blockchain-based IoT systems can provide tamper-proof data storage, where any changes to the data can be detected and traced back to the source.

Hence, any proposed crypto-system for IoT with HE should:

1. Provide high security and privacy for the data.
2. Be scalable and handle large amounts of data from various IoT devices.
3. Be efficient and not add significant overhead to the computational resources required for data processing.
4. Be compatible with existing support interoperability between connected low-end devices.
5. Define a user-friendly mode and easy to implement for developers and end-users.

The crucial question in this paper is which is the most suitable HE scheme dedicated to the IoT environment. Classical symmetric key encryption systems have the disadvantage that the user must have a secure private channel to transfer the encryption key to the receiver, which, when compromised, may expose all exchanged data. Furthermore, several symmetric key encryption schemes provide a weak digital signature method [3]. Asymmetric homomorphic encryption has the potential to enable secure computation of data while keeping it confidential. However, one of the main challenges of implementing

asymmetric homomorphic encryption is the computation complexity associated with the encryption and decryption operations.

Despite these challenges, research in asymmetric homomorphic encryption continues to progress, and new schemes are being proposed to address the computation complexity and security issues. This paper extends and completes the work in [4] to present an improved Partially Homomorphic Encryption (PHE) asymmetric scheme based on the Polynomial Reconstruction Problem. The completed design of PHE offers order-preserving capabilities, making it suitable for IoT-constrained devices. In this proposed multi-key encryption scheme, any decimal number will be fragmented into digits, and each digit will be multiplied by a key using several small secret keys. An asymmetric version that allows an efficient range of queries on encrypted data is proposed as a secondary goal to ensure the order-preserving aspect of the proposed scheme. The order-preserving scenario is that the owner may ask the untrusted cloud to return ciphertexts in the database whose decryptions are in a given range $[x; y]$. The current version also introduces 'data integrity ensuring' after performing homomorphic functions by an untrusted third party. Thanks to the extended version, the client can easily verify the results validity of operations executed by the cloud without the need to execute all these operations.

The rest of the paper is organized as follows: Section 2 shows relevant related work. In Section 3, we present the features of the proposed partially homomorphic encryption scheme and formulate the depth of operations. At the end of this section, we explain and demonstrate the order-preserving propriety of the proposed scheme. Section 4 describes how the proposal ensures data integrity after an untrusted cloud performs homomorphic addition. In Section 5, we analyze the hardness level of the proposed crypto-system. Section 6 dictates the implementation results and shows the efficiency compared with others. Finally, Section 7 concludes the paper.

2. Related Work

The ever-growing data generated by increasingly connected environments require new measures to protect user privacy, security, and safety [5]. Many encryption mechanisms have been designed to protect the privacy of user data in storage and during communication. This section analyzes relevant existing HE cryptosystems in the literature.

In this Section, we will categorize these schemes based on various properties, such as their security assumptions, the types of homomorphic operations they support, and their computational efficiency. Hereafter, we will review some of the most relevant work in the literature.

2.1. Security Assumptions

The security assumptions categorize multi-key with partially homomorphic encryption schemes based on the underlying mathematical problems that provide their security. These problems can come from various fields of mathematics, such as number theory, coding theory, and lattice theory. The security of these schemes is based on the assumed hardness of solving these mathematical problems. This section highlights three common types of security assumptions: Public key encryption, lattice-based encryption, and code-based encryption. By understanding the security assumptions of these schemes, we can better evaluate their suitability for different applications and potential vulnerabilities.

2.1.1. Public Key Encryption

In this category, we have encryption schemes that rely on the security of a public key encryption algorithm. Examples include the Paillier cryptosystem and the BGN cryptosystem. These schemes typically rely on the hardness of the Decisional Composite Residuosity (DCR) or the Decisional Diffie-Hellman (DDH) problem.

2.1.2. Lattice-Based Encryption

Lattice-based encryption schemes use the hardness of specific lattice problems to provide security. Examples include the Gentry-Sahai-Waters (GSW) scheme and the Brakerski-Gentry-Vaikuntanathan (BGV) scheme. These schemes are typically based on the Learning with Errors (LWE) problem or the Ring Learning with Errors (RLWE) problem.

2.1.3. Code-Based Encryption

Code-based encryption schemes rely on the hardness of certain coding theory problems to provide security. Examples include the McEliece cryptosystem and the Niederreiter cryptosystem. These schemes are typically based on the hardness of decoding a random linear code or a random quadratic residue code.

2.2. Homomorphic Operations

Homomorphic operations allow computations to be performed on encrypted data without the need for decryption. We highlight three common types of homomorphic operations: Additive homomorphic encryption, multiplicative homomorphic encryption, and fully homomorphic encryption. Additive homomorphic encryption allows for the homomorphic addition of ciphertexts, while multiplicative homomorphic encryption allows for the homomorphic multiplication of ciphertexts. Fully homomorphic encryption allows for both homomorphic addition and multiplication of ciphertexts. By understanding the types of homomorphic operations these schemes support, we can better evaluate their usefulness for applications requiring secure computation on encrypted data.

2.3. Computational Efficiency

We divide the computation efficiency into two types of computational efficiency: Asymptotic efficiency and practical efficiency. Asymptotic efficiency refers to the theoretical running time of an algorithm as the size of the input grows. In contrast, practical efficiency refers to the actual running time of an algorithm on real-world data. By understanding the computational efficiency of these schemes, we can better choose the appropriate scheme for a given application, considering the trade-off between security and computational cost.

Multi-key with partially homomorphic encryption schemes are based on their security assumptions, the types of homomorphic operations they support, and their computational efficiency. By understanding the different properties of these schemes, we can better choose the appropriate scheme for a given application. HE includes a variety of schemes that allow arbitrary computation over encrypted data [6]. The difference between them is related to the types and the periodicity of mathematical operations that can be performed. The common types of HE schemes are Partially Homomorphic Encryption (PHE), Leveled Fully Homomorphic Encryption (LFHE), Somewhat Homomorphic Encryption (SHE), and Fully Homomorphic Encryption (FHE).

PHE is a type of encryption in which only one operation, such as addition or multiplication, can be performed on the ciphertext. This type includes the RSA [7] and El-Gamal [8] cryptosystems, both of which are multiplicative schemes. Common additive cryptosystems include Naccache and Stern 1998 [9], Paillier 1999 [10], Galbraith 2002 [11] and Kawachi et al., 2007 [12].

LFHE supports finite operations over ciphertexts with a limited number of multiplication and addition [13], mainly from the perspective of circuit depth. The circuit depth is predetermined in the setup algorithm. SHE is another term sometimes used interchangeably with LFHE [6,14]. The authors in [15] present an SHE scheme and analyze for which parameters the scheme is correct and how many homomorphic operations can be performed before decryption fails to ensure correct decryption in the presented scheme. Both homomorphic operations (addition and multiplication) can be achieved, but only for a limited number of times [15]. In this family of HE, the works of Boneh et al., 2005 [16] and Ishai and Paskin 2007 [17] are widely known in the literature. The first cryptosystem in FHE is the Gentry 2009 scheme [18]. Other proven mechanisms were published by

Van Dijk et al., 2010 [19], Brakerski and Vaikuntanathan 2011 [20], and López-Alt et al., 2012 [21].

To satisfy the Paillier cryptosystem [10] for data mining that preserves privacy, paper [22] presented a homomorphic technique. This scheme allows multiple cloud users to have different public keys. The proposed variant-Paillier cryptosystem is from $c = g^m \times r^n \bmod N^2$ to $c = g^m \times h^r \bmod N^2$ with conditions on the selection of the integers N and g . The smallest common factor prime of the values of the Euler function of the large prime numbers p and q is set as the public key. The parameter h is a component of the public key. The problem with this scheme is that it is probabilistic, i.e., we can find $c_1 = \text{Enc}(m_1)$ and $c_2 = \text{Enc}(m_1)$ with $c_1 \neq c_2$, this allows an attacker to extract some secret information.

In [23], the authors implemented an LFHE technique. Using a public key k , the encryption function of a bit $b \in \{0, 1\}$ is represented by $c = k \times r_1 + 2 \times r_2 + b$, where r_1 and r_2 are random. The decryption is as follows: $m = c \times f \bmod 2$ where f is the secret key. A re-linearization operation must be performed during multiplication.

The authors of [24] proposed a fully HE scheme based on a symmetric key. Their encryption function is defined as follows: $C = K^T \times d(M; 1; 2; 3) \times K$, where K^T is the transpose of matrix K of dimension 4, $d(M; 1; 2; 3)$ denotes the diagonal matrix with diagonal elements as parameters. This cryptosystem has a refresh procedure whereby the key is refreshed periodically to maintain forward, and backward, secrecy ($K' \leftarrow \text{randortho}(t)$; $\text{randortho}(x)$ is a randomized function that generates random orthonormal new matrix K of dimension t); the refresh procedure can slow down homomorphic operations. In addition, this technique is symmetrical, which will impose certain restrictions on their use. A targeted fully HE was proposed in [25]. Based on El-Gamal, the authors used a ciphertext of three parts. The first is $c_1 = \alpha^k$ where k is random for each value to be encrypted, and c_1 is exploited to decrypt parts one and two. The second part is $c_2 = m \times \beta^k$, knowing that $\alpha^p = \beta$ (where p is the secret key) and $pk = (\alpha, \beta)$, the third part $c_3 = \beta^{k+m}$ has been added to ensure the addition of two ciphertexts. The major problem with this technique is using a discrete logarithm to decrypt $\beta^{m_1+m_2}$ to obtain $m_1 + m_2$. It will take a very long time, and hence it minimizes the size of m .

In 2018, a fully HE technique was proposed in [26]. The encryption function is $c_i = (m_i + \text{rand}_i \times k) \bmod (k \times p)$, where $\text{rand}_i = (m_i \times k) \bmod p$, the decryption is very simple as follows, $f^{-1} : M_i = c_i \bmod k$. To preserve order, the authors used a linear expression shown by the equation: $\text{index}_i = p \times m_i + \text{rand}_i$. In reality, this system is vulnerable to a known-plaintext attack, where the attacker has both the ciphertext and its plaintext. If $c_i = m_i + \text{rand}_i \times k$, then $\text{rand}_i \times k = c_i - m_i = x$, where x is known. By knowing the public key $pk = k \times p$, the attacker just has to do a successive division to extract the value of the secret key k .

A novel fully HE scheme based on learning with errors (LWE) is presented in [27]. To avoid the complex matrix operations of the existing key switching mechanism, the authors modified the re-linearization method developed by Brakerski et al., [28] and improved a new technique called non-matrix key switching. This proposed mechanism includes key switching with re-linearization and pure key switching. Firstly, the authors built a leveled fully HE scheme without bootstrapping from LWE, then transformed it into FHE. The technique has improved compared to the Brakerski scheme, but the key switching time remains non-negligible in certain applications.

Most schemes suffer from ciphertext size or run-time. The authors in [29] propose a general construction of MKFHE scheme with compact ciphertext. They proceed by accumulating each party's public key under the CRS model to create the accumulated public key of the parties set with compact, after which all parties provide the ciphertext of their secret keys, which is encrypted by the accumulated public key; that is then used as the accumulated evaluation key. Next, they refresh the ciphertext by running the key-switching process on each party's ciphertext and accumulating the evaluation key. Eventually, they homomorphically calculate the refreshed ciphertext and decrypt it using the joint secret key.

This paper proposes an asymmetric PHE scheme with feasibility in an IoT environment where these devices store their data in the cloud.

3. Cryptosystem Design

An interaction scenario in homomorphic encryption is the client's desire to perform computational operations with unreliable outsourcing. The client must first use a function that allows operations to be performed on encrypted values. After that, it has to encrypt the input values before sending it and decrypt the cloud result. In Figure 1, the user wants to compute the result of $\alpha \theta \beta$, defying untrusted providers and insecure channels. Equation (1) can be used to improve the security level of the scheme and obtain reliable results. Where ψ denotes ciphertext, π denotes plaintext, k denotes the secret key, and r denotes a random number.

$$\psi = (\pi \times k + r \times p) \bmod n \quad (1)$$

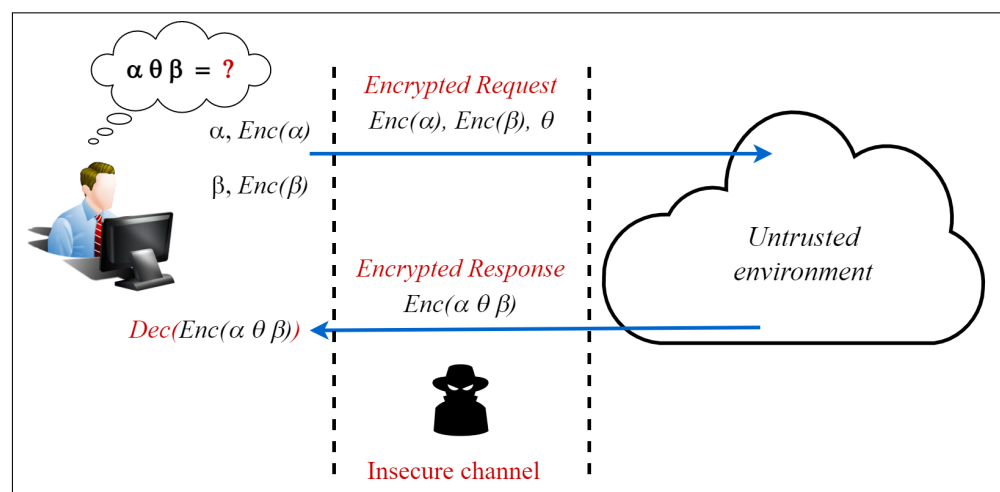


Figure 1. Interaction scenario in Homomorphic Encryption.

We consider the plaintext m as a set of decimal digits and separately manage each digit m_i where $0 \leq m_i \leq 9$. Finally, we multiply each digit m_i by a secret key sk_i . The use of public keys (pk_i) gave us the asymmetric version of this technique, and it is worth noting here that k_i must be small.

For construction:

- KeyGen: (pk_i, sk_i) is equal $((k_i + r_i \times p, n), (k_i, p))$
- $Enc(m) : c = m_0 \times pk_0 + m_1 \times pk_1 + \dots m_i \times pk_i$, where $pk_i < pk_{i+1}$
- $Dec(c) : m = \sum_{i=1}^0 (\frac{c}{k_i}) \times 10^i, c \leftarrow c - c \times k_i$, with $(\frac{c}{k_i})$ presents the quotient of $c \div k_i$.

3.1. System Description

The proposed scheme is based on treating each decimal digit of the plaintext separately (Figure 2). We do not consider the carry bit because we perform the operations on the whole number rather than the bits.

3.1.1. Keys Generation

Let s be the number of digits of m . We propose the following equation:

$$k_0 > 10, k_j > 9 \times \sum_{i=0}^{j-1} k_i \quad \forall j > 0 \quad (2)$$

Equation (2): In the decryption operation, we calculate the quotient of $c \div k_i$; we use a recursive process in which we eliminate the largest element in the sum of ciphertext to

calculate the quotient; after that, we eliminate the quotient to recover the largest element; finally, we calculate the factor which is multiplied by its secret key. Therefore, the secret key k_1 must be greater than $9 \times k_0$, and k_2 must be greater than $9 \times k_0 + 9 \times k_1$ (9 represents the largest possible value of m_i). For further clarification, consider the following example: $c = 2 \times k_0 + 3 \times k_1 + 7 \times k_2$; to calculate the value 7, we calculate c/k_2 which is equal to $(2 \times k_0 + 3 \times k_1)$ if k_2 is greater than $(2 \times k_0 + 3 \times k_1)$; after, we calculate $c - \frac{c}{k_2}$ which is equal to $7 \times k_2$; finally, we can extract the value 7 by calculating $7 \times k_2 \div k_2$, and so on. When this example is generalized, we get Equation (2).

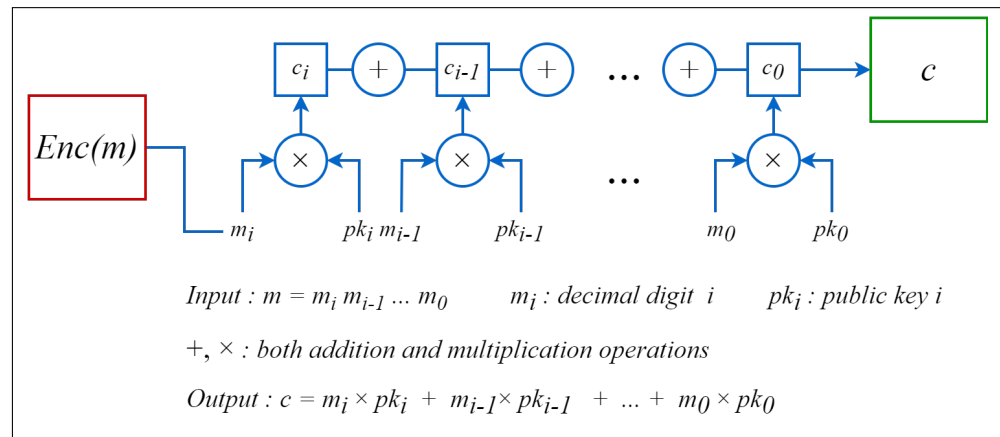


Figure 2. Proposed encryption scheme.

To simplify the key generation, we propose a programmable formula. If $k_j = h_j + 9 \times \sum_{i=0}^{j-1} k_i$ with h_j denotes a small random number, and $h_i = 1 \forall i > 1$, $h_j = 1 \Rightarrow k_j = 1 + 9 \times \sum_{i=0}^{j-1} k_i = (1 + 9 \times k_0) + 9 \times k_1 + \dots + 9 \times k_{j-1}$.

Posing $(1 + 9 \times k_0) = k_1 \Rightarrow k_j = k_1 + 9 \times k_1 + \dots + 9 \times k_{j-1}$.

Posing $k_1 + 9 \times k_1 = k_2 \Rightarrow k_j = k_2 + 9 \times k_2 + \dots + 9 \times k_{j-1}$ etc.

Therefore, $k_j = 10 \times k_{j-1}$.

$$k_0 > 10, k_1 = 1 + 9 \times k_0, k_j = 10^{j-1} \times k_1 \quad (3)$$

If $c = m_0 \times k_0 + m_1 \times k_1 + \dots + m_{s-1} \times k_{s-1}$, then $c < k_s = 10^s \times k_0$.

Let d denotes the addition depth, with $C = \sum_{i=1}^d c_i$, k_i coefficient should be less than k_i that implies $k_0 > \sum_{i=1}^d m_i$, if $\max(m_i) = 9 \Rightarrow k_0 > \sum_{i=1}^d 9 \Rightarrow k_0 > 9 \times d$. With $\sum_{i=1}^d c_i < p$ and $c < 10^s \times k_0$ that implies $d \times (10^s \times k_0) > d \times c$ that implies $p > d \times (10^s \times k_0)$, where $k_0 > 9 \times d$ i.e., $p > 9 \times d^2 \times 10^s$.

Finally, the parameter conditions are:

- $s = \text{size}(M)$, d is the number of addition operations, and h denotes a small random number.
- $k_0 > 10$, $k_j = h_j + 9 \times \sum_{i=0}^{j-1} k_i$.
- In addition operation: $k_0 = h + 9 \times d$, $p = h' + d \times (10^{s-1} \times k_1)$.

The KeyGen function, see Algorithm 1 is demonstrated using Equation (3) where $\text{size}(M)$ designates the number of digits of the plaintext ring M written in decimal. It is worth noting that $pk_i = k_i + r_i$, where r_i is a random number.

Algorithm 1 KeyGen algorithm**Require:** M, k_0 **Ensure:** $(k_1, k_2 \dots k_j)$

```

1: function KEYGEN
2:    $s \leftarrow \text{size}(M)$ 
3:    $k_1 \leftarrow 1 + 9 \times k_0$ 
4:   for  $i \leftarrow 2$  to  $s - 1$  do
5:      $k_i \leftarrow 10^{i-1} \times k_1$ 
6:   end for
7:   return  $(k_1, k_2 \dots k_{s-1})$ 
8: end function

```

3.1.2. Encryption

As shown in Algorithm 2, if $m = m_j m_{j-1} \dots m_0$ with m_0 denotes the coefficient of 10^0 , $m_i \in \{0, \dots, 9\} \forall i \in \{0, \dots, j\}$, and $m_j \neq 0$.

$c = m_0 \times pk_0 + m_1 \times pk_1 + \dots m_j \times pk_j$ so $c = m_0 \times (k_0 + r_0 \times p) + m_1 \times (k_1 + r_1 \times p) + \dots m_i \times (k_j + r_j \times p)$.

We get :

$$c = (m_0 \times k_0 + m_1 \times k_1 + \dots m_j \times k_j + r \times p) \mod n \quad (4)$$

with $r = m_0 \times r_0 + m_1 \times r_1 + \dots m_i \times r_j$.

Noting that: $m_0 \times k_0 + m_1 \times k_1 + \dots m_j \times k_j < p$.

Algorithm 2 Encryption algorithm**Require:** m_i, pk_i, n **Ensure:** (c, t)

```

1: function ENC2
2:    $s \leftarrow \text{size}(m)$ 
3:    $c \leftarrow 0$ 
4:   for  $j = 0$  to  $s$  do
5:      $c \leftarrow \text{Enc}_{pk_j, n}(m)$ 
6:      $t \leftarrow pk_0^m \mod n$ 
7:   end for
8:   return  $(c, t)$ 
9: end function

```

3.1.3. Decryption

To get the original digits of plaintext m from the ciphertext c (Figure 3), the last part $r \times p$ must be eliminated by computing the modulo operation. Then, decreasing i , c must be divided successively on k_i . Finally, the obtained digits must be multiplied successively by 10^i (decreasing i).

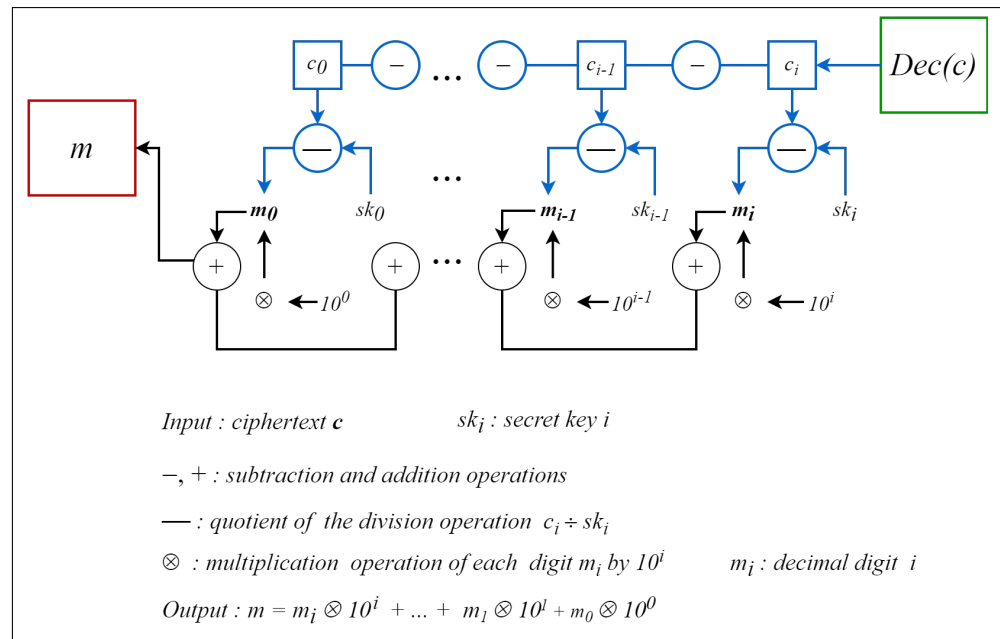


Figure 3. Proposed decryption scheme.

Lemma 1. If $\frac{c}{k_i} = m_i \Rightarrow Dec(c) = m$.

Proof. Lemma 1 $\Leftrightarrow m_0 \times k_0 + m_1 \times k_1 + \dots m_{j-1} \times k_{j-1} < m_j \times k_j \forall m_i \ 0 \leq m_i \leq 9$, that is $\frac{m_0 \times k_0 + m_1 \times k_1 + \dots m_{j-1} \times k_{j-1}}{k_j} = 0$.

Let $m_i = 9 \forall i \in \{0, \dots, j-1\}$ and $m_i = 1$ for $i = j$ that implies $9 \times k_0 + 9 \times k_1 + \dots 9 \times k_{j-1} < k_j$.

So, $9 \times (k_0 + k_1 + \dots k_{j-1}) < k_j$ i.e., $9 \times \sum_{i=0}^{j-1} k_i < k_j \Rightarrow \frac{c}{k_j} = m_j$.

The subtraction: $c \leftarrow c - c \times k^j$ must be calculated. So, $c = m_0 \times k_0 + \dots m_{j-1} \times k_{j-1}$ that implies $\frac{c}{k_{j-1}} = m_{j-1}$. Thus, we will get m_0, m_1, \dots, m_j ; finally, we can calculate $m = m_0 \times 10^0 + m_1 \times 10^1 + \dots m_j \times 10^j$. \square

To retrieve m , the division must be started on k^j with $j = size(c)$. Let $i = size(m)$, that implies $j > i$ because $c > m$. Furthermore, the decryption operation is correct for $m_i < k_0$.

3.2. Homomorphic Addition

In practice, additive HE is efficient enough to be used, although it has limits in terms of operations. The goal is to guarantee the additive property of the proposed scheme. When two plaintexts are added, the additive homomorphic scheme realizes $Enc(m_1 + m_2) = Enc(m_1) + Enc(m_2)$.

Let $i < j$, $c = m_0 \times k_0 + m_1 \times k_1 + \dots m_i \times k_i + r \times p$, $c' = m'_0 \times k_0 + m'_1 \times k_1 + \dots m'_j \times k_j + r' \times p$, so $Enc(m) + Enc(m') = (m_0 + m'_0) \times k_0 + (m_1 + m'_1) \times k_1 + \dots (m_i + m'_i) \times k_i + \dots m'_j \times k_j$.

If: $(m_x + m'_y) < k_s \forall x \in \{0, \dots, i\}, y \in \{0, \dots, j\}, s \in \{0, \dots, j\}$, $\frac{Enc(m) + Enc(m')}{k_v} = m_v \forall v \in \{0, \dots, j\}$ that implies $Dec(c + c') = m + m'$.

Algorithm 3 could define the addition operation.

Algorithm 3 Addition 1 algorithm

Require: c_1, c_2, n

Ensure: $c_3 = Add(c_1, c_2)$

```

1: function ADD
2:    $c_3 \leftarrow (c_1 + c_2) \bmod n$ 
3:   return  $c_3$ 
4: end function

```

Depth Demonstration

Table 1 shows some practical examples. In this test (size in digits), the size of the private key p is limited to 2 kbit.

Table 1. Number of addition.

Size (Digits)	m	k_0	p	Depth
	2	301	603	10^{300}
	5	301	606	10^{300}
	10	301	611	10^{300}
	20	301	621	10^{300}
	60	301	661	10^{300}
	100	251	601	10^{250}
	200	201	601	10^{200}

3.3. Order-Preserving

When the order of encrypted data must be preserved, an order-preserving scheme should be proposed as an ideal solution. In the proposed cryptosystem, an order-preserving approach can index and process unencrypted data, allowing an efficient range of queries. The data server can locate ciphertexts in logarithmic time through standard tree data structures. To avoid the treatment of key management issues, we assume that the message to be encrypted should not be too large.

We can exploit the proposed technique as a symmetric scheme, where there are no public keys, neither n ($n = p \times q$) nor pk_i ($pk_i = k_i + r \times p$). It should just have $sk_i = k_i$ with this version to preserve order, i.e., $m > m' \Rightarrow c > c'$.

Lemma 2. $m > m' \Rightarrow Enc(m) > Enc(m')$

$\forall m, m' \leq M$.

Proof. Let $\begin{cases} m = m_j m_{j-1} \dots m_0, 0 \leq m_i \leq 9 \forall i \\ m' = m'_{j'} m'_{j'-1} \dots m'_0, 0 \leq m'_{i'} \leq 9 \forall i' \end{cases}$

$m > m'$ that implies $\begin{cases} j > j' \dots (A) \\ or \\ j = j' \text{ and } m_j > m'_{j'} \dots (B) \end{cases}$

$(A) \Rightarrow \begin{cases} c = m_0 \times k_0 + m_1 \times k_1 + \dots m_j \times k_j \\ c' = m'_0 \times k_0 + m'_1 \times k_1 + \dots m'_{j'} \times k_{j'} \end{cases}$

Let, $\begin{cases} m_i = 0 \forall i < j \text{ and } m_j = 1 \\ j = j' + 1 \\ m'_{i'} = 9 \forall i' \leq j' \end{cases}$

$(A) \Rightarrow \begin{cases} c = k_j \\ c' = 9 \times \sum_{i'=0}^{j'} k_{i'} = 9 \times \sum_{i'=0}^{j-1} k_{i'} \end{cases}$

$$\begin{aligned}
& \text{Knowing that: } k_j > 9 \times \sum_{i=0}^{j-1} k_i \text{ so } c > c' \\
(B) \Rightarrow & \begin{cases} c = m_0 \times k_0 + m_1 \times k_1 + \dots m_j \times k_j \\ c' = m'_0 \times k_0 + m'_1 \times k_1 + \dots m'_j \times k_j \end{cases} \\
& \text{Let, } \begin{cases} m_i = 0 \forall i < j \text{ and } m_j = m'_j + 1 \\ m'_i = 9 \forall i < j \end{cases} \\
(B) \Rightarrow & \begin{cases} c = m'_j \times k_j + k_j \\ c' = m'_j \times k_j + 9 \times \sum_{i=0}^{j-1} k_i \end{cases} \\
& \text{Knowing that: } k_j > 9 \times \sum_{i=0}^{j-1} k_i \text{ that implies } c > c' \quad \square
\end{aligned}$$

So, in any case, if we have $m > m'$ we will get $c = \text{Enc}(m) > c' = \text{Enc}(m')$, then the symmetric version of our proposal preserves the order in an additive HE.

4. Data Integrity Ensuring

Homomorphic encryption techniques are particularly useful in cloud computing environments, where sensitive data is often stored and processed on remote servers. However, HE can be vulnerable to attacks that compromise the integrity of the encrypted data. For example, an attacker could modify the encrypted data in a way that would cause the computation to produce incorrect results when the data is decrypted. This attack could have serious consequences, such as financial losses and data breaches. To prevent these types of attacks, data integrity is certainly crucial in cloud homomorphic encrypted data. Furthermore, for a data owner, data integrity (DI) [30] is a very important and sensitive point in the design, implementation, and use of any data system when he stores its data and then processes or retrieves it, especially if the matter comes to the cloud. DI can be defined as the validity, completeness, accuracy, and consistency of data (Figure 4). This also includes data integrity in terms of privacy and security. After proper data validation and error checking, the owner can ensure that sensitive data is not used, exploited, improperly classified, or stored incorrectly. All incorrect changes to the data due to storage, computation, or retrieval operations, including unexpected hardware failure, malicious intent, or human errors, will inevitably lead to a fatal error in exploiting this data and its use later.

DI can be easily guaranteed in local databases to prevent intentional information changes. For example, it can first be ensured that internal users will handle the data correctly and harmlessly. However, it will be much more difficult when using a third party (untrusted cloud) to make operations on the encrypted data.

While validation of these homomorphic calculations is a prerequisite for data integrity, we will modify the proposal by adding a second part of the cipher, as this part will prevent any manipulations of homomorphic processing outcomes by a third party, whether these changes are intentional or unintentional. The ciphertext will now be in two parts; the first part provides linear message encryption allowing the cloud to compute the sum of the two first parts. The cloud uses the second part to prove its sincerity in calculation processes. This second part depends on discrete logarithm hardness, where the public key is raised to the power of the plaintext. Therefore, the cloud cannot access the value of the original message in order to change the result of the sum operation.

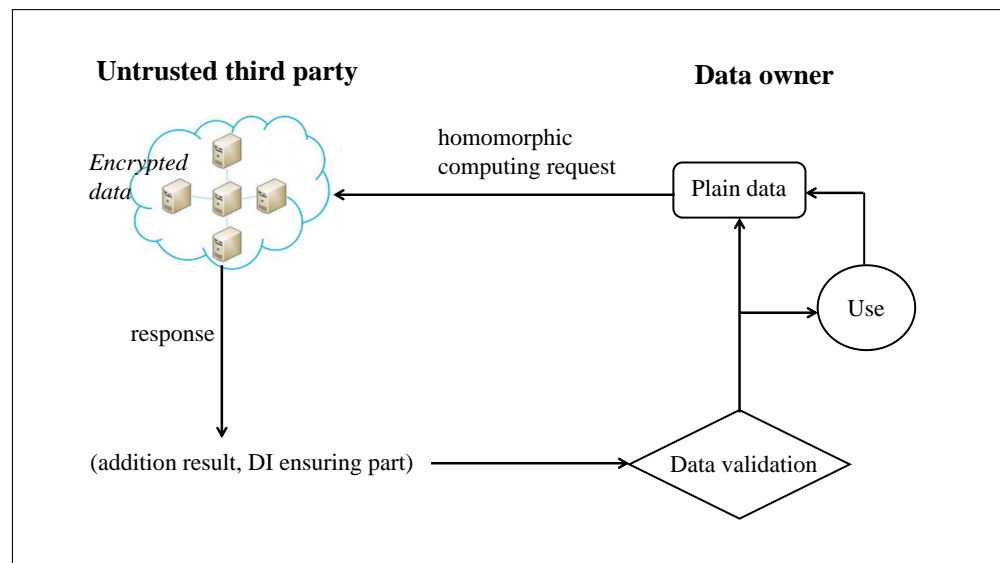


Figure 4. Data integrity ensuring architecture.

4.1. Scenario

Suppose a data owner requests the cloud to accomplish a homomorphic addition operation of two ciphertexts. After obtaining that c_3 ($c_3 = c_1 + c_2$), the data owner will utilize c_3 considering that it is a valid compute, i.e., c_3 was not manipulated. Nevertheless, the untrusted cloud may have changed that computed result, whether by error or deliberately. The data owner cannot 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 $Dec(c'_z) = z' \neq x + y$!

The proposed encryption scheme allows the user to confirm whether the computed result of a homomorphic operation transmitted by the cloud is valid and can be used later. In the proposal, the cloud must multiply the second parties for each addition homomorphic operation, i.e., computing $t_z = t_x \times t_y$.

After decrypting c'_z as shown in Algorithm 4 and getting z , the data owner has to calculate $t'_z = pk_0^z$. If $t'_z = t_z$, the computed result is valid.

Algorithm 4 Decryption algorithm

Require: c, t, k_i, p

Ensure: m

```

1: function DEC2
2:    $m \leftarrow Dec_{k_i, p}(c)$ 
3:    $t' \leftarrow pk_0^m \bmod n$ 
4:   if  $t' = t$  then
5:     return  $m$ 
6:   else
7:     return error
8:   end if
9: end function
  
```

So, the addition Algorithm 5 will be as follows:

Algorithm 5 Addition 2 algorithm

Require: c_x, c_y, t_x, t_y, n

Ensure: c_z, t_z

```

1: function ADD2
2:    $c_z \leftarrow c_x + c_y \bmod n$ 
3:    $t_z \leftarrow t_x \times t_y \bmod n$ 
4:   return  $(c_z, t_z)$ 
5: end function

```

4.2. Demonstration

Let $t_x = pk_0^x \bmod n$ and $t_y = pk_0^y \bmod n$.

As illustrated in Algorithm 5, computing $c_z = c_x + c_y \rightarrow$ computing $t_z = t_x \times t_y$.

$\rightarrow t_z = pk_0^{x+y} \bmod n$.

$Dec(c_z) = z = x + y, t'_z = pk_0^{Dec(c_z)} \bmod n = pk_0^z = pk_0^{x+y}$.

So, if $t'_z = t_z$ then the DI of homomorphic addition is guaranteed.

Noting here that the third part cannot manipulate $t_x \times t_y$ because t_x (respectively t_y) contains x (respectively y) as an exponential parameter, and c_i uses m_i in a linear addition-multiplication. Thus, it is impossible to simultaneously manipulate c_z and t_z without having data inconsistency.

5. Proof of Security and Performance

The proposed technique has the secret keys, k_i , and the trapdoor p . If an adversary gets p , he will get all k_i using the public keys where $pk_i = k_i + r_i \times p$. Thus, to obtain p , the adversary has to solve the factorization problem, which cannot be solved in polynomial time. If λ is the parameter of security with $2^\lambda < p < 2^{\lambda+1}$, so the cryptanalysis requires 2^λ operations to get p .

Known-plaintext attack: When the adversary has m and c and tries to get the secret keys. If $size(m) = 1$ digit, the adversary cannot obtain anything, $c = m \times pk$ where c, m , and pk are known. In addition, $c \times pk^{-1} = m \times pk \times pk^{-1} \Rightarrow \alpha = m + \beta \times n \Rightarrow \alpha \bmod n = (m + \beta \times n) \bmod n \Rightarrow m = m$ with β is independent of k ; therefore, the adversary will have no information. If $size(m) = j$ where $j > 1$. For example $j = 2, c = m_0 \times pk_0 + m_1 \times pk_1$, if the adversary computes pk_0^{-1} and pk_1^{-1} , $c \times pk_0^{-1} = m_0 + m_1 \times pk_1 \times pk_0^{-1}$ that implies $\alpha + \beta_1 \times n = \alpha + \beta_2 \times n$ so $\alpha = \alpha$ with β_1 and β_2 are independent of k_0 and k_1 ; the adversary will have no information. If the adversary has m and m' where $size(m) = size(m') = 1$, there is no information. Let, $size(m) > 1$ and $size(m') > 1$,

$$\begin{cases} c = m_0 \times pk_0 + m_1 \times pk_1 \\ c' = m'_0 \times pk_0 + m'_1 \times pk_1 \end{cases}$$

$$\text{So, } \begin{cases} m'_0 \times c = m'_0 \times m_0 \times pk_0 + m'_0 \times m_1 \times pk_1 \\ m_0 \times c' = m'_0 \times m_0 \times pk_0 + m_0 \times m'_1 \times pk_1 \end{cases}$$

To remove pk_1 , the coefficients $m'_0 \times m_1$ and $m_0 \times m'_1$ must be different. The adversary will get another value of the public key pk_1 .

So, to get information, it is insufficient to obtain m, m', c , and c' . If $size(m) = 3$ digits, to get p , the adversary must have three plaintexts with conditions and so on. If we use this technique as a symmetrical scheme (without $r \times p$), the hardness of our scheme will be based on the polynomial reconstruction problem, which can be written as $Enc(m) = F(k_i)$, where $F : M, K \rightarrow C$.

Brute force attack (BFA): In the RSA cryptosystem, the attacker must perform 2^d operations to get p with $m = c^d \bmod n$. In the proposed technique, if there are s keys, the attacker must perform $(2^{k_0})^s$ operations. So, there are two layers of security, to find m, c and to find sk . In the asymmetric version of the proposal, the attacker can not directly make an exhaustive search on k_i since it is hidden using $r \times p$ where $c = F(m, k) + r \times p$,

the attacker has to perform 2^p operations to get p . In order to protect against known attacks and mitigate quantum computing attacks, noisy encryption techniques can be exploited. These techniques involve using large key sizes, making it more difficult for attackers to factor the key using a classical or quantum computer. However, recent studies prove that quantum computing poses a significant threat to classical cryptographic systems such as RSA, which may break large keys in just a few hours [31]. Larger key sizes also increase the computational overhead and the size of the encrypted message. To further protect against these kinds of attacks, key rotation and nested encryption can be implemented by regularly changing the keys based on randomly chosen values and encrypting the message more than once using distinct keys. This method can help to mitigate the threat of quantum computing attacks, as an attacker would need to break multiple keys to compromise the security of the encryption scheme. By randomizing the keys, noisy encryption makes it more complex for an attacker to break the encryption scheme than a single key. These techniques have proven effective in securing data against various types of attacks.

Computation complexity: If $size(m) = j$, we have $C(j + 1) = C(j) + \alpha$ in the encryption operation where $C(j)$ denotes the complexity and α is a constant, if $maxsize(j) = size(M)$, then the time complexity is linear: $T(M) = O(M)$. The proposed technique performance in terms of complexity is shown in Table 2.

Table 2. Encryption and decryption complexity comparison relative to plaintext size.

Schemes	Enc	Dec
Proposed scheme	$O(\lambda)$	$O(\lambda)$
[32]	$O(\lambda^4)$	$O(\lambda^4)$
[24]	$O(\lambda^5)$	$O(\lambda^{4.8})$
[33]	$O(\lambda^6)$	$O(\lambda^5)$
[34]	$O(\lambda^6)$	$O(\lambda^5)$
[35]	$O(\lambda^{13})$	$O(\lambda^{12})$

Small Key and Ciphertext Sizes: The ciphertext size has great importance in cryptography because it is the most exchanged element between a sender and a receiver, where the plain message is converted into a ciphertext in order to send it through an unreliable channel. Unlike the public key, the secret key, or the private key, no matter how big it is, it will be exchanged between communicants no more than once. The size of the ciphertext is more important in low-energy environments such as the IoT, which are spreading more day by day and being applied in various fields. Therefore, a lot of cryptographic research focuses on creating techniques that enable the generation of ciphertexts of a small size, so that these techniques can be practical in the largest possible number of fields. Hence, we focused in the proposed scheme on this point and were able to create a relatively small ciphertext compared to other work.

The simplified formula as given in Equation (3) results in a relatively small key and ciphertext size.

Lemma 3. $size(c) = size(m) + \alpha$ with α is a constant.

Proof. If $size(m) = j$, knowing that $\begin{cases} m = m_0 \times 10^0 + m_1 \times 10^1 + \dots m_j \times 10^j \\ c = m_0 \times k_0 + m_1 \times k_1 + \dots m_j \times k_j \end{cases}$.

If we put $k_1 = 10 \times k_0$, we will get: $c = m_0 \times 10^0 \times k_0 + m_1 \times 10^1 \times k_0 + \dots m_j \times 10^j \times k_0$ that is $Enc(m)c = k_0 \times (m_0 \times 10^0 + m_1 \times 10^1 + \dots m_j \times 10^j) = k_0 \times m$ that implies $c = k_0 \times m$. So, $\alpha = size(k_0) - 1$. \square

6. Implementation and Comparative Analysis

The implementation was done in Python on a computer with an Intel Core i5-3230M CPU 2.6 GHz, 2 Core(s), and 8 GB RAM.

Whether in the encryption or decryption operation, Table 3 shows how much the proposed scheme's execution time was reduced compared to the other schemes when encrypting a 16-bit message. In the other schemes, the shortest encryption time [36] was estimated to be two times our encryption time. As for the fastest decryption operation [36], it was estimated to be over 290 times. This is due to the technique used in each scheme.

In the study represented in Figure 5, we performed many tests to calculate the encryption and decryption time if $size(m) = 16$ bits and $size(n) = 360$ bits. The decryption time was higher than the encryption time because there were five steps in the decryption process, while there were three steps in the encryption process. As well as that, the ciphered text size grows faster than the plaintext size.

Table 3. Execution time (ms).

Schemes	Enc	Dec
Proposed scheme	0.036	0.041
[36]	0.07	11.95
[22]	11.91	17.67
[37]	47	15
[38]	50	10
[39]	255	493
[40]	899	785

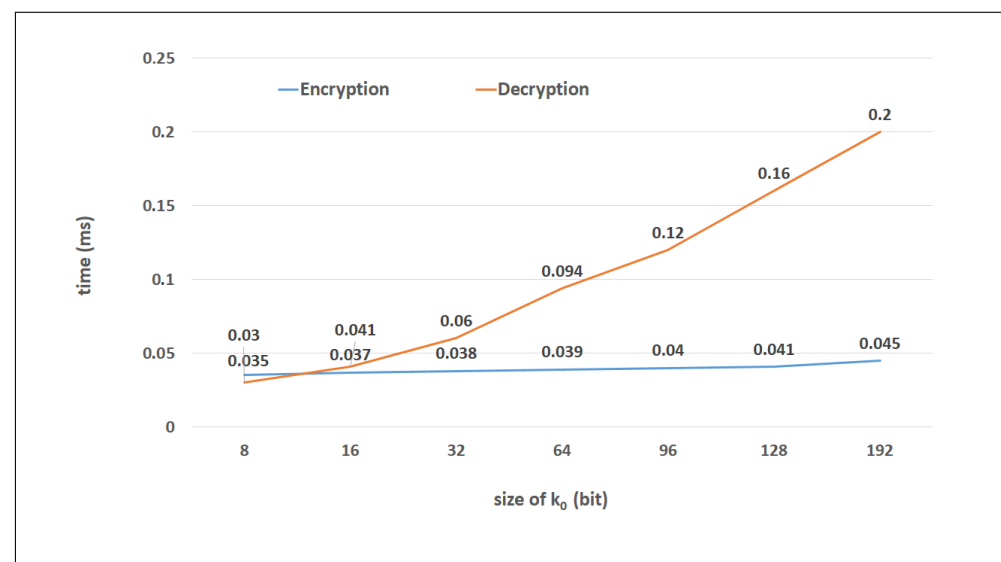


Figure 5. Encryption and decryption time relative to the increase in key size.

6.1. Comparative Study

In this section, we conduct a comparative analysis of our crypto-system against relevant state-of-the-art systems. The aim is to show the major gaps filled compared with others. The listed crypto-systems in Table 2 cover a range of homomorphic encryption schemes for different data types and with various efficiency and security levels. The introduced work in [24,32] proposed fully HE schemes based on symmetric keying, which imposes certain restrictions on their use. The analysis confirms the schemes' efficiency and practicality for adoption in various cloud computation applications. However, compared with the

proposed crypto-system in this paper, the restriction of symmetric key sharing and the encryption complexity proves the limitations of [24,32] and prevents their uses in the context of IoT scenarios. The proposed somewhat homomorphic encryption scheme over integers in [33] is another kind of asymmetric homomorphic encryption. The scheme encryption principle focuses on the Eurocrypt van Dijk et al. crypto-system [19]. Gentry's techniques are used in the contribution of [33] to easily convert the somewhat scheme into a practical, fully homomorphic encryption scheme available in cloud computing. The analysis results show the robustness and the dynamicity of the proposed scheme. Compared with the proposed crypto-system in this paper, the authors of [33] focused on reducing the key size of van Dijk to mitigate the computation complexity. However, Gentry's encryption techniques are still considered the most complicated encryption techniques and are not feasible to be implemented in many domains.

The work in [34] presents a new variant of the DGHVs integer-based somewhat homomorphic encryption scheme, including an efficient public key generation method. The authors claim that the proposed scheme has significantly lower complexities in various algorithms involved in the encryption process, making it more practical for real-world applications. The security is based on the two-element Partial Approximate Greatest Common Divisors (PAGCD) problem. The experimental results demonstrate that the proposed scheme is more efficient than any other integer-based SHE scheme currently available, making it a practical solution for homomorphic encryption. The authors in [35] extended the DGHV fully homomorphic encryption (FHE) scheme over integers by enabling batch FHE. This allows a vector of plaintext bits to be encrypted and homomorphically processed as a single ciphertext. The scheme is semantically secure because it relies on the approximate-GCD problem without errors. The paper demonstrates how arbitrary permutations can be performed on the plaintext vector using the ciphertext and public key. The scheme shows competitive performance and implements a fully homomorphic evaluation of AES encryption; the results are promising compared to the timings presented by Gentry et al. at Crypto 2012 for implementing a Ring-LWE-based FHE scheme. However, the PAGCD problem can be computationally intensive, making it challenging to efficiently implement homomorphic encryption schemes based on this problem. The generated noises in the encrypted data are considered significant. These concerns prevent the implementation of such encryptions in low-end connected devices.

In general, the proposed crypto-system is suitable for IoT environments. The small key and encrypted data sizes, the multiple keying, the execution speed, and the robustness are considered. The shown time execution in Table 3 confirms the highlight of our proposed encryption and decryption schemes. The strongest factor of the proposed encryption scheme is the constant progression of the encryption time in the function of data size, as illustrated in Figure 5. This could be particularly exploited in IoT–Cloud communications, where the connected low-end devices efficiently encrypt the data before sending it to the cloud. The cloud plays its role by storing and manipulating encrypted data.

To demonstrate the complexity of encryption and decryption algorithms, as represented by $O(\lambda)$, Figure 6 illustrates six tests with plaintext sizes equal to 8, 16, 32, 64, 96, and 128 bits respectively. Curve Figure 6A presents the rate $plaintextsize/encryptiontime$, the curve Figure 6B presents the rate $plaintextsize/decryptiontime$; we note that the percentage increase is constant and it expresses $O(\lambda)$.

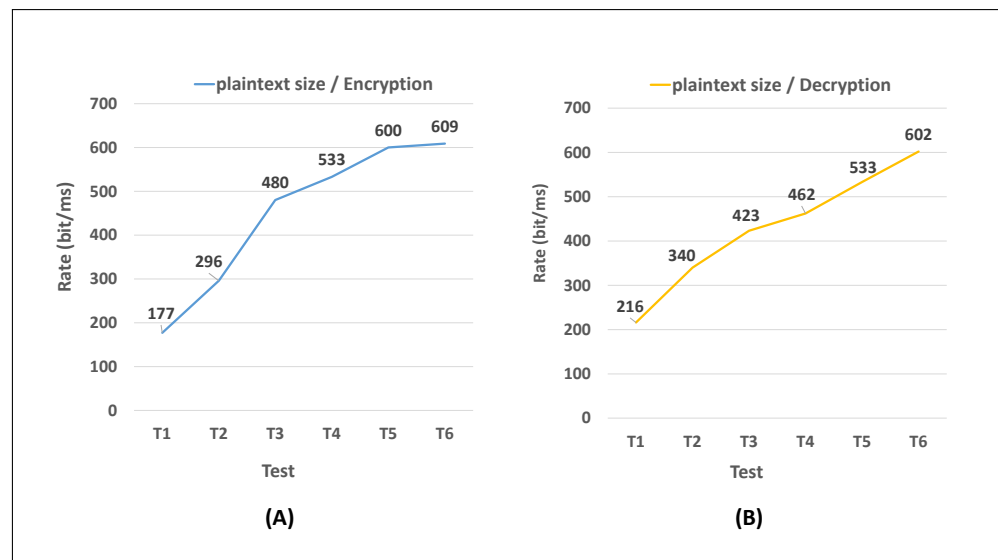


Figure 6. Rate of plaintext size to encryption and decryption time.

Figure 7 shows the ratio of plaintext size to ciphertext size in Figure 7A, as it is clear in Figure 7B that they are close, especially as the plaintext size increases, which shows how effective the proposed scheme is in using a small ciphertext size.

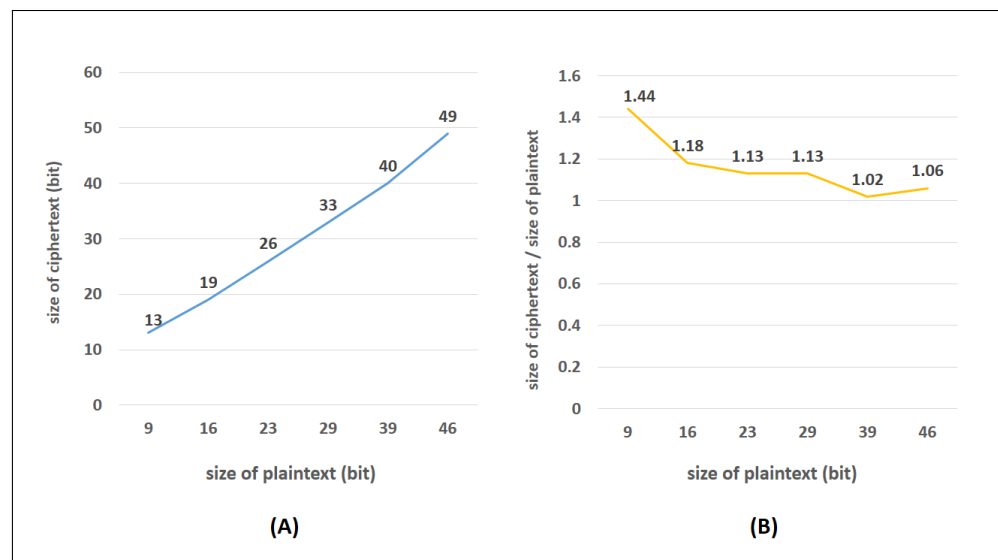


Figure 7. Rate of plaintext size to ciphertext size in the symmetric version that preserves order.

6.2. A Comparative Study against Attacks

Brute Force Attack is an attack where an attacker tries all possible combinations of passwords or keys to gain unauthorized access to encrypted data. To counter this attack, various homomorphic encryption techniques have been developed. This comparative study will compare brute force attacks against these techniques, including our proposed work. One of the key advantages of our proposed work is that it provides an asymmetric multi-key with a partially homomorphic encryption scheme. This means that each decimal digit of the given integer number is encrypted separately using a special key, providing an additional layer of security against brute force attacks. On the other hand, homomorphic encryption techniques rely on only a single key to encrypt and decrypt the data and are therefore vulnerable and prone to brute-force attacks.

Regarding data integrity, our proposed work also addresses the issue of untrusted third parties performing homomorphic operations on encrypted data, ensuring that the

integrity of the data is maintained. This is important because it helps prevent attacks like tampering or data alteration. Data alteration attacks involve manipulating encrypted data without knowing the secret key, leading to incorrect decryption results. The attacker can modify the ciphertext to produce a different plaintext upon decryption, causing data loss, unauthorized access, or other malicious outcomes. The proposed work offers an asymmetric multi-key with partially HE scheme that encrypts each decimal digit of the given integer number separately using a special key. This approach offers a higher degree of security against data alteration attacks, as an attacker must modify each digit of the ciphertext separately.

Overall, the proposed work offers significant improvements compared with other homomorphic encryption techniques regarding security against brute force attacks, providing data integrity, and computational efficiency.

7. Conclusions

This paper presents an improved partially homomorphic encryption asymmetric scheme for IoT devices based on number factorization and polynomial reconstruction problems. The proposed multi-key encryption scheme fragments decimal numbers into digits and multiplies each digit by a small secret key. The scheme has an order-preserving capability, making it suitable for IoT-constrained devices, enabling the owner to query ciphertexts in a specified range. The size of the generated ciphertext is almost equal to plaintext, and order-preserving is ensured using a symmetrical encryption version. The paper also addresses security and privacy concerns of cloud-based IoT solutions, addressing known issues such as encrypted data size, slow operations at the hardware level, and high computing costs at the provider level.

Further real experiments are needed, and alternative approaches could be explored to address the limitations of the current scheme. Future work could also focus on integrating the proposed encryption scheme with blockchain technology; a secure and distributed system that could enhance the overall security and privacy of IoT systems.

Author Contributions: Conceptualization, A.L.; Methodology, S.M.; Validation, M.H.; Formal analysis, A.E.; Investigation, M.K.; Supervision, T.B. and M.A.-K. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Alsoubi, T.; Hammoudeh, M.; Bandar, Z.; Nisbet, A. An overview and classification of approaches to information extraction in wireless sensor networks. In Proceedings of the 5th International Conference on Sensor Technologies and Applications (SENSORCOMM'11), Saint Laurent du Var, France, 21–27 August 2011; p. 255.
2. Kara, M.; Laouid, A.; Bounceur, A.; Lalem, F.; AlShaikh, M.; Kebache, R.; Sayah, Z. A novel delegated proof of work consensus protocol. In Proceedings of the 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), El Oued, Algeria, 20–21 November 2021; pp. 1–7.
3. Kara, M.; Laouid, A.; Hammoudeh, M. An Efficient Multi-Signature Scheme for Blockchain. Cryptology ePrint Archive, Paper 2023/078. 2023. Available online: <https://eprint.iacr.org/2023/078> (accessed on 3 April 2023).
4. Chait, K.; Laouid, A.; Laouamer, L.; Kara, M. A Multi-Key Based Lightweight Additive Homomorphic Encryption Scheme. In Proceedings of the 2021 International Conference on Artificial Intelligence for Cyber Security Systems and Privacy (AI-CSP), El Oued, Algeria, 20–21 November 2021; pp. 1–6.
5. Ghafir, I.; Prenosil, V.; Hammoudeh, M.; Han, L.; Raza, U. Malicious ssl certificate detection: A step towards advanced persistent threat defence. In Proceedings of the International Conference on Future Networks and Distributed Systems, New York, NY, USA, 19–20 July 2017. [CrossRef]
6. Sniatala, P.; Iyengar, S.; Ramani, S.K. Homomorphic Encryption. In *Evolution of Smart Sensing Ecosystems with Tamper Evident Security*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 69–76. [CrossRef]
7. Rivest, R.; Shamir, A.; Adleman, L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems. *Commun. ACM* **1978**, *21*, 120–126. [CrossRef]

8. Elgamal, T. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Trans. Inf. Theory* **1985**, *31*, 469–472. [\[CrossRef\]](#)
9. Naccache, D.; Stern, J. A New Public Key Cryptosystem Based on Higher Residues. In Proceedings of the 5th ACM Conference on Computer and Communications Security (CCS '98), New York, NY, USA, 2–5 November 1998; pp. 59–66. [\[CrossRef\]](#)
10. Paillier, P. Public-key cryptosystems based on composite degree residuosity classes. In Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques, Prague, Czech Republic, 2–6 May 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 223–238. [\[CrossRef\]](#)
11. Galbraith, S.D. Elliptic curve Paillier schemes. *J. Cryptol.* **2002**, *15*, 129–138. [\[CrossRef\]](#)
12. Kawachi, A.; Tanaka, K.; Xagawa, K. Multi-bit cryptosystems based on lattice problems. In Proceedings of the International Workshop on Public Key Cryptography, Beijing, China, 16–20 April 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 315–329. [\[CrossRef\]](#)
13. Brakerski, Z.; Gentry, C.; Vaikuntanathan, V. (Leveled) fully homomorphic encryption without bootstrapping. *ACM Trans. Comput. Theory TOCT* **2014**, *6*, 1–36. [\[CrossRef\]](#)
14. Brakerski, Z. Fundamentals of Fully Homomorphic Encryption—A Survey. *Proc. Electron. Colloq. Comput. Complex.* **2018**, *25*, 125.
15. Smart, N.P.; Vercauteren, F. Fully homomorphic encryption with relatively small key and ciphertext sizes. In Proceedings of the International Workshop on Public Key Cryptography, Paris, France, 26–28 May 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 420–443. [\[CrossRef\]](#)
16. Boneh, D.; Goh, E.J.; Nissim, K. Evaluating 2-DNF Formulas on Ciphertexts. In *Proceedings of the Theory of Cryptography*; Kilian, J., Ed.; Springer: Berlin/Heidelberg, Germany, 2005; pp. 325–341. [\[CrossRef\]](#)
17. Ishai, Y.; Paskin, A. Evaluating branching programs on encrypted data. In *Proceedings of the Theory of Cryptography Conference*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 575–594. [\[CrossRef\]](#)
18. Gentry, C. *A Fully Homomorphic Encryption Scheme*; Stanford University ProQuest Dissertations Publishing: Stanford, CA, USA, 2009.
19. Van Dijk, M.; Gentry, C.; Halevi, S.; Vaikuntanathan, V. Fully homomorphic encryption over the integers. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, France, 30 May–3 June 2010; Springer: Berlin/Heidelberg, Germany, 2010; pp. 24–43.
20. Brakerski, Z.; Vaikuntanathan, V. Fully homomorphic encryption from ring-LWE and security for key dependent messages. In Proceedings of the Annual Cryptology Conference, Santa Barbara, CA, USA, 14–18 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 505–524.
21. López-Alt, A.; Tromer, E.; Vaikuntanathan, V. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Proceedings of the Forty-Fourth Annual ACM Symposium on Theory of Computing, New York, NY, USA, 19–22 May 2012; pp. 1219–1234.
22. Pang, H.; Wang, B. Privacy-preserving association rule mining using homomorphic encryption in a multikey environment. *IEEE Syst. J.* **2020**, *15*, 3131–3141. [\[CrossRef\]](#)
23. Doröz, Y.; Shahverdi, A.; Eisenbarth, T.; Sunar, B. Toward Practical Homomorphic Evaluation of Block Ciphers Using Prince. In *International Conference on Financial Cryptography and Data Security*; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8438, pp. 208–220.
24. Biksham, V.; Vasumathi, D. A lightweight fully homomorphic encryption scheme for cloud security. *Int. J. Inf. Comput. Secur.* **2020**, *13*, 357–371. [\[CrossRef\]](#)
25. Yang, Y.; Zhang, S.; Yang, J.; Li, J.; Li, Z. Targeted fully homomorphic encryption based on a double decryption algorithm for polynomials. *Tsinghua Sci. Technol.* **2014**, *19*, 478–485. [\[CrossRef\]](#)
26. Yagoub, M.A.; Abdelkader, L.; Kazar, O.; Bounceur, A.; Euler, R.; AlShaikh, M. An adaptive and efficient fully homomorphic encryption technique. In Proceedings of the 2nd International Conference on Future Networks and Distributed Systems, New York, NY, USA, 26–27 June 2018; pp. 1–6. [\[CrossRef\]](#)
27. Ding, Y.; Li, X.; Lü, H.; Li, X. A novel fully homomorphic encryption scheme based on LWE. *Wuhan Univ. J. Nat. Sci.* **2016**, *21*, 84–92. [\[CrossRef\]](#)
28. Brakerski, Z.; Vaikuntanathan, V. Efficient fully homomorphic encryption from (standard) LWE. *Siam J. Comput.* **2014**, *43*, 831–871. [\[CrossRef\]](#)
29. Zhou, T.; Zhang, Z.; Chen, L.; Che, X.; Liu, W.; Yang, X. Multi-key Fully Homomorphic Encryption Scheme with Compact Ciphertext. Cryptology ePrint Archive, Paper 2021/1131. 2021. Available online: <https://eprint.iacr.org/2021/1131> (accessed on 3 April 2023).
30. Kara, M.; Laouid, A.; Hammoudeh, M.; Bounceur, A. One Digit Checksum for Data Integrity Verification of Cloud-executed Homomorphic Encryption Operations. Cryptology ePrint Archive, Paper 2023/231. 2023. Available online: <https://eprint.iacr.org/2023/231> (accessed on 3 April 2023).
31. Gidney, C.; Ekerå, M. How to factor 2048 bit RSA integers in 8 hours using 20 million noisy qubits. *Quantum* **2021**, *5*, 433. [\[CrossRef\]](#)
32. Gai, K.; Qiu, M.; Li, Y.; Liu, X.Y. Advanced fully homomorphic encryption scheme over real numbers. In Proceedings of the 2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud), New York, NY, USA, 26–28 June 2017; pp. 64–69.

33. Yang, H.M.; Xia, Q.; Wang, X.F.; Tang, D.H. A new somewhat homomorphic encryption scheme over integers. In Proceedings of the 2012 International Conference on Computer Distributed Control and Intelligent Environmental Monitoring, Zhangjiajie, China, 5–6 March 2012; pp. 61–64.
34. Ramaiah, Y.G.; Kumari, G.V. Towards practical homomorphic encryption with efficient public key generation. *Int. J. Netw. Secur.* **2012**, *3*, 10.
35. Cheon, J.H.; Coron, J.S.; Kim, J.; Lee, M.S.; Lepoint, T.; Tibouchi, M.; Yun, A. Batch fully homomorphic encryption over the integers. In Proceedings of the Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, 26–30 May 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 315–335.
36. Kara, M.; Laouid, A.; Euler, R.; Yagoub, M.A.; Bounceur, A.; Hammoudeh, M.; Medileh, S. A Homomorphic Digit Fragmentation Encryption Scheme Based on the Polynomial Reconstruction Problem. In Proceedings of the 4th International Conference on Future Networks and Distributed Systems (ICFNDS), New York, NY, USA, 26–27 November 2020; pp. 1–6.
37. Thangavel, M.; Varalakshmi, P. Enhanced DNA and ElGamal cryptosystem for secure data storage and retrieval in cloud. *Clust. Comput.* **2018**, *21*, 1411–1437. [[CrossRef](#)]
38. Coron, J.S.; Mandal, A.; Naccache, D.; Tibouchi, M. Fully Homomorphic Encryption over the Integers with Shorter Public Keys. In Proceedings of the Advances in Cryptology—CRYPTO 2011, Santa Barbara, CA, USA, 14–18 August 2011; Rogaway, P., Ed.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 487–504. [[CrossRef](#)]
39. Dasgupta, S.; Pal, S. Design of a Polynomial Ring based Symmetric Homomorphic Encryption Scheme. *Perspect. Sci.* **2016**, *8*, 692–695. [[CrossRef](#)]
40. Boer, D.; Kramer, S. Secure Sum Outperforms Homomorphic Encryption in (Current) Collaborative Deep Learning. *arXiv* **2020**, arXiv:2006.02894.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.