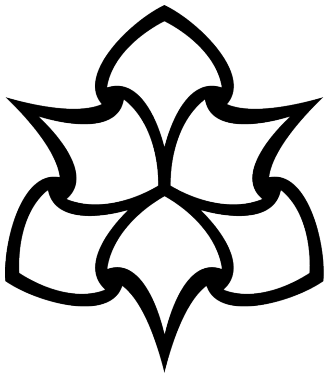

Privacy-preserving using homomorphic encryption in Mobile IoT systems

Wang Ren, Xin Tong, Jing Du, Na Wang, Shan Cang Li, Geyong Min, Zhiwei Zhao, and Ali Kashif Bashir

Abstract—The data privacy concerns are increasingly affecting the Internet of things (IoT) and artificial intelligence (AI) applications, in which it is very challenging to protect the privacy of the underlying data. In recent, the advancements in the performances of homomorphic encryption (HE) make it possible to help protect sensitive and personal data in IoT applications using homomorphic encryption based schemes. This paper proposed a practical homomorphic encryption scheme that can enable data users in IoT systems to securely operate data over encrypted data, which can effectively protect the privacy of key data in the system. The experimental results demonstrated the effectiveness proposed scheme.

Index Terms—Mobile Internet of Things, Data privacy, homomorphic encryption, cloud computing, artificial intelligence

1 Introduction

The increasing use of number of smart devices are connected by the Internet of Things (IoT), which makes data security and privacy concerns are rising. In IoT, data privacy is a one of key security concerns in major IoT business verticals, such as industrial IoT (IIoT), healthcare, retail, financial industrial, smart city, *etc.* [**?**]. In the past years, increasing number of data breaches (such as health information, *etc.*) in IoT has been reported [**?**], [**?**] that cyber criminals will steal an estimated 146 billion records by 2023 [**?**]. The security regulations, such as General Data Protection Regulation (GDPR) [**?**], Health Insurance Portability and Accountability Act (HIPAA) [**?**], and state-specific regulations, have emerged and evolved to address these growing security concerns [**?**]. Due to the diversity of devices and application, the IoT systems are becoming primary targets of cyberattacks. It is clear that security regulations, GDPR, Data Protection Act (DPA), *etc.*, have been committed to protect personal data and identification. It is key way to avoid data breaches before making disclosure because the risk and consequences of identification.

An IoT system may generates/collects sensitive data, such as personal data, patients' privacy data in healthcare, businesses data, *etc.*, which is usually transmitted and stored on cloud server(s). In many existing solutions, data is encrypted before transmitting to a cloud server. However, they often fail to deal with complicated attacks at the time of data conversion into cipher and after the cipher transmission [**?**]. Cyberattacks like DDoS attacks, insider, ransomware, fraud scam, *etc.*, can always cause data breaches. In many cloud-based IoT systems, data processing, storage, management, and data anlytics are more and more shifted to the cloud, in which the security and privacy increasingly relies on the third party cloud providers. However, it is noted that the third party cloud service providers (CSPs), such as Amazon Web Service (AWS), Microsoft Azure, *etc.*, may have major security flaws [**?**], [**?**], [**?**]: (1) Many data breaches were caused by misconfiguration of customers. Meanwhile, according to McAfee 99% misconfigurations cannot be detected [**?**]; (2) Most public CSPs cannot provide sufficient trustworthy, specifically for public sections, such as healthcare service, Goverment, *etc.*; (3) In-cloud data protection, mainly focus on the data is stored "in the cloud", data should not flow to unauthorised parties, including cloud-insiders as well as cloud users [**?**], [**?**]. Even the encryption cannot ensure protection against the provider leaking data, due to misconfiguration, bugs, malicious insider, *etc.* [**?**].

The homomorphic encryption (HE) is a promising way that can enable computation to be performed over encrypted data without retrieving the plaintext. The advancements in HE have make it partially practicable. The fully homomorphic encryption (FHE) can provide privacy protection by fully supporting homomorphic operations over encrypted data [**?**], [**?**]. The HE based privacy solutions can keep undecrypt ciphertext while still conducting operations over encrypted data to protect the privacy of the underlying data [**?**]. Previously, the expensive computational cost of HE is a big challenge in practice, but in the last few years, there the advances in the performance make it is possible to implement practical homomorphic-cryptosystems [**?**]. The HE based solution in IoT that do not necessarily trust each other for the CSPs. In IoT, the HE addresses the biggest problems, namely, access to large, diverse datasets from different data owners. In HE-enabled IoT systems can securely bring together data owners (DOs) and data users (DUs) who need data. And very importantly, the HE can implement this in a way that preserves security and privacy of the data, DOs, DUs, as well as CSPs.

The main contributions in this work are summarised as follows:

- *Wang Ren is with College of Electronics and Information Engineering, Sichuan University, Chengdu 610065, China; he is also with the China Information Technology Security Evaluation Center, Beijing, China.*
- *X. Tong, J. Du, and N. Wang are with China Information Technology Security Evaluation Center, Beijing, China.*
- *Prof. Li is with University of the West of England and Prof. Zhao is with the School of Computer Science and Engineering at the University of Electronic Science and Technology of China, Chengdu, China.*
- *Prof Min is with University of Exeter, Exeter EX4 4PY, UK.*
- *Dr Bashir is with the Manchester Metropolitan University, Manchester M15 6BH, UK.*

1) We proposed a practical homomorphic encryption based privacy-preserving scheme that can make DUs securely operate data over third-party cloud server without leakage any information;
2) A enhanced protocol is proposed to improve the security and privacy of the cloud-based IoT systems;
3) Experimental results shows the effectiveness and performance of proposed scheme and protocol.

The remain of this paper is organised as: Section 2 reviews the most recent works in homomorphic encryption based privacy-preserving; Section 3 propose the HE-based data anonymous solution. Section 4 provides detailed algorithms using HE and Section 5 concludes this paper.

2 Related works

The homomorphic property of homomorphic encryption schemes allows to carry out certain operations over the encrypted data and provide encrypted results. After decryption, the same results can be obtained with operations was performed in plaintext. For two message m_1 and m_2, Eq.(1) shows an HE scheme that supports any operation

$$\begin{aligned} c_1 = \mathsf{Enc}(\mathsf{pk}, m_1), \;\; c_2 = \mathsf{Enc}(\mathsf{pk}, m_2) \\ m_1 \oplus m_2 = \mathsf{Dec}(\mathsf{sk}, c_1 \oplus c_2) \end{aligned} \tag{1}$$

in which $\mathsf{Enc}(\cdot)$ and $\mathsf{Dec}(\cdot)$ are the encryption and decryption algorithms; c_1 and c_2 are the ciphers of m_1 and m_2, respectively; and pk is the public key, sk is the private key.

As mentioned above, HE shows great promises in privacy-preservation in the untrust cloud computing environment due to the resource available. Specifically, in the increasing use of artificial intelligence and machine learning, to securely building models by utilising massive data, HE can help maintaining security and access to large, diverse datasets. The HE based systems can be categorised into three groups: full homomorphic encryption (FHE) systems, partially homomorphic encryption system (PHE), and somewhat homomorphic encryption (SHE) systems [**?**].

2.1 Full Homomorphic Encryption (FHE)

The FHE allows a DU to perform arbitrarily complex programs over encrypted data without the need to know the secret key [**?**], [**?**], [**?**], [**?**]. FHE based schemes have been extensively studied since it is proposed by Gentry [**?**]. However, the efficiency needs to be greatly improved to make it practicable in IoT environment. In the past few years, many research efforts have been conducted from the view point of mathematical, focusing on the addition and multiplication operations. In [**?**], a SHE solution was proposed for cloud systems that can guarantee the security and privacy of data by providing homomorphic encryption and complex mathematical operations. In summary, the FHE schemes includes Gentry's scheme and its optimisation, Bootstrapping based FHE scheme, FHE scheme without Bootstrapping, *etc*. In the past few years, the FHE based solutions have been widely used in many applications, including

- E-voting systems [**?**], mainly use FHE to ensure to maintain the privacy, accuracy, verifiability, fairness the votes casted and the authentication before participants casts their votes.
- Privacy information retrieval (PIR) protocol, it allows an application (DU) to retrieve records from a database without disclosing any information about the DO that could be used to determine which records were selected. Many research efforts have been done on improve the speed of database lookups without increasing the server-side storage requirements [**?**].
- Private set intersection (PSI), it allows two parties to independently compute the intersection of their sets without revealing anything private items except the intersection itself [**?**].

2.2 Partially Homomorphic Encryption (PHE)

Unlike conventional FHE-based solutions, the PHE based schemes are computationally practical but come with the cost of supporting only limited mathematical operations over encrypted data [**?**]. A PHE solution mainly contains two operations: additive homomorpic encryption schemes and multiplicative homomorphic encryption schemes. Murthy *et al.* propose a cloud based data privacy preserving solution based on partial homomorphic encryption that permits performing mathmatical operations over encrypted data while not compromising the ciphertext in [**?**]. In PHE, Paillier cryptosystems support addition and ElGamal cryptosystems support multiplication, order-preserving encryption supports comparison, and deterministic encryption, respectively. In recent, the PHE based solutions have been used in machine learning and artificial intelligence [**?**] in which PHE shows great potentials in protecting privacy of the source data.

2.3 Somewhat Homomorphic Encryption (SHE)

Somewhat homomorphic encryption (or SWHE) can evaluate two types of gates, but only for a subset of operations. In [**?**], Dijk *et al.* extended Gentry's lattice-based idea that has similar properties could improve the efficiency and homomorphic properties. Smart and Vercauteren improved Gentry's FHE scheme by enabling public key and private key by giving smaller size of ciphertext and shorter key than Gentry's original scheme [**?**]. In the past few years, more efficient SHE solutions have been proposed, such as the BGV11 [**?**], LTV12 [**?**], BLLN13 [**?**], CKKS16 [**?**], [**?**], *etc.* The security of BGV11 and CKKS is based on the ring learning with error (RLWE), while for LTV and BLLN schemes the security relies on Number Theory Research Unit (NTRU) problems, which is based on the shortest vector problem in a lattice. Gentry *et al.* proposed a new scheme GSW for building FHE by avoiding an expensive "relinearization" in multiplication in 2013, which was further improved as FHEW and TFHE.

SHE schemes are widely used in practical systems: in [**?**], Busom *et al.* proposed a privacy mechanism by homomorphically aggregating the data for multiple data users to guarantee the privacy of data aggregation while depending on an additive homomorphic scheme. This work is based on the ElGammal cryptosystem, in which each n member-group has a public key pk that encrypts the readings and private key sk is the same for each member. It requires all members are honest and same parameters are used

for each member. The SHE schemes are also commonly used in image/water marking systems. In [**?**], a reversible information hiding solution was developed using SHE.

The HE can well provide end-to-end privacy. In cloud-based systems, HE allows performing powerful data analytic (such as AI and ML algorithms) in the cloud server. This work will focus on untrust data owners and data users in cloud-based IoT systems.

3 Proposed Scheme

As shown in Fig.1, in this work we consider a simple cloud-based IoT system with following entities participating in the data encryption system: the IoT devices, such as smart sensors, medical devices, *etc.*, are able to generate and collect data and they technically own the data; The cloud system provides data storage, computing services, *etc.*; and the applications, such as big data anlaytics, machine learning and artificial intelligence (AI) algorithms, need the data and act as data consumers in this system, namely, data users.

- Data owner (DO), the device or application that owns the raw data, processes it, and stores it in the cloud server;
- Cloud server (CS), cloud-based third party server(s) which only store encrypted data from DO, and waiting for the query from a user;
- Data user (DU), application or devices that consumers the data, it has access to the CS and have been authorised to decrypt encrypted data by DO.

The DOs might be IoT devices with limited resource, in terms of battery, storage, and computational resources, which can communicate each other or with IoT gateway/cloud server using wireless technologies, such as Low Power WiFi, Bluetooth Low Engergy (BLE), Low Power Wide Area Network (LPWAN), *et al.* In application level, protocols such as CoAP, MQTT, *etc.*, are widely used. In this work, the DU can send queries to CS and retrieve some specific encrypted data. It can also ask CS perform homomorphic operations, like *addition, multiplication, etc.* and CS is expected to learn nothing from the encrypted data.

As discussed above, the HE is a data encryption method that allows encrypted data can be processed and manipulated, which make it possible that the third party to perform algebraic functions over encrypted data without needing to reveal the real values of the data. This feature can be use to protect data privacy in zero trust IoT environment. In a HE crypto-systems, HE can enable the third party to work with and use the the encrypted data without having access to or knowing the contents of the decrypted data.

In a cloud-based IoT system, if a smart sensor (DU) wants to store it generated/collected data on a cloud server, which need to be analysed by third party application (DU), the basic procedures are:

1) DU encrypts data, $\mathsf{c} = \mathsf{Enc}(\mathsf{m})$ and then transmit the ciphertext c to the cloud server (CS);
2) The third party application (DU) submits query $Q(f)$ to CS;
3) The CS performs $Q(f)$ over c and sends back the result out to DU;
4) DU decrypts $\mathsf{m}' = \mathsf{Dec}(\mathsf{out})$.

in which the f is a data *manipulate function* provided by the data user. Currently, there two solutions for HE: (1) FHE based solution, which means the above solution supports arbitrary function, however it is very computational expensive and cannot be affordable in practice; (2) SHE based solution, which supports some support functions, like simple addition, multiply, query, *etc.*

To evaluate a HE solution, semantic security means that the ciphertext does not reveal any information contained in plaintext.

$$m_0, m_1,\ \mathsf{Encrypt}\ (\mathsf{pk}, m_0) \approx \mathsf{Encrypt}\ (\mathsf{pk}, m_1) \tag{2}$$

In an IoT system, the FHE scheme can prevent malicious attacks on the third-party cloud server from gaining access to the encrypted data. However, when a DU needs to alter or update a specific record item, it often needs to transfer the encrypted updates to the server or decrypts cipher then alter and then re-encrypted in the trusted cloud. However, this needs the entire system is in a trusted environment.

In this work, we propose a privacy-preserving protocol for cloud-based IoT systems utilising the Paillier cryptosystem, which is a probabilistic asymmetric algorithm for public-key cryptosystem that provides following features:

- Probabilistic encryption, for a given $a \in \mathcal{Z}$, there exist N encryption of a, as $[a]_{\mathsf{pk}} \in \mathsf{Enc}(a) \subset \mathcal{Z}_{N^2}$, in which $\mathsf{Enc}(a)$ is the set of all encryption of a with $|\mathsf{Enc}(a)| = N$.
- Addition, the Paillier cryptosystem allows to add two encrypted values, as

$$[a]_{\mathrm{pk}} \oplus [b]_{\mathrm{pk}} \in \mathsf{Enc}(a+b) \text{ for any } a, b \in \mathcal{Z}_N \tag{3}$$

- Multiplication, the Paillier system allows the multiplication of an encrypted value with a plaintext one, as

$$a \otimes [b]_{\mathrm{pk}} \in \mathcal{E}nc(ab) \text{ for any } a, b \in \mathcal{Z}_N \tag{4}$$

In a PHE cryptosystem [**?**], if the RSA public key has modulus n and encryption exponent e, then the encryption of a message m is given by $\mathsf{Enc}(m) = m^e$, the homomorphic property can be described by

$$\begin{aligned} \mathsf{Enc}(m_1) \cdot \mathsf{Enc}(m_2) &= m_1^e \cdot m_2^e \ \ mod\ \ n \\ &= (m_1 \cdot m_2)^e \ mod\ n \\ &= \mathsf{Enc}(m_1 \cdot m_2) \end{aligned} \tag{5}$$

3.1 PHE based solution vs FHE

The FHE supports arbitrary computation on ciphertext, which can enable the construction of programs for any desirable function running over ciphertext. In recent, a number of libraries have been implemented for cloud systems, include IBM HElib, Microsoft SEAL, PALISADE, *etc.* In cloud-based IoT systems, the FHE is quite inefficient and resource restrained IoT devices might be unable to afford the computational costs of FHE. Therefore, FHE is not a doable solution for IoT systems.

In recent, a number of FHE-based solutions have been developed, which can provide secure and privacy-preserving operations over data even in the untrust environment. However, most FHE-based solutions cannot be utilised over resource constrained devices. The PHE-based

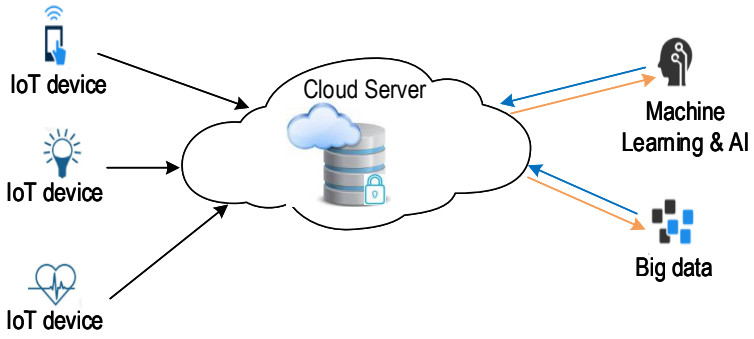

Fig. 1. Cloud-based IoT system (The blue arrows denote enquiries from DU, and orange arrows denote HE results.)

solution is feasible for computationally lightweight applications in IoT. Figure 1 shows an example of cloud-based IoT system, in which both the IoT devices (DOs) and applications (DUs) can perform lightweight data encryption and computationally heavy HE can be performed over untrusted third-party cloud server.

3.2 PHE based Protocol

The PHE has been proved to be effective in many applications, like electronic voting, medical data analysis, gene analysis, etc. This work focus on the application of PHE in IoT data privacy protection. Figure 2 shows an example of a PHE-enabled IoT system, which includes n DOs $m_1, m_2, \ldots, m_n$ and one or more DUs. The blue arrows in Figure 2 denote the data queries from DUs(such as ML or AL applications, big data applications, *etc.*), and the orange lines denote the encrypted results.

In a typical IoT system, the data owner, such as smart sensors, mobile, devices, *et al.*, collects/generates data and then transmit to a cloud server, and data users, such as researchers, applications, *et al.* can access these data. The problem here is three folds: (1) to guarantee the secret of the data, the DO will encrypt data before transmitting to cloud server, and a DU can send queries to cloud server for asking specific data; (2) for the server, it should learn nothing from both DO and DU; (3) for DU, it can only access the data authorised by DO which is defined in the queries. To address these questions, we propose following protocols.

In IoT systems, DOs are continuously generating or collecting data. In this work, each DO divides the original data with size of n_1 into non-overlapping data blocks with size of n_b.

3.3 Procedure

In this works, we use single google cloud server (S) to store the data created and encrypted by sensors DOs, applications can retrieved data from the cloud server S. The procedures includes three stages:

(1) *Initialisation*. Crypto module at the DO and DU independently generate the key pairs (pk, sk) for the Paillier crypto HE system.

(2) *Extract, Transformation and Loading (ETL)*. In this stage, each DO uses its pk is used to encrypt the generated data, which then be uploaded to the cloud server. If specific aggregation model is needed, the data needs to be transformed to match the requirements of the aggregation model. Encryption using the shared key ensures that as long as an adversary does not obtain the sk of both DO and DU, the confidentiality of the data is preserved. By doing this, we can avoid a single point of failure in the system.

(3) *Data Query*. A DU can ask to send a query to the S by specifying a specific operations (this work supports addition and integer multiplication). The server can conduct the paillier homomorphic encryption using the shared keypairs depends the query. Once done, the server will return the encrypted result to DU that can be decrypted using its secret key.

As shown in above scenario in Fig.1, the server S can conduct the Paillier homomorphic encryption but cannot retrieve the query result. In some case, the S can then homomorphically add random Laplacian noise to obfuscate the query result and achieve differential privacy.

Table 1 shows an example that data created by sensors stored in a google cloud server (in which only part of encrypted data are presented).

TABLE 1
Encrypted data stored on a google cloud server

Sensor ID	readings	cipher
1	0x02	0x3fd2. . .a876
2	0x04	0x386e. . .3f7b
3	0x06	0x722c. . .d887
4	0x08	0x4aad. . .5e7c
. . .	. . .	. . .

3.4 PHE-based Privacy-preserving scheme for IoT

Inspired by BGH13 [**?**], this subsection will detail the design and implementation of PHE based IoT privacy-preserving solution that allows a DU to query a remote database held by a cloud server, with the guarantees that the server does not learn anything from the query.

In a cloud-based IoT system, a DU cannot retrieve more information than what it queries for, in which the

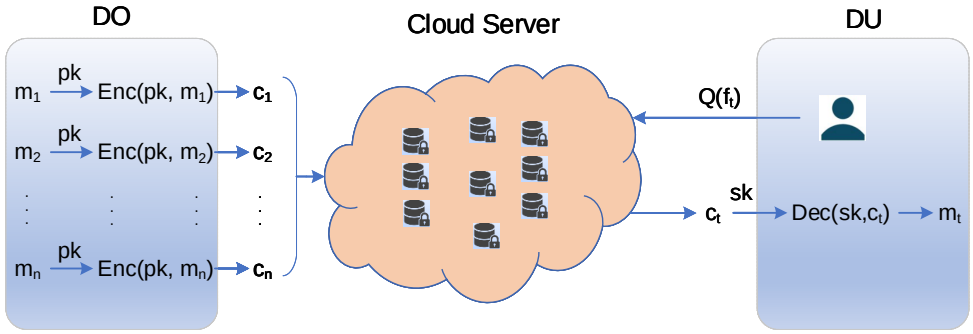

Fig. 2. Homomorphic encryption decryption time in cloud-based IoT systems

cloud server S holds a database with records (all encrypted) $c_1, c_2, \ldots, c_n$ and a DU DU_i may want to learn $c_i, i \in \{1 \ldots n\}$ but another DU DU_j may want to retrieve $c_j i \in \{1 \ldots n\}$. Actually, with proposed solution, a DU may retrieve the result of homomorphic operations of encrypted records, as

$$c_r = c_i \oplus c_j,\ i \in \{1 \ldots n\} \tag{6}$$

or

$$c_r = I_{mpz} \oplus c_i,\ i \in \{1 \ldots n\} \tag{7}$$

in which I_{mpz} is an arbitrary magnitude integer. In this work, we present a protocol that allows a DU to retrieve a record from server without revealing which record has been queried. The DU learns only the record corresponds to its query. Actually, a DU can submit multiple queries to learn more than one record.

For simplicity, we consider an IoT system with three key actors: one DO, one cloud Server (S), and one DU. The S holds database and each record is composed of $\{uID, feature\ \#1, \ldots feature\ \#u\}$, each *feature* corresponds to a specific feature.

1) The system first generate parameters, including prime numbers p, q, $n = pq$, and security parameter $\lambda = lcm(p-1, q-1)$, which can be used to generate $(\mathsf{pk}, \mathsf{sk})$ for each DO, S, and DU; For a trusted DU, DO(s) will share key information like sk, *search token*, *encryption key* via secure channel.
2) The DO generates a inverted table for the data generated, in which each record associates to all the features. Then, DO encrypts the inverted table using its sk, each *feature* is converted into a search token using a pseudo-random function, e.g.,

$$\mathcal{P}_{k_o}(\text{"}uID = 0xf3ad\text{"}), \mathsf{Enc}_{k_o}(\mathcal{I})) \tag{8}$$

in which $\mathcal{I}$ is the set of indices in the database of the individuals for which the uI is equal to $0xf3ad$, encrypted with a key DO encryption key k_o only known to DU;
3) Records query: *Simple query, homomorphic operation query*; For simple query, a DU can check whether there is a match for its query in the inverted database using the search token Θ. If yes, DO sends relevant information to the trusted party DU a $(\mathcal{Q}, \Theta)$; For *homomorphic operation query*, the server will first verify whether there are related match(s) and then conduct required homomorphic operations (addition, multiplication);
4) DU decrypts the results received from the S.

Steps 1) and 2) will be performed at the *Initialisation* stage when the database needs to be updated. After the setup, DU can interact with other S and DU to obtain an answer to a certain query. Step 3) is for simple query and homomorphic operation query. In practice the data is batched, and each DO may split as many data values as available slots in the ciphtertext. Protocol 3.4 shows the details.

4 Experimental Results and Analysis

In this work, we tested the performance of proposed scheme in a real cloud-based environment. To evaluate the performance of proposed scheme, we excluded the time consumed over communication channel since it significantly relies on the network traffic. The experimental setting only focuses on the performance tests of time used for *encryption*, *homomorphic operations*, and *decryption*. Both the DO and DU instances were running over a Mac OS X on Intel i7 2.6GHz machine with 16GB ram. Google API services were used to act as cloud server, and the system is implemented using Python 2.7.

In this cloud-based IoT scenario, the DO generates data and then encrypts to a remote cloud service, which can simply run homomorphic operations, *addition and integer multiplication*. A DU is able to retrieve the data from cloud server and by send a query to the server. Once receives the result, the DU can decrypt the result.

We first tested the performance of data encryption and decryption without homomorphic operations as shown in Figure 3 encryption with 128-bit keysize. In this test, 26 sensors instances create data, and each sensor then using *paillier-encrypt*$(\cdot)$ to encrypt the data, and cipertext were sent to the remote cloud server S using *Drive Activity API* [**?**]. To test the data query, we use a DU to send a *query* $Q(f, m_t)$, in which f is the homomorphic operation, and x_t is the token for a specific data x which is shared from DO in a secure channel. In this test, we set f as 0 that means no operation needs but only retrieve the encrypted data. The the DU can use the shared keys to decrypt the ciphertexts. It can be seen

Protocol 1 Homomorphic encryption based privacy preserving for IoT system

System Initialisation. Each DO creates keypair $K_D^i(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(1^\lambda)$, and each DU creates keypair $K_U^i(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Keygen}(1^\lambda)$, in which λ is security parameter. All keys exchange will be made on secure communication channel.

Goal. Parties jointly compute a random shuffle of their inputs.

The protocol:

1) **DO.**
   a) A **DO** generates and encrypts a message m using $\mathsf{c} = \mathsf{Enc}(\mathsf{pk}, m)$;
   b) transmits c to the **CS** in a communication channel;
2) **Server.**
   a) S waits the $queries$ from DU
   b) Once receives a query $Q(f, m_{token})$, take homomorphic operations as $\mathsf{Enc}(m) = \mathsf{Enc}(m_1) + \mathsf{Enc}(m_2)$;
   c) perform f, for addition $\mathsf{c} = \mathsf{Add}(\mathsf{pk}, c_1, \cdots, c_t)$; and for multiplication, $\mathsf{c} = \mathsf{Mul}(\mathsf{pk}, c, a)$; then return the result out to **DU**;
   d) return m;
3) **DU**
   a) build a query $Q(f, m_{token})$, in which f could be addition or multiplication;
   b) send $Q(f, m_{token})$ to server and wait for response;
   c) once receives c_{new}, in which $\mathsf{c_x}$ is the HE encryption of the input data x
   d) decrypts the result using $m' = \mathsf{Dec}(\mathsf{sk}, out)$.

   return;

return;

from Figure 3 that the time consumed by different sensors are very similar and the average time consume is about $0.3s$, and the time consumed for decryption is pretty low and average time for decryption is $0.0017s$. Actually, the operations *write, read* cost the main time, average $0.4s$ and $0.5s$, respectively, which significantly relies on the design of database. Table 4 shows the performance of HE data privacy encryption with key size of 256 bits. The average times for *Enc* at DO, *Write, Read* at S, and *Dec* at DU, are $0.4307, 0.6211, 0.4084$, and 0.0016 respectively. Compare the *Enc, Write, Read*, the *Dec* spends only about 3.2%, which means it is possible that even resource-constrained IoT devices are able to support data query as DU.

Table 4 shows the performance of HE data privacy encryption with key size of 128 bits. The average times for *Enc* at DO, *Write, Read* at S, and *Dec* at DU, are $0.0745, 0.4257, 0.3716$, and 0.0004 respectively. It can be seen the performance is slightly better than the case with key size of 256 bits.

TABLE 2
Cloud system HE data privacy encryption (key size 256 bits)

Sensor ID	Enc(s)	Write(s)	Read(s)	Dec(s)
1	0.3956	0.6391	0.7290	0.0017
2	0.3951	0.5001	0.2544	0.0018
3	0.4179	0.6490	0.5029	0.0017
4	0.3898	0.6532	0.5997	0.0018
5	0.4626	0.8488	0.3774	0.0015
6	0.4147	1.5889	0.5129	0.0018
7	0.4161	0.8151	0.4076	0.0018
8	0.4082	0.5887	0.3010	0.0018
9	0.3857	0.3151	0.4318	0.0017
10	0.4057	0.3360	0.4927	0.0018
11	0.3825	0.6218	0.2240	0.0010
12	0.4279	0.8381	0.3038	0.0011
13	0.4075	0.8569	0.3083	0.0017
14	0.3961	0.5515	0.2249	0.0018
15	0.3798	0.5851	0.6957	0.0018
16	0.3952	0.5677	0.7749	0.0014
17	0.3562	0.3115	0.3173	0.0018
18	0.3988	0.5414	0.2536	0.0018
19	0.3873	0.3375	0.3050	0.0014
20	0.3997	0.3963	0.2962	0.0018
21	0.4333	0.8013	0.3926	0.0015
22	0.3923	0.4738	0.5388	0.0014
23	0.3784	0.7143	0.4357	0.0017
23	0.3864	0.6644	0.3229	0.0018
25	0.4361	0.6470	0.3746	0.0018
26	0.4463	0.3072	0.2415	0.0017
Average:	0.4037	0.6211	0.4084	0.0016

TABLE 3
Cloud system HE data privacy encryption (key size 128 bits)

Sensor ID	Enc(s)	Write(s)	Read(s)	Dec(s)
1	0.0659	0.4023	0.6231	0.0003
2	0.0729	0.3081	0.2828	0.0004
3	0.0712	0.2267	0.3449	0.0005
4	0.0725	0.3792	0.2071	0.0004
5	0.0685	0.4576	0.3224	0.0004
6	0.0732	0.5304	0.3465	0.0004
7	0.0736	0.7114	0.5157	0.0002
8	0.0683	0.6830	0.2451	0.0002
9	0.0772	0.5163	0.3079	0.0004
10	0.0804	0.3295	0.3051	0.0004
11	0.0745	0.5615	0.3015	0.0002
12	0.0749	0.3822	0.6154	0.0002
13	0.0768	0.2322	0.3070	0.0002
14	0.0748	0.2358	0.3092	0.0003
15	0.0728	0.2254	0.2336	0.0004
16	0.0730	0.3391	0.2869	0.0002
17	0.0702	0.3085	0.9293	0.0002
18	0.0749	0.3075	0.2406	0.0002
19	0.0717	0.5159	0.5504	0.0004
20	0.0728	0.3112	0.2159	0.0004
21	0.0741	0.3898	0.3650	0.0004
22	0.0728	0.3132	0.2171	0.0003
23	0.0751	0.3325	0.3064	0.0003
23	0.0725	0.3208	0.2437	0.0002
25	0.0735	0.5362	0.4091	0.0003
26	0.0735	0.3229	0.2210	0.0002
Average:	0.0745	0.4527	0.3716	0.0004

To test the homomorphic encryption operations, we evaluated the performance of a HE systems, in which 26 sensors encrypt and upload data to cloud server (google cloud server); the data user sends query of sum of two data and then download the ciphertext. This can make data user conduct add operations over encrypted data without leak any information. In this work, the cloud server will conduct the addition operation of ciphertext, and the DU only download the $\mathsf{Enc}(x_1, x_2)$ and then decrypt the sum. with $\mathsf{Dec}(\mathsf{Enc}(x_1, x_2))$. Table 4 shows the performance only with *homomorphic addition* (with key size of 256 bits), the average time for *queries (addition)* and *decryption* are $0.3821s$ and $0.0016s$, respectively.

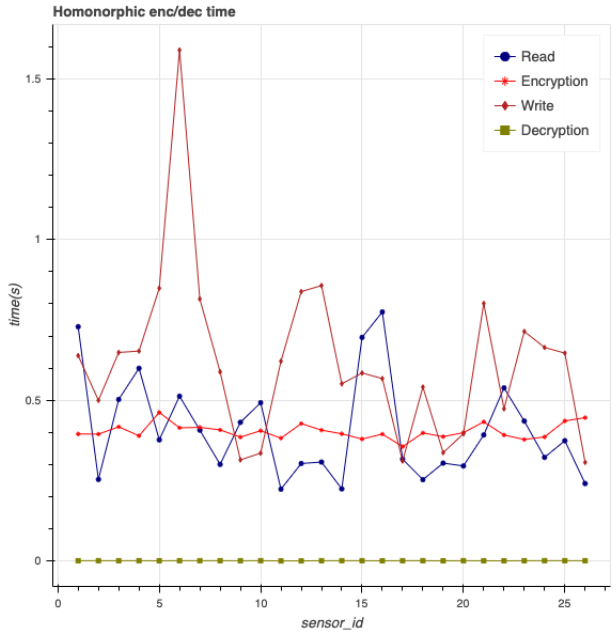

Fig. 3. Performance of data encryption/decryption without HE operations in cloud-based IoT systems (key size = 256)

TABLE 4
DU conduct HE addition over encrypted data (key size 256 bits)

Sensor ID	Request(s)	Dec(s)
1	0.3826	0.0016
2	0.3473	0.0017
3	0.2818	0.0017
4	0.3897	0.0017
5	0.4089	0.0013
6	0.3028	0.0017
7	0.4069	0.0017
8	0.3192	0.0016
9	0.4978	0.0016
10	0.2924	0.0016
11	0.3198	0.0019
12	0.5277	0.0017
13	0.3982	0.0017
14	0.4018	0.0013
15	0.3096	0.0017
16	0.4194	0.0017
17	0.3971	0.0017
18	0.4742	0.0017
Average:	0.3821	0.0016

To further evaluate the performance of *homomorphic addition, homomorphic multiplication*, we tested 18 DUs querying

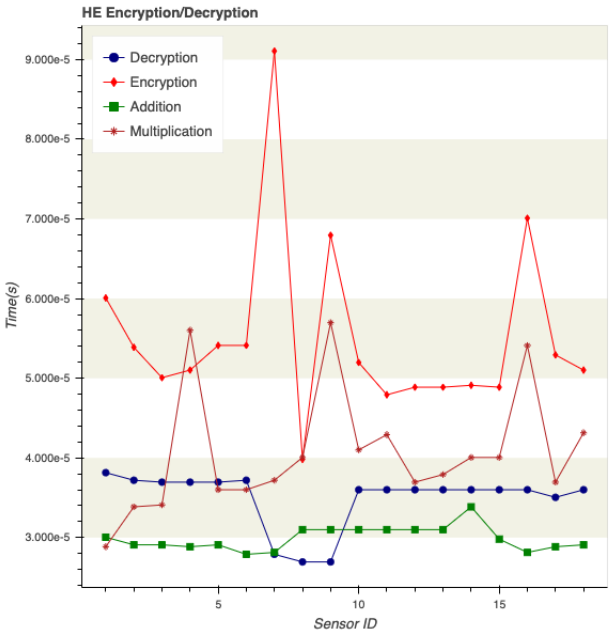

Fig. 4. Homomorphic encryption decryption performance in cloud-based IoT systems (key size = 128)

with both *homomorphic addition and multiplication* from the server, Table 4 shows the performances, in which the average $request$ time is about $0.3623s$, the average time for a single *homomorphic addition* is 0.002954, and for a single *homomorphic multiplication* is 0.007908.

TABLE 5
DU conduct HE addition and multiplication over encrypted data (key size 128 bits)

Sensor ID	Request(s)	Add-Dec(ms)	Mul-Dec(ms)
1	0.3287	0.003099	0.009060
2	0.3079	0.003099	0.008821
3	0.5045	0.002861	0.007868
4	0.3209	0.002861	0.007868
5	0.3553	0.002861	0.007868
6	0.3435	0.001907	0.006914
7	0.3266	0.003099	0.008106
8	0.4200	0.002146	0.006914
9	0.3317	0.002861	0.006914
10	0.3058	0.002861	0.007868
11	0.6458	0.003099	0.008106
12	0.2412	0.003099	0.007868
13	0.3520	0.004053	0.008106
14	0.5780	0.002861	0.008106
15	0.3230	0.003099	0.007868
16	0.2349	0.003099	0.008106
17	0.2556	0.003099	0.008106
18	0.3463	0.003099	0.007868
Average:	0.3623	0.002954	0.007908

5 Conclusion and Discussion

The emerging HE can effectively protect data privacy in IoT environments. This paper propose a privacy preserving solution to help protect the privacy of data owner, cloud server, and data user in complicated IoT environments. The experimental results demonstrated the effectiveness of proposed solution. Actually, this work can be further extended

to privacy preserving for machine learning applications, which are increasingly applied in new IoT apps, products and services.

REFERENCES

[1] L. D. Xu, W. He, and S. Li, "Internet of things in industries: A survey," *IEEE Transactions on Industrial Informatics*, vol. 10, no. 4, pp. 2233–2243, 2014.

[2] H. Tao, M. Z. A. Bhuiyan, A. N. Abdalla, M. M. Hassan, J. M. Zain, and T. Hayajneh, "Secured data collection with hardware-based ciphers for iot-based healthcare," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 410–420, 2019.

[3] P. Huang, L. Guo, M. Li, and Y. Fang, "Practical privacy-preserving ecg-based authentication for iot-based healthcare," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 9200–9210, 2019.

[4] S. Li, K. R. Choo, Q. Sun, W. J. Buchanan, and J. Cao, "Iot forensics: Amazon echo as a use case," *IEEE Internet of Things Journal*, vol. 6, no. 4, pp. 6487–6497, 2019.

[5] "Ieee 3d body processing - industry connections (3dbp ic): Communication, security, and privacy," *IEEE 3D Body Processing - Industry Connections (3DBP IC): Communication, Security, and Privacy*.

[6] S. Yulianto, C. Lim, and B. Soewito, "Information security maturity model: A best practice driven approach to pci dss compliance," in *2016 IEEE Region 10 Symposium (TENSYMP)*, 2016, pp. 65–70.

[7] J. H. Cheon, D. Kim, Y. Kim, and Y. Song, "Ensemble method for privacy-preserving logistic regression based on homomorphic encryption," *IEEE Access*, vol. 6, pp. 46 938–46 948, 2018.

[8] J. Wang, H. Qi, Y. He, W. Li, K. Li, and X. Zhou, "Flowtracer: An effective flow trajectory detection solution based on probabilistic packet tagging in sdn-enabled networks," *IEEE Transactions on Network and Service Management*, vol. 16, no. 4, pp. 1884–1898, 2019.

[9] Z. Yao, J. Ge, Y. Wu, and L. Jian, "A privacy preserved and credible network protocol," *Journal of Parallel and Distributed Computing*, vol. 132, pp. 150–159, 2019.

[10] Y. Ma, Y. Wu, J. Li, and J. Ge, "Apcn: A scalable architecture for balancing accountability and privacy in large-scale content-based networks," *Information Sciences*, vol. 527, pp. 511–532, 2020.

[11] J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty security considerations for cloud-supported internet of things," *IEEE Internet of Things Journal*, vol. 3, no. 3, pp. 269–284, 2016.

[12] R. Zhou, X. Zhang, X. Wang, G. Yang, H. Wang, and Y. Wu, "Privacy-preserving data search with fine-grained dynamic search right management in fog-assisted internet of things," *Information Sciences*, vol. 491, pp. 251–264, 2019.

[13] Z. Brakerski, C. Gentry, and S. Halevi, "Packed ciphertexts in lwe-based homomorphic encryption," in *International Workshop on Public Key Cryptography*. Springer, 2013, pp. 1–13.

[14] W. Gao, W. Yu, F. Liang, W. G. Hatcher, and C. Lu, "Privacy-preserving auction for big data trading using homomorphic encryption," *IEEE Transactions on Network Science and Engineering*, pp. 1–1, 2018.

[15] Y. Shoukry, K. Gatsis, A. Alanwar, G. J. Pappas, S. A. Seshia, M. Srivastava, and P. Tabuada, "Privacy-aware quadratic optimization using partially homomorphic encryption," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 5053–5058.

[16] S. Ravindram and P. Kalpana, "Data storage security using partially homomorphic encryption in a cloud," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 4, 2013.

[17] B. Alaya, L. Laouamer, and N. Msilini, "Homomorphic encryption systems statement: Trends and challenges," *Computer Science Review*, vol. 36, p. 100235, 2020.

[18] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the forty-first annual ACM symposium on Theory of computing*, 2009, pp. 169–178.

[19] M. Van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2010, pp. 24–43.

[20] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *International Workshop on Public Key Cryptography*. Springer, 2010, pp. 420–443.

[21] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," *ACM Transactions on Computation Theory (TOCT)*, vol. 6, no. 3, pp. 1–36, 2014.

[22] M. M. Potey, C. Dhote, and D. H. Sharma, "Homomorphic encryption for security of cloud data," *Procedia Computer Science*, vol. 79, pp. 175 – 181, 2016, proceedings of International Conference on Communication, Computing and Virtualization (ICCCV) 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S187705091600154X

[23] R. Tekade, R. Kharat, V. Magade, M. Shaikh, and P. Mendhe, "E-voting system using visual cryptography & homomorphic encryption," *International Journal of Advanced Research in Computer and Communication Engineering (IJARCCE)*, vol. 1, 2016.

[24] H. Corrigan-Gibbs and D. Kogan, "Private information retrieval with sublinear online time," in *Advances in Cryptology – EUROCRYPT 2020*, A. Canteaut and Y. Ishai, Eds. Cham: Springer International Publishing, 2020, pp. 44–75.

[25] H. Chen, K. Laine, and P. Rindal, "Fast private set intersection from homomorphic encryption," in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017, pp. 1243–1255.

[26] S. Murthy and C. R. Kavitha, "Preserving data privacy in cloud using homomorphic encryption," in *2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)*, 2019, pp. 1131–1135.

[27] Z. Brakerski, "Fully homomorphic encryption without modulus switching from classical gapsvp," in *Advances in Cryptology – CRYPTO 2012*, R. Safavi-Naini and R. Canetti, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 868–886.

[28] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "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*, 2012, pp. 1219–1234.

[29] J. W. Bos, K. Lauter, J. Loftus, and M. Naehrig, "Improved security for a ring-based fully homomorphic encryption scheme," in *IMA International Conference on Cryptography and Coding*. Springer, 2013, pp. 45–64.

[30] J. H. Cheon, A. Kim, M. Kim, and Y. Song, "Homomorphic encryption for arithmetic of approximate numbers," in *International Conference on the Theory and Application of Cryptology and Information Security*. Springer, 2017, pp. 409–437.

[31] H. Chen, W. Dai, M. Kim, and Y. Song, "Efficient multi-key homomorphic encryption with packed ciphertexts with application to oblivious neural network inference," in *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*, 2019, pp. 395–412.

[32] N. Busom, R. Petrlic, F. Sebé, C. Sorge, and M. Valls, "Efficient smart metering based on homomorphic encryption," *Computer Communications*, vol. 82, pp. 95 – 101, 2016.

[33] L. Xiong and D. Dong, "Reversible data hiding in encrypted images with somewhat homomorphic encryption based on sorting block-level prediction-error expansion," *Journal of Information Security and Applications*, vol. 47, pp. 78 – 85, 2019.

[34] Wikipedia, "Homomorphic encryption," June 2020, https://en.wikipedia.org/wiki/Homomorphic_encryption#Fully_homomorphic_encryption.

[35] Google, "Drive activity api library," April 2020, https://developers.google.com/drive/activity/v2/reference/rest.