




A Design of Secure Communication Protocol Using RLWE-Based Homomorphic Encryption in IoT Convergence Cloud Environment

Byung-Wook Jin¹ · Jung-Oh Park²  · Hyung-Jin Mun³

Published online: 27 November 2018

© Springer Science+Business Media, LLC, part of Springer Nature 2018

Abstract

A super-connected society is emerging in which things and things or people and things communicate with each other through the Internet of Things. Since devices in the existing IoT environment have limitations such as low power, volume, and performance, a new paradigm has been suggested by integrating with cloud computing technology. However, there are still issues to resolve in the new convergence paradigm in order to mitigate vulnerabilities regarding data management and information protection for security. Thus, this study designs a RLWE-based homomorphic encryption communication protocol for user authentication and message management in a cloud computing-based IoT convergence environment. We conducted performance analysis on a communication protocols in the existing IoT environment and the proposed communication protocol to ensure safety and security. The study verified safety and security by conducting performance analysis of current IoT environment communication protocol and proposed communication protocol. The study conducted comparative analysis on time complexity and space complexity in accordance with encryption and decoding of proposed communication protocol to confirm that it provides strong security and equivalent level of efficiency. Also, by designing a communication protocol, the study aimed to provided a safe communication infrastructure from user authentication to data transfer to users.

Keywords Authentication · Cloud computing · Homomorphic encryption · User privacy protect

✉ Jung-Oh Park
jopark02@sungkyul.ac.kr

Byung-Wook Jin
wlsquddnr@koipa.re.kr

Hyung-Jin Mun
jinmun@gmail.com

¹ Department of Strategy and Project, Korea Intellectual Property Protection Agency, Anyang, South Korea

² Department of Paideia, Sungkyul University, Sungkyuldaehakro-53, Manan-gu, Anyang-City, Gyeonggi-Do 430-742, South Korea

³ Division of Information and Communication Engineering, Sungkyul University, Anyang, South Korea

1 Introduction

The Internet of Things (IoT) technology has been in the center of the 4th industrial revolution, providing highly efficient services to users. The spread of IoT devices has created various data quickly, generating big data. In order to collect, manage, and analyze the created data, convergence with the cloud computing technology is required [1]. Such integrated technology is becoming a new paradigm of ICT environment. Thus, the environment, where cloud computing technology and IoT are converged, can provide various services to users by improving the limitations of the existing IoT environment, with increased efficiency and availability compared to the existing way.

In the convergent IoT environment based on cloud computing, there is an increasing need for the safety of the user account management and access security systems and a solution to personal information infringement and leakage. Furthermore, with the coming of the hyper-connected network era, the possibility of social anxiety from user information leakage, monetary damage and life threat is increasing. In addition to the existing security threats of the IoT environment, new and altered attack techniques have appeared to require technical improvement and new encryption algorithms [2].

This paper will apply a completely homomorphic cryptogram algorithm, design a mutual authentication technique between a user and gateway in a cloud computing-based IoT convergence environment, and ensure safety to prevent the leakage of important user information and perform authentication. This study will also design a data verification technique by applying a completely homomorphic cryptogram algorithm in a gateway between the data management server and IoT environment in cloud computing.

The purpose of this paper is first to apply a fully homomorphic encryption algorithm in the convergent IoT environment based on cloud computing and to provide safety related to important user information leakage and certification process by designing a mutual certification technique between the user and gateway. Second, a data verification technique is designed using the fully homomorphic encryption algorithm between the data management server in the cloud computing environment and the gateway in the IoT environment. The data verification and management techniques have higher efficiency and safety compared to the conventional encryption algorithms. The designed data verification technique offers higher safety for data transfer and against information leakage. Lastly, the certificate generated by the proposed protocol maintains reliability of the signature value created in the certification process using the self-certification infra hash-tree technique, quickly responding to any infringements of integrity caused by alteration of the signature value. The hash-tree technique verifies the signature value to create the certificate and reduces the overhead compared to the existing methods when performing the verification process.

The proposed encryption protocol is designed for registration and key generation by exchanging the identification value and random number parameter between the user and gateway in the convergent IoT environment based on cloud computing.

The signature value and identification value are verified between the gateway and cloud computing server to design the data verification and management techniques. A self-certification hash-tree technique is designed to prevent infringements of integrity and maintain reliability of the signature value generated in the registration process and message management process. Lastly, a communication protocol is designed to provide a safe communication infrastructure for the user, from user verification to data transfer.

The composition of this paper is as follows. Section 2 analyzes the convergent IoT environment based on cloud computing, the public key encryption algorithm and its

vulnerabilities, and the certificate management technique. Section 3 describes the design for the fully homomorphic communication protocol proposed in the IoT environment based on cloud computing. This includes user registration, session key generation, message management technique and message communication protocol. Section 4 verifies safety of the proposed encryption protocol and comparatively analyzes its efficiency and security. Section 5 draws a conclusion.

2 Related Work

2.1 Analysis on Issues of IoT Convergence Paradigm and Security Requirements

Based on the advantages of cloud computing, cloud computing-based IoT convergence environment overcomes the limits of IoT and provides the maximization of resource use, data processing on-demand service, and mutual operability [3]. It has been studied as it can solve issues such as user's reliability and security in a cloud environment. However, there are still weaknesses and assignments to solve although the aforementioned advantages are provided.

2.1.1 Device

IoT devices have various operation systems ranging from low power to high specifications. Thus, the provision of appropriate resources for each device in an environment converged with cloud computing technology must be researched. IoT devices must have their functions optimized, and they must also be managed so that the remaining resources can join virtualization management to efficiently use other equipment. If remaining resources are not used during virtualization management, the resource virtualization service must operate effectively so that the resources are managed stably. In addition, as an IoT-based device can join a cloud computing domain, overhead can occur in terms of additional data and operational aspect. Accordingly, effective device management using high-efficient algorithm is required in contrast to intelligent calculation and low power performance [4, 5].

2.1.2 Data

Various data are created in an IoT-based environment. Such transferred and collected data are analyzed and processed in real-time in a cloud service environment, and must be managed separately through various service providers. Research needs to be conducted on a solution that manages data safely and effectively because there is none yet [6].

2.1.3 Service

For an IoT device to participate in a cloud computing service, users must be paid and provide enough services to participating users. Service should include device performance, sensing and actuator functions, as well as communication with the data from the devices [4, 6].

2.1.4 Security

In a cloud computing-based IoT convergence environment, protecting data security and user’s personal information is an important issue. When a user receives service from a cloud computing domain, not only reliable security for data processing and storage but also data confidentiality, availability, integrity, and information protection must be guaranteed. Moreover, there can be vulnerabilities from the operation of IoT convergence environment and during the creation and processing of data. Thus, to ensure safe transfer and reliable management of data, safe accessibility and connectivity are essential [3, 4].

2.2 Algorithm of Completely Homomorphic Cryptogram

A homomorphic cryptogram is an encryption function that saves defined operation in a plain text and cryptogram space, and enables operation on plain texts by applying arithmetic operations + and \times to cryptogram. A homomorphic cryptogram is used to perform operation on a plain text by applying addition and multiplication, the basic arithmetic operations, on cryptogram. A cryptogram that differentiates from homomorphic encryption and stores random logical operations is called completely homomorphic cryptogram [5, 7].

Rivest announced the very first homomorphic cryptogram algorithm by modifying an RSA cryptogram algorithm, but it could not be applied due to safety issues. However, studies on homomorphic cryptogram algorithms different from the existing techniques have been carried out recently; in 2009 Gentry presented a complete cryptogram algorithm with proven safety, which mainly uses learning with error (LWE) [7] (Fig. 1).

LWE-based completely homomorphic encryption performs encryption process by using the elements in a vector space, and the cryptogram includes an Error value. When a homomorphic operation ($\text{Eval}_{pk}(\cdot, c, c^*)$) is performed for the multiplication of two cryptograms with an n -dimension, the encryption grows to n^2 and thus the dimension of cryptogram must be lowered through key change. Ring learning with error (RLWE) is an LWE-based completely homomorphic cryptogram algorithm on a ring, and consists of key setting, encryption decode, and key switch stages [4, 8].

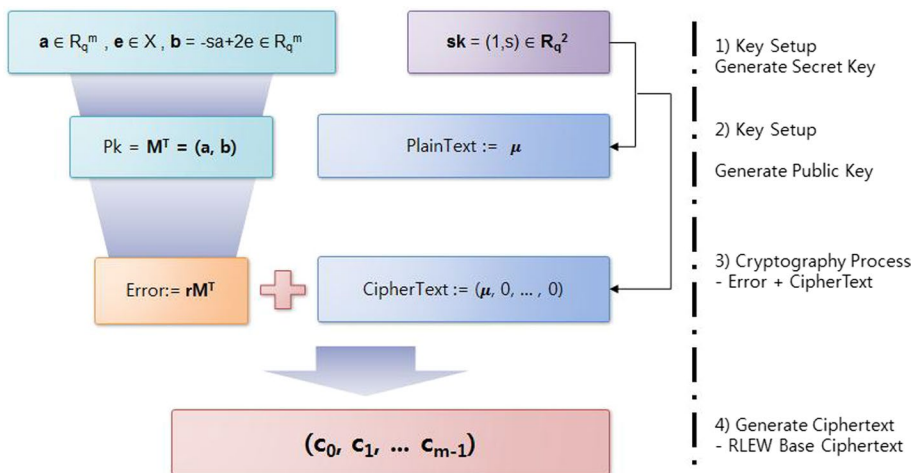


Fig. 1 RLWE Based fully homomorphic encryption algorithm process

2.3 Trend of Fully Homomorphic Encryption Algorithm in Cloud Environment

The latest homomorphic encryption algorithm is applied to the cloud computing environment. It uses the verifiable computation technique, searchable encryption technique and encryption data sharing technique.

- (1) **Verifiable computation:** This is a technique in which a device with relatively poor computing capability requests outsourcing of computation services that cannot be handled on its own. The outcome of outsourcing can be provided in an efficient way. The technique is comprised of three steps including the pre-treatment process, input preparation process, and output computation and verification process [4, 9].

First off, the pre-treatment process computes supplementary information that includes the public key and private key associated with the client function. When the client sends a request to the cloud server for computation, the input preparation process sends the public key, private key and supplementary information found in the pre-treatment process to the client. Lastly in the output computation and verification process, the cloud server computes π , which is the result of encrypting $f(x)$ using on public information associated with F and x . The client computes $f(x)$ and compares it with the result computed by the cloud server [10].

The verifiable computation technique is a non-interactive technique between users with verified identities. It has an advantage of protecting privacy by comparing and verifying the input and output values [2, 9].

- (2) **Searchable encryption:** This is an encryption technique developed to increase the efficiency of search by attaching an index to search specific information while guaranteeing safety of encrypted information, similar to the conventional encryption technologies. This technique is comprised of four steps including the key generation process, encryption process, trapdoor generation and search testing process.

In the key generation step, the user generates and stores a private key and discloses a public key and public information to other users. The encrypted message is created by encrypting data, and an index is also created to search data keyword information. The user generates a trapdoor for the keyword using the private key. Lastly, the receiver can use the trapdoor to find data sent by the sender [11, 12].

- (3) **Sharing encryption data:** In the cloud computing services, the user wants to share encrypted data with other users while protecting confidentiality and privacy. Since the existing services fail to provide reliability, a technique that can achieve such service is demanded. The encryption data sharing technique is a technique designed to share data using the re-encryption key generated from the private key of the data owner and the public key used by the receivers. This technique is comprised of four steps including the key generation process, encryption process, proxy server re-encryption process and decryption process [2, 9, 13].

First, the user generates and stores a private key and discloses a public key and public information to other users. The sender generates an encrypted message that can execute the decryption process without exposing the encrypted message. The encrypted message is sent

to the cloud server, along with a re-encryption key that is used to execute the re-encryption process on the cloud server. When decrypting the encrypted message generated by the sender, the receivers receive the re-encryption key to execute the decryption process and confirm the encrypted message. Data can be obtained later by decrypting with the private key.

3 Design of Proposed Cryptography Communication Protocol in IoT Convergence Cloud Environment

This chapter covers these: Mutual authentication between a user and gateway, Message management technique for saved data, and Proposed encryption communication protocol in a cloud computing-based IoT convergence environment.. The domain consists of Gateway, MS:Management Server, Cloud Computing Server, and Application. The table of abbreviations for user registration of the proposed environment, session key generation, data management technique and message communication protocol is as presented in Table 1.

3.1 User Registration and Session Key Creation

The parameters created during user and gateway registration are a secret key, public key, identification value, and a random signature value; the exchange parameters are user and gateway's identification value, signature value, and a key creation parameter. In addition, MS:SECURITY is shared by the user and gateway before they are registered. First, the user is verified with verification value (user ID, user random value) from MS:SECURITY before transmitting the encrypted user verification value from IoT based gateway. Then, gateway creates V_SK1 based on user verification value confirmation and monomorphic password, the verification value of gate way is sent to MS:SECURITY. MS:SECURITY verifies verification value of gateway and saves it. Through monomorphic password based ($E_{pk}(\cdot, c, c^*) =$) calculation of user's gateway parameter and gateway's parameter, V_SK2 is created. $\vec{c} \oplus \vec{c}^*$. The details of identification value parameter exchange and verification process in a cloud computing-based IoT environment are as shown in Fig. 2.

User registration and signature value request: Using an application, a user transfers a registration request message from MS: OSS, which requests the user's identification value and signature value. Creation of user identification value and signature value: The user creates $Nonce_USER$ ($Nonce \in R_q^m$) value in R_q^m and creates CA_1 and CA_2 using a homomorphic cryptogram algorithm.

$$CA_1 = E_{RLWE-PUB-G}(USER-SK \oplus ME:SECURITY) || E_{RLWE-PUB-G}(Nonce-USER) \quad (3-1)$$

$$CA_2 = E_{RLWE-PUB-S}(Nonce-USER \oplus USER-ID) || E_{RLWE-PUB-S}(Nonce-USER) \quad (3-2)$$

User identification value and verification request: The user transfers the cryptogram ($CA_1 || CA_2$) created from MS:OSS. After that, MS:OSS transfers the verification request message received from the user to MS:SECURITY and makes a verification request. Verification of identification value: MS:SECURITY detects CA_2 of the received message and decodes the RLWE-based completely homomorphic algorithm with the secret key of MS:SECURITY. After decoding, it performs XoR calculation on the extracted $Nonce_USER$ and compares, analyzes, and verifies the user's identification value. {If $Exist_USER_ID = USER_ID$ }.

Table 1 Abbreviations of the proposed communication protocol

Abbreviation	Description
Nonce_User	User generated random numbers in the application
User_SK	User's Secret_Key
SECURITY_SK	Secret_key of MS:SECURITY
V_SK1	Secret_Key created through fully homogeneous tensor product in gateway based on parameter values for user and gateway
V_SK2	Secret_Key created through fully homogeneous tensor product in application based on parameter values of user and gateway
GATEWAY_ID	Identifier of gateway
SIG_G	Signature value generated by gateway
SIG_User	User signature value created by application
GATEWAY_SK	Secret_Key of gateway
G_DATA	Data collected from IoT-based devices
CCS_ID	Cloud computing server identifier
SIG_CCS	Signature value generated by cloud computing server
SIG_C-S	Signature value generated based on parameters of users, gateways, and cloud servers with hashtrips
User_Info	User information
User_ID	User identification
ERLWE_PUB_O	Encryption is performed using the open key of the MS:OSS of the fully semi-active password-based
ERLWE_PUB_S	Encryption is performed using the open key of the MS:SECURITY of the fully semi-active password-based method
ERLWE_PUB_G	Cryptographically encrypts GATEWAY with the disclosure key of a fully semi-active password-based
ERLWE_PUB_A	Encryption is performed using the open key of a fully semi-synchronous password-based application
ERLWE_PUB_C	Encrypts cloud computing servers with fully homogeneous passkey-based encryption
Evalpk(+, c, c*)	Co-operation of fully semi-current cryptographic algorithms
Evalpk(., c, c*)	Multiplexing of fully semi-current cryptographic algorithms

User identification and verification response: MS:SECURITY transfers a user identification and verification response message to MS:OSS. The MS:OSS which received the response message transfers the cryptogram (CA₁) received from the user and requests a gateway identification and signature value. Confirmation of user secret key: Using the received message as a secret key, the gateway performs RLWE-based completely homomorphic algorithm decoding and XoR calculation with MS:SECURITY shared previously and acquires the user's secret key.

$$\text{Verify USER-SK} \oplus \text{ME:SECURITY} = \text{USER-SK} \quad (3-3)$$

After parameter verification, MS:OSS saves the signature value created between the gateway and user based on the parameter. After that, the user creates V_SK2 based on MS:SECURITY_SK of gateway and the user's USER_SK. The key creation and user registration process in a cloud computing-based IoT environment is as shown in Fig. 3.

V_SK key creation: The gateway calculates the confirmed user's secret key and the secret key of the gateway calculates Tensor Product to create V_SK1 key. After that,

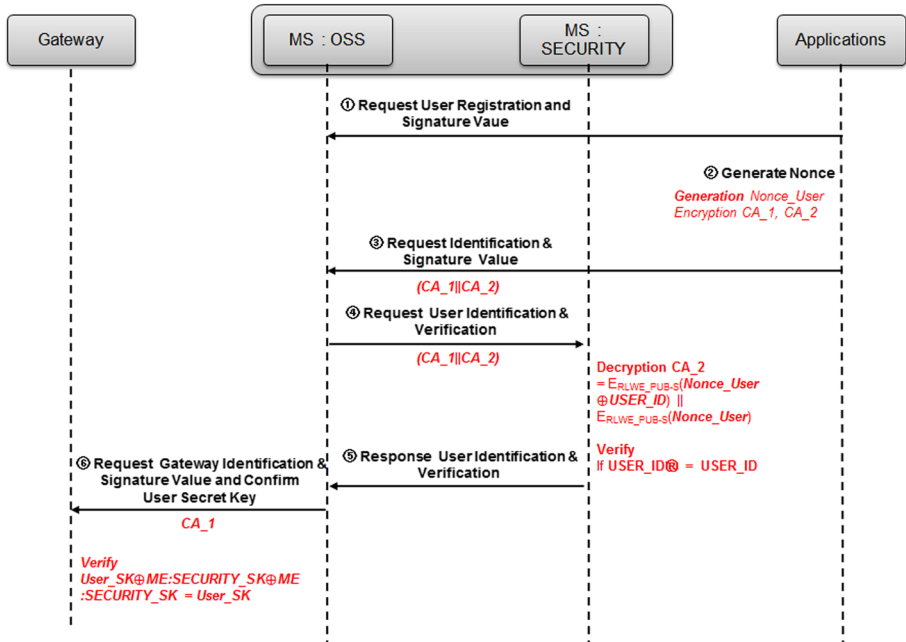


Fig. 2 Key setting and identification parameter exchange process between user and gateway in cloud centric IoT environment

XoR calculation is performed on the gateway’s identification value and the secret key of MS:SECURITY, and RLWE-based completely homomorphic algorithm encryption of MS:SECURITY is performed to create a cryptogram. Encryption is then performed on V_SK1 and creates CG_1 through connection. XoR calculation is performed on the gateway’s secret key and MS:SECURITY’s secret key, which are encrypted with the user’s open key to create CG_2. After that, the authentication information and index between the user and gateway are created and saved.

$$\text{User} \cdot \text{Gateway_Auth_Value} = CA_1' \otimes CG_2 \tag{3-4}$$

$$U \cdot G_Index = (\text{Gateway_ID} \oplus \text{USER_ID})$$

$$\text{Storing } \text{User} \cdot \text{Gateway_Auth_Value} || U \cdot G_Index$$

Gateway identification value response and signature value request: By attaching the cryptogram created in 7, the gateway transfers the response message to MS:OSS. After receiving the message, MS:OSS transfers the request message of gateway identification, verification, and signature value. Verification of identification value and the creation of signature value: MS:SECURITY decodes CG_1 received and acquires the gateway’s identification value and V_SK1. After that, it performs XoR calculation on the user and gateway’s identification value and user’s random value and then creates a signature value. It connects the created signature value with the identification value of gateway and encrypts them using the open key of MS:OSS to create CS_1.

$$\text{SIG_USER} = E_{\text{RLWE_PUB-S}}((\text{USER-ID}) \oplus \text{User-Nonce}) \tag{3-5}$$

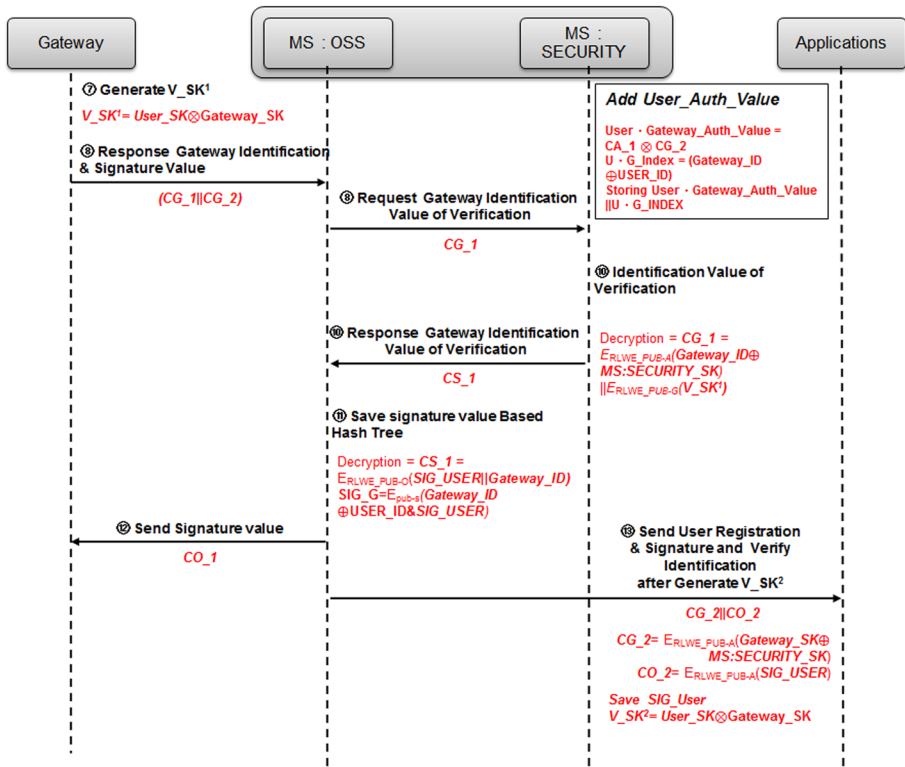


Fig. 3 Key generation and user registration process in cloud centric IoT environment

$$CS_1 = E_{RLWE_PUB_O}(SIG_USER) || Gateway-ID \tag{3-6}$$

Gateway identification value and signature value response: MS:SECURITY transfers the response message according to the request from MS:OSS by attaching the cryptogram created in 9. Saving signature value based on hash tree: MS:OSS decodes the received cryptogram CS_1 and confirms signature value (SIG_USER) and the gateway identification value. After that, it performs XOR calculation on the gateway identification value and user identification value, performs AND operation on the confirmed signature value, and creates SIG_G . MS:OSS creates CO_1 by encrypting with the gateway's open key and creates CO_2 by encrypting with the user's open key.

$$SIG_G = E_{PUB_S}(Gateway-ID \oplus USER-ID \oplus SIG_User) \tag{3-7}$$

$$CO_1 = E_{RLWE_PUB_G}(SIG_G) \tag{3-8}$$

$$CO_2 = E_{RLWE_PUB_A}(SIG_USER) \tag{3-9}$$

Transfer of signature value: The signature value of MS:OSS transfers the signature value created based on hash tree to the gateway by attaching the cryptogram created in 11. After that, MS:OSS attaches the cryptogram created in 9 and 11 and transfers the user registration completion and signature value to the user.

$$CO_2 = E_{PUB_S}(Gateway-ID \oplus USER-ID \oplus SIG_User) \tag{3-10}$$

Verification of identification value and the creation of V_SK2 : The user decodes the cryptogram received and confirms the secret key and signature value of gateway. After performing Tensor Product on the gateway’s secret key and user’s secret key, V_SK2 is created.

$$V_SK2 = USER-SK \otimes \text{Gateway-SK} \tag{3-11}$$

3.2 Design of Data Management Technique

This part will describe the data storage and management process in a cloud computing service domain. Gateway extracts generated signature value from user and gateway registration process from previous phase and data that is collected in real time from device and transfers it to CSS:DB. Then, CSS:DB transfers verification value of gateway from MS.SECURITY. MS:OSS verifies the value and saves the signature value based on hash tree. CSS:DB that is verified with verification value conducts monomorphic password($\text{Evalpk}(+, c, c^*) = c \oplus c^*$) with data received by gateway to manage data. Figure 4 shows the block diagram of message management technique.

Extraction of signature value: Gateway performs RLWE-based completely homomorphic encryption on the collected data with V_SK1 .

$$CG_1 = E_{RLWE-PUB-CSS}(\text{Gateway-ID}||\text{SIG-G}||\text{G-DATA}) \tag{3-12}$$

Collected data transfer: The gateway transfers the cryptogram created in I from a cloud computing server. The cloud computing server decodes the received cryptogram,

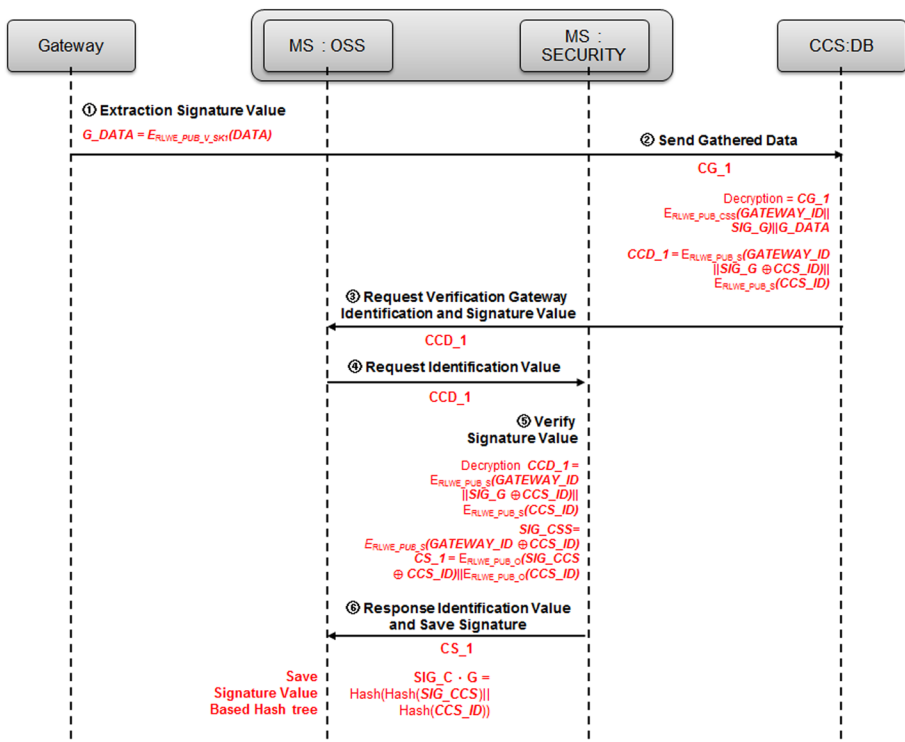


Fig. 4 A design of message management method in cloud centric IoT environment

confirms the linked Gateway ID and signature value (SIG_G), and encodes them using MS:SECURITY’s open key and creates CCD_1.

$$CCD_1 = E_{RLWE-PUB-S}(Gateway-IDSIG-G) \oplus CCS-ID || E_{RLWE-PUB-S}(CCS-ID) \tag{3-13}$$

Request Gateway Identification value and verification Process: In the cloud computing server, the MS:OSS generates a message containing the identification and signature value verification request by attaching the password generated by the MS:OSS. MS:OSS transmits the identification verification value request from MS: SECURITY SECURITY by EDC_1. Verification of signature value: MS:SECURITY decodes the received message and confirms the identification value of cloud computing server, identification value of gateway, and the signature value (SIG_G). After, it performs XoR calculation on the identification value of cloud computing server and gateway, encodes them with an open key, and creates a signature value (SIG_CSS) between the cloud server and gateway.

$$SIG_CSS = E_{RLWE-PUB-S}(Gateway-ID \oplus CCS-ID) \tag{3-14}$$

$$CS_1 = E_{RLWE-PUB-O}(SIG-CSS \oplus CCS-ID) || E_{RLWE-PUB-O}(CCS-ID) \tag{3-15}$$

Identification value verification response and signature value saving: MS:SECURITY transfers the response message according to the request by attaching the cryptogram created in 4. MS:OSS, which receives the message, decodes the received cryptogram and confirms the cloud server’s identification value and signature value (SIG_CSS). Using hash tree base, it creates a signature value (SIG_C·G) as well as CC_1 and CC_2 cryptograms using RLWE-based cryptogram algorithm.

$$CC_1 = E_{RLWE-PUB-C}(SIG \cdot G) || (CCS-ID \oplus Gateway-ID) \tag{3-16}$$

$$CC_2 = E_{RLWE-PUB-G}(Gateway-ID) || E_{RLWE-PUB-O}(SIG-C \cdot G) \tag{3-17}$$

Gateway identification value response and data saving: MS:OSS transfers the response message according to the cloud computing server request by attaching the cryptogram created in 5. After that, cloud computing server decodes the received cryptogram (CC_1, CC_2) and confirms its identification value. The server creates index (IN_C·G) based on completely homomorphic addition.

$$IN_C \cdot G = E_{RLWE-PUB-O}(Gateway-ID) \oplus E_{RLWE-PUB-O}(SIG-C \cdot G) \tag{3-18}$$

3.3 Proposed Design of Encryption Communication Message Protocol

This part designs a message communication protocol using the signature value and key created in the previous part. The user logs into CSS:DB. CSS verifies user information and requests the ID value from the user. The user generates the ID value and verifies the ID value from MS:OSS. The signature value is verified after requesting and receiving the signature value from the gateway. Then, after verifying the signature values of the user and gateway, the data are verified by performing the sum operation based on fully homomorphic encryption. Lastly, the data are sent from the user by encrypting V_SK1·2. Figures 5 and 6 show the details of the communication protocol in a cloud computing-based IoT convergence environment. Login: A user extracts the signature value (SIG_USER), creates CA_1 and CA_2 cryptograms using the open key of MS:OSS, and transfers the cryptograms to (CSS) cloud computing server.

$$CA_1 = E_{RLWE-PUB-O}(User-Info) || SIG-USER \tag{3-19}$$

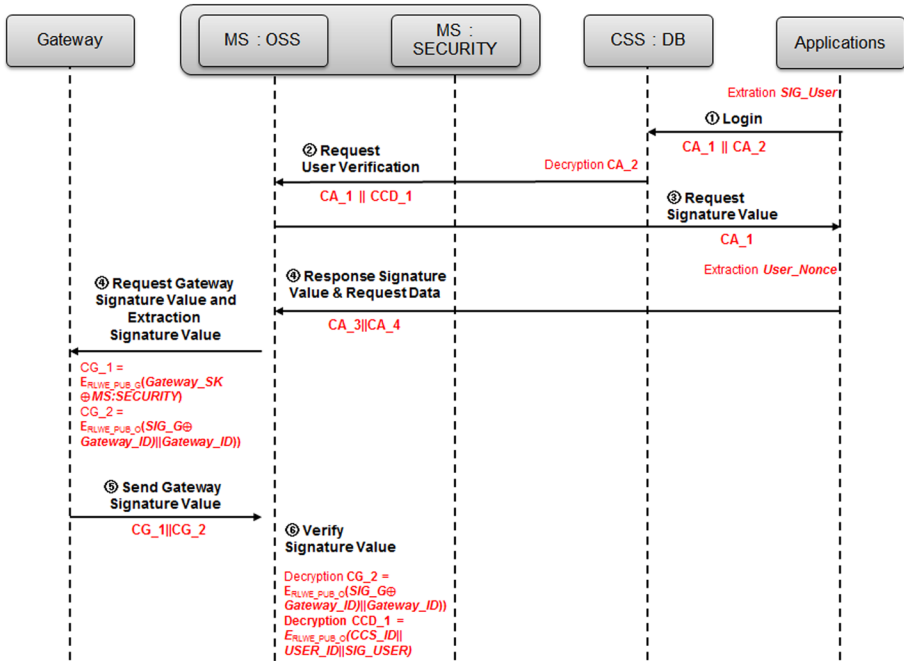


Fig. 5 A design of communication protocol in cloud centric IoT environment-1

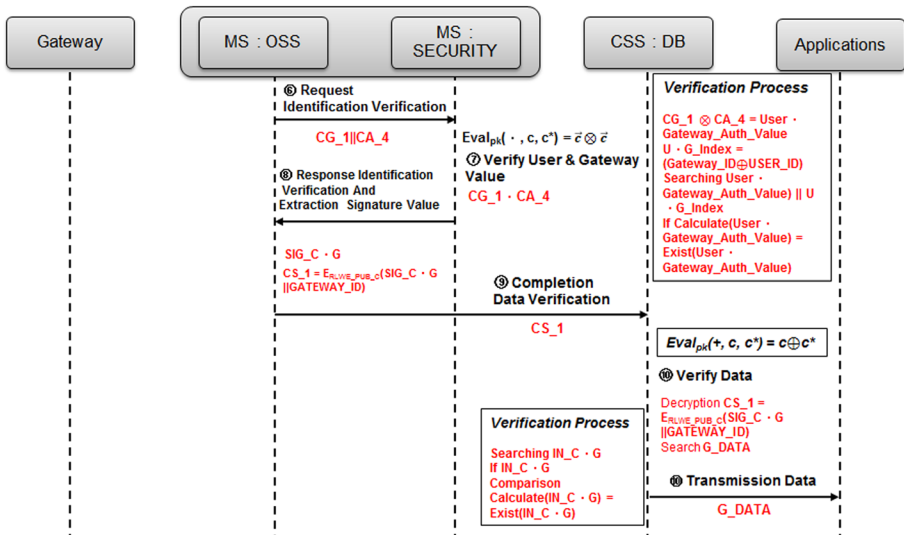


Fig. 6 A design of communication protocol in cloud centric IoT environment-2

$$CA_2 = E_{RLWE-PUB-C}(User-Info) || USER-ID \tag{3-20}$$

Confirmation of identification value and signature value request: The cloud server decodes CA₂ transferred and confirms the identification value. After that, it performs encryption algorithm with the open key of MS:OSS and creates CCD₁.

$$CCD_1 = E_{RLWE-PUB-O}(CCS-ID||USER-ID||SIG-USER) \tag{3-21}$$

Signature value request and detection: To verify the information and identification value transferred to the cloud server, MS:OSS requests the signature value from the user. After receiving the request message, the user extracts a random value and creates cryptogram (CA₃) using the open key. Then the user creates CA₄ using the gateway open key and transfers the response message to MS:OSS.

$$CA_3 = E_{RLWE-PUB-S}(User-ID)||USER-Nonce) \tag{3-22}$$

$$CA_4 = E_{RLWE-PUB-G}(USER-SK)||ME:SECURITY-SK) \tag{3-23}$$

Gateway signature value request and extraction: MS:OSS receives the data from user and requests the signature value from the gateway. After that, the gateway performs encryption with the user’s open key and creates cryptogram (CG₁). Also, the gateway creates cryptogram (CG₂) with its open key and sends a response message.

$$CG_1 = E_{RLWE-PUB-A}(Gateway-SK \oplus MS:SECURITY) \tag{3-24}$$

$$CG_2 = E_{RLWE-PUB-G}(SIG-G \oplus Gateway-ID)||Gateway-ID) \tag{3-25}$$

Signature value verification: MS:OSS receives the signature value from the gateway and decodes cryptogram (CG₂, CCD₁). The identification value and signature value (SIG_G) of the gateway are confirmed in CG₂. CCD₁ confirms the cloud identification value, user identification value, and the user’s signature value as the cryptograms transferred from the cloud computing server. With the SIG_G of confirmed parameter based on hash tree, the signature value (SIG_{CG}) is verified.

$$CG_2 = E_{RLWE-SECURITY-O}((SIG-G \oplus Gateway-ID)||Gateway-ID)) \tag{3-26}$$

$$N'SIG_G = E_{PUB-S}(Gateway-ID \oplus USER-ID||SIG-USER) \tag{3-27}$$

$$\text{Confirms } E'SIG-G = N'SIG-G \tag{3-28}$$

Verification of identification value: MS:OSS verifies the signature value and transfers a verification request message to MS:SECURITY by attaching the cryptograms (CG₁, CA₄).

Verification of identification value between the user and gateway: MS:SECURITY performs Tensor Product on the cryptograms (CG₁, CA₄) and verifies the mutual authentication value based on $Eval_{pk}(\cdot, c, c^*) = \vec{c} \otimes (c^*)$, the completely homomorphic cryptogram algorithm.

$$CG_1 = E_{RLWE-PUB-G}(Gateway0-SK \oplus MS:SECURITY) \tag{3-29}$$

$$CA_4 = E_{RLWE-PUB-G}(USER-SK)||ME:SECURITY-SK)$$

$$CG_1 \otimes CA_4 = \vec{c} \otimes (c^*) = User \cdot Gateway_Auth_Value,$$

$$U \cdot G_Index = (Gateway_IDUSER_ID)$$

$$\text{Searching } User \cdot Gateway_Auth_Value||U \cdot G_Index$$

$$\text{If Calculate } (User \cdot Gateway_Auth_Value) = Exist(User \cdot Gateway_Auth_Value)$$

Identification value verification response and signature value extraction: MS:SECURITY transfers a verification response request message from MS:OSS. MS:OSS extracts a hash tree-based signature value, connects it with the identification value of gateway, and encrypts it using the open key of cloud computing server to create cryptogram (CS₁).

$$CS_1 = E_{RLWE-PUB-C}(SIG-C \cdot G || \text{Gateway-ID}) \quad (3-30)$$

Transfer of data verification completion message: MS:OSS attaches the cryptogram (CS₁) created during 8 from the cloud computing server and transfers the verification completion message. Data verification and transfer: The cloud computing server decodes the received message and confirms the gateway identification value and signature value. It then creates an index value based on it and searches data by referring to the identification value of gateway. After that, the cloud computing server compares and analyzes the index with the index created in the previous part. Finally, the cloud computing server transfers the corresponding data after comparison and analysis to the user.

$$CS_1 = D_{RLWE-SECURITY-C}(SIG-C \cdot G || \text{Gateway-ID}) \quad (3-31)$$

Searching $IN_C \cdot G$

Comparison Calculate $(IN_C \cdot G) = \text{Exist}(IN_C \cdot G)$

4 Simulations or Evaluation

4.1 Efficiency Evaluation

To conduct comparison and analysis with the existing cryptogram IoT environment, the performance analysis environment used Java (Jdk 1.8.0_31)-based Mysql 5.7.18, SQL developer, and Eclipse Software in Intel Core2 Quad CPU Q9400 2.66 GHz, 4.00 GHz, Windows 7 Ultimate K 62bit OS environment. A performance analysis was conducted on mutual authentication and verification among message encryption, message decode, and between user and gateway. The speed of existing encryption system (T-DES with RSA in IoT, AES with RSA in IoT, AES with ECC in IoT) and the proposed encryption communication protocol (Ring Learning with error-based completely homomorphic cryptogram protocol) is shown in Fig. 7. The proposed encryption communication protocol confirmed a 30 ms improved speed for encryption compared to the 3DES-symmetric key-based encryption communication protocol and about 6.1 ms improved speed of decode performance in the AES encryption communication protocol. In an open key-based ECC encryption protocol, the encryption speed of 0.2 ms and decode speed of 0.4 ms were confirmed. To apply to each environment the study equally set the information, verification information and signature value of user to 1024 bit, 1024 bit, 160 bit each, and used key value of encrypted algorithm applied to each environment to conduct performance analysis. In previous comparative analysis, when conducting RSA verification based T-DES data transfer, the study used RSA2048, Triple-Des, when conducting RSA verification based AES data encryption, the study used RSA2048, Triple-Des and RSA2048, AES-CTR, and when conducting ECC verification based AES data encryption, the study used ECC(233bit), AES-CCMP.

The graph comparing the speed of authentication and verification between user and gateway is as in Fig. 8. After creating a key, mutual authentication was carried out based on the user's information and identification value and the gateway's identification value,

Fig. 7 Cryptography communication analysis exist system and proposed cryptography communication in cloud centric IoT environment

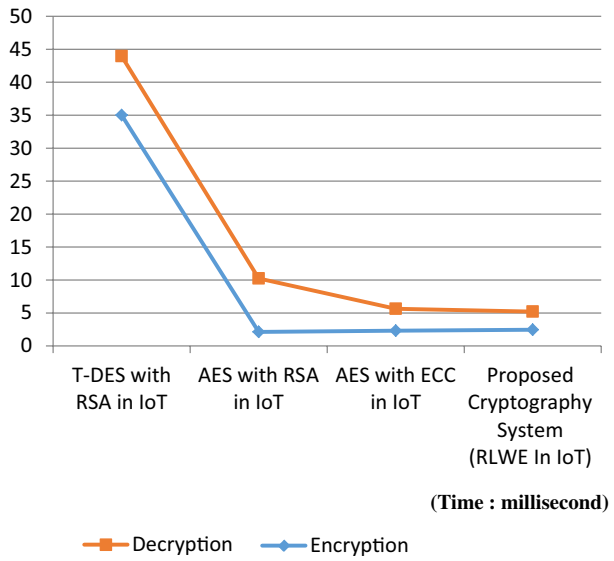
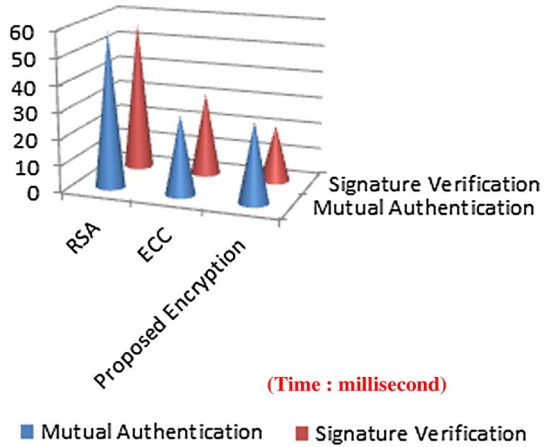


Fig. 8 Speed analysis mutual authentication and signature verification of exist cryptography system and proposed cryptography system



resulting in speed improvements by 21 ms compared to the existing RSA cryptogram and 2 ms compared to ECC cryptogram. In addition, signature value verification was performed during mutual authentication process based on the signature value created in a cloud computing server and IoT-based Management Service Server. This showed speed improvements by 23 ms from the existing RSA cryptogram and 10 ms from ECC cryptogram. Also, there was case of not considering the space complexity as the performance of volatile memory of recent PC specification was sufficient, but as device of IoT environment requires limitation of performance, the study conducted comparative analysis on usage of memory in accordance with message encryption. The difficulty in accordance with message decoding of proposed password protocol is learning with error problem, which adds error value when conducting coding process with other key, not a secret key which makes it safe from differential attack.

4.2 Safety Analysis and Security Evaluation

In this section, we analyze the efficiency of the proposed protocol and conduct a safety analysis according to the vulnerability. We analyze the time complexity of encryption and decryption in the proposed communication protocol, space complexity of message encryption, difficulties in decrypting encrypted messages, and attack success rate against random attacks. Big-O notation is used for the complexity analysis process and the memories used in space complexity are written as ω (Table 2).

1. *Threat of user's privacy exposure* A risk on the infringement of user's personal information is increasing in an ICT environment. There is an enormous amount of data collected through various IoT devices which are facing the risk of data leakage. To resolve this, cryptogram was performed on user and gateway information with a completely homomorphic cryptogram technique to verify $CG_1 \otimes CA_4 = \vec{c} \otimes c^* = \text{User_Gateway_Auth_Value}$, without decoding; to infer the safely decoded message, there is a difficulty for the n-th equation on the Ring.
2. *Threat from non-authorized access* When an attacker steals user's or gateway's identification value and tries to access via MS Domain, MS:OSS performs verification on the created signature value (SIG_G), while MS:SECURITY verifies user authentication information and calculates and confirms the value of Index ($U_G_Index = (\text{Gateway_ID} - \text{USER_ID})$), preventing non-authorized access.
3. *Middle and replay attack* For an attacker to decode RLWE-based cryptogram algorithm when trying to analyze the message he stole from the designed communication, noise value (rM^T) amplifies as encryption is attempted with a different key, not an entity's personal key, failing the middle attack.
4. *Threat of message leakage* The mutual authentication and access control technique in the existing cloud computing-based is not appropriate to use in a cloud computing-based IoT environment, and there is a security threat on information management. The proposed cryptogram protocol has higher efficiency than the existing encryption algorithm (RSA, ElGamal, ECC) as well as ensures higher message safety in terms of security. When the encrypted message is leaked, $d = 2048$ $q \approx 260$ is asked at $R_q = \mathbb{Z}_q[x]/(x^d + 1)$, which makes decoding impossible.
5. *Expandability on different devices* The open key cryptogram (RSA, ECC) in the existing IoT and cloud computing service methods has performance limitations or causes overhead due to huge calculation load. A cloud computing-based IoT environment requires light weight and high speed. While ABE (Attribute-based Encryption) can be applied, it causes too many restrictions on DB and too much calculation load. The proposed cryptogram protocol manages data using RLWE-based $\text{Eval}_{pk}(+, c, c^*) = c \oplus c^*$, providing expandability to different devices.

5 Conclusions

This research applied a RLWE-based, completely homomorphic cryptogram algorithm to a cloud service -IoT convergence environment and designed a communication protocol, for the proposed encryption communication protocol. After performing authentication between user and IoT-based gateway, user registration was completed and key creation protocol was designed, and the data collected from device was transferred to a cloud

Table 2 Exist cryptography and proposed cryptography of efficiency comparison

	RSA	Elgamal	ECC	Proposed cryptography
Time complexity of encryption	MessageO(n)	MessageO(n) + MessageO(1)	Message-O(1) + Message-O(1)	Message-O(1)-O(2n)
Time complexity of decryption	C(Message)O(n)	C(Message)O(n) + C(Message)O(1)	C(Message)O(1) + C(Message)O(1)	C(Message)-O(1)
Space complexity of encryption	$\omega_i O(n)$	$\omega_i O(n) + \omega_i O(n)$	$2\omega_i$	ω_i
Difficulty decryption of encryption message	Prime factorization problem	Discrete logarithm problem	Discrete logarithm problem	Learning with error problem

computing-based server and designed a technique to create and manage an index. Next, a communication protocol was designed based on user registration, key creation procedure, and data management technique, preventing the damages from data leakage and privacy threat issues. This study designed user registration, key creation, message management technique, communication protocol to prevent damages from data leakage and privacy threats. It also effectively responded to data falsification by designing a hash tree-based certificate management technique in signature value management.

The proposed encryption protocol sets user information, identification value, gateway identification value, and cloud identification value, and we conducted performance analysis and security evaluation with the existing encryption communication system. By considering various IoT-based devices, the study analyzed time and space complexity and confirmed safety against differential attack. The study also analyzed safety against the threat of user's privacy exposure, unauthorized access, middle and replay attack, and message leakage, which are existing vulnerabilities. The proposed encryption communication protocol provided flexibility with the expandability of different devices, in contrast to the existing encryption communication protocol. Thus, the encryption communication protocol proposed in this paper is expected to solve vulnerabilities such as data leakage and infringement on user information.

References

1. Park, J. (2014). Future of Internet of Things (1st ed.). ETNEWS.
2. Choi, Kyung, & Kim, Mi-Hui. (2016). Reseach on convergence of internet-of-things and cloud computing. *JKCA*, 16(5), 1–12.
3. Nam, H.-J. (2017). Security and privacy issues of fog computing. *KICIS*, 1(42).
4. Pyo, C. S., Kang, H. Y., Kim, N. S., & Bang, H. C. (2013). Trends and development prospects of IoT(M2M). *The Journal of Korean Institute of Communication and Information Sciences*, 30(8), 3–8.
5. Gentry, C. (2009). *A fully homomorphic encryption scheme*. Stanford University, Ph.D. Thesis.
6. Cha, H.-J. (2015). Design of the secure data management system using homomorphic encryption. *KICTS*, 4(15).
7. Gentry, C., & Halevi, S. (2011). Implementing gentry's fully homomorphic encryption scheme. In *Advances in Cryptology –EUROCRYPT*, Lecture Notes in Computer Science (Vol. 6632, pp. 129–148).
8. Kim, J.-H. (2013). Fully homomorphic encryption scheme without key. *FHE, KICS*, 5(38), 428–433.
9. Kim, J.-H., You, S.-K., & Lee, S.-H. (2013). Fully homomorphic encryption scheme without key switching. *KICS*, 38(5), 428–433.
10. Yang, H., Kim, H., Tang, D., & Li, H. An Efficient Somewhat HE scheme over Integers and Its Variation. IETE Technical Review.
11. Song, Y.-J., & Park, K.-Y. (2009). Homomorphic encryption technique for database outsourcing. *KIISC*, 19(3), 80–89.
12. Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2011). Fully homomorphic encryption without bootstrapping. *IACR Eprint Archive*.
13. Yang, H., & Kim, H. (2013). A fully homomorphic encryption scheme based on somewhat homomorphic encryption scheme with two integers. In *Proceedings of KICS winter conference 2013* (pp. 76–77).
14. Kim, S.-J., Kim, J.-M., & Cho, I.-J. (2012). Design of configuration management using homomorphic encryption in mobile cloud service. *KIICE*, 16(10), 2217–2223.
15. Kim, H.-S., & Lee, S.-W. (2013). Homomorphic encryption scheme and applications for cloud computing security. *SERSC*, 10(2), 213–224.
16. Cho, N.-S., & Hong, D.-W. (2008). Technical trend of the searchable encryption system. *ETRI*, 23(4).
17. Stehle, D., & Steinfeld, R. (2010). Faster fully homomorphic encryption. *Lecture Notes in Computer Science*, 6477, 377–394.
18. Damgard, I., & Jurik, M. (2001). A generalization, a simplification and some applications of Paillier's probabilistic public-key system. In *Public Key Cryptography—PKC* (pp. 119–136).
19. Gennaro, R., Gentry, C., & Parno, B. (2010). Non-interactive verifiable computing: Outsourcing computation to untrusted workers. *Lecture Notes in Computer Science*, 6223, 465–482.

20. Rivest, R., Addleman, L., & Dertouzos, M. (1978). On data banks and privacy homomorphism. In *Foundations of secure computation* (pp. 169–177).
21. Kolesnikov, V., Sadeghi, A., & Schneider, T. (2009). How to combine homomorphic encryption and garbled circuits—Improved circuits and computing the minimum distance efficiently. In: *Proceedings of SPEED 2009* (pp. 100–121).

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.



Byung-Wook Jin received his B.S. degree in Multimedia Science from ChungWoon University, Chungnam, Korea in 2011, and M.S. degree in Computer Science from Soongsil University, Seoul, Korea, in 2013. He is currently a Ph.D. Course in the Computer Science, Soongsil University. His research interests include Internet of Thing, Authentication System, Network Security.



Jung-Oh Park is a Professor of Department of Paideia, Sungkyul University, Korea. His research interests include: PKI and Ubiquitous Computing. His research interests include Internet Of Thing, Authentication System, Network Security.



Hyung-Jin Mun received his B.S., and Master degree in Mathematics from ChungNam National University, Republic of Korea in 1996 and 2002. He received Ph.D. degrees in Computer Science from Chung-Buk National University in 2008. He was an assistant professor and associate professor in Yanbian University Science and Technology in China, in 2009 and 2010. Currently, he works at Sungkyul University as an assistant professor. His research interests include Personal Privacy, Access Control, Authentication, and Network Security.