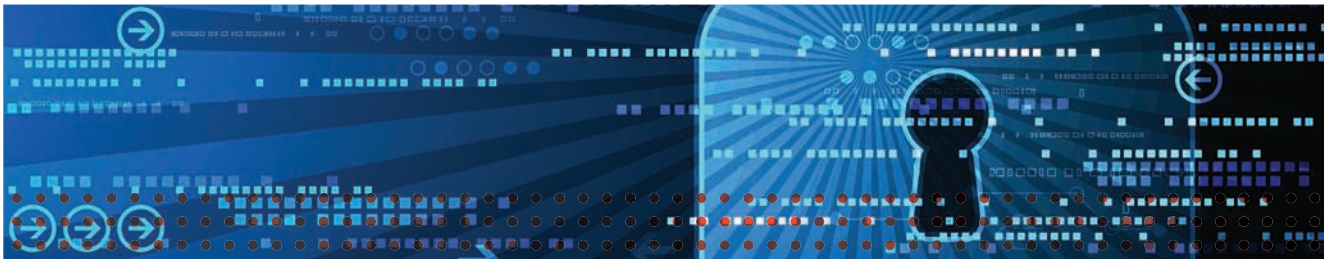

Secure Data Sharing and Searching at the Edge of Cloud-Assisted Internet of Things

Muhammad Baqer Mollah and Md. Abul Kalam Azad, Jahangirnagar University
Athanasios Vasilakos, Luleå University of Technology

This article proposes an efficient data-sharing scheme that allows smart devices to share securely data with others at the edge of cloud-assisted Internet of Things (IoT). We also propose a secure searching scheme to search desired data within own/shared data on storage.

The Internet of things (Iot)[1] is considered as a future internet that extends the connection of the internet to all kinds of real-world physical smart devices. A report by Cisco (www.cisco.com/c/dam/en_us/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf) estimates that by 2020 around 50 billion of such smart devices will be connected to the Internet. By connecting these billions of smart devices to the Internet, the IoT will provide developed smart and autonomous cyber-physical environments in the area of smart grids, smart cities, smart homes, smart medical and healthcare systems, wearable technologies, transportation systems, etc. However, the majority of these devices are part of a large platform, hence, a huge amount of data are generated that requires high computational capabilities for storage, processing, and analyzing purposes in a secure and efficient manner. Generally, the smart devices have limited resources. On the other hand, cloud resources have virtually unlimited storage and processing capabilities with scalability and on-demand accessibility anywhere. Thus with the help of the cloud, the IoT smart devices can relieve the burden of limited resources.[2] For IoT applications, smart devices require low latency, high data rate, fast data access, and real-time data analytics/processing with decision-making and mobility support. Due to several drawbacks, the cloud cannot fulfill the aforesaid requirements. However,

 2325-6095/17/$33.00 © 2017 IEEE

edge computing adds many benefits to cloud-assisted IoT[3] and supports aforesaid requirements by keeping data processing, communications, and storage operation on edge servers that are close to the devices at the edge of the networks. Moreover, due to smart devices' limited range of connectivity, the edge servers can serve as intermediaries for communications over long distances. These edge servers are any personal device or mobile device, stand-alone servers, or network devices that are hosted within one hop far from the end devices. In addition, the edge servers also cooperate and connect strongly with cloud servers.

With the increasing number and availability of smart devices, data sharing is offered within cloud-assisted IoT applications. The data are of little use if the smart devices do not share data with other devices. Data sharing at the edge allows smart devices to share data with lower latency and have fast data access and higher bandwidth. The next generation wireless communications technology (5G) will greatly depend on such solutions where massive IoT smart devices are interconnected with high data rates at ultralow latency. Yi et al. evaluate a performance comparison of the cloud and edge/fog server in terms of latency and bandwidth.[4] The results show that when using fog and cloud server, the latencies are 1.416 and 17.989 ms, respectively, and the uplink/downlink bandwidth for fog and cloud are 83.723/101.918 and 1.785/1.746 Mbps, respectively.

When the IoT smart devices share data with other devices, potential security issues arise such as data leakage, modification, integrity, and unauthorized access.[5] Hence, it is essential that such shared data be ensured confidentiality, integrity, and access control while sharing at the edge. Furthermore, a secure data-searching technique is needed to search and retrieve the shared data by authorized devices.

At present, there are few solutions to address the challenges of secure data sharing and searching in clouds. Typically, to ensure confidentiality of shared data, symmetric key,[6] public key,[7] and homomorphic[8] encryption-based mechanism are currently used. Access control policies based on access control list[9] and dynamic attribute[10] are used for access control purposes. Searchable encryption based on symmetric[11] and public[12] keys are used for searching the desired data. In all these schemes, for data security, major security-oriented processing such as encryption, decryption, and access control mechanisms are handled by the user's device itself. In IoT, the resource-limited smart devices cannot handle these computation intensive operations because the security-oriented operations will increase the heavy computational burden.

In this paper, by considering the aforementioned limitations of current solutions for resource-limited smart devices, we propose a lightweight cryptographic scheme so that IoT smart devices can share data with others at the edge of cloud-assisted IoT wherein all security-oriented operations are offloaded to nearby edge servers. Furthermore, although initially we focus on data-sharing security, we also propose a data-searching scheme to search desired data/shared data by authorized users on storage where all data are in encrypted form. Finally, security and performance analysis shows that our proposed scheme is efficient and reduces the computation and communication overhead of all entities that are used in our scheme.

The key contributions of our work are summarized as follows:

1. First, we propose a secure data-sharing scheme at the edge of cloud connected IoT smart devices that utilizes both secret key encryption and public key encryption. In this scheme, all security operations are offloaded to nearby edge servers, thereby, greatly reducing the processing burden of smart devices.
2. Next, we propose a searching scheme to search desired data securely by authorized users within encrypted, stored, shared data in edge/cloud without leaking keyword, secret key, and data, thereby reducing both computation and communication overhead during search and data retrieval.
3. Then, we show the verification process of the shared data as well as data retrieval after searching. Hence, our proposed scheme attains the integrity of shared data and searching resultant data.
4. Finally, we analyze the performance of our proposed scheme and prove that our scheme is efficient and can be used in IoT applications.

The rest of the paper is organized as follows. In the second section, we present the related cryptographic mechanisms that are used in our proposed scheme. We provide a background overview followed by a brief description of our proposed scheme. We then analyze the performance and compare it to related works. Finally, we conclude our paper by stating future works.

Related Cryptographic Mechanisms

This section describes the most important related cryptographic mechanisms those are used in our proposed scheme.

Secret Key Encryption

In secret key encryption, the user device first generates a secret key. Then the data are encrypted with the key and is sent to the recipient user device. By using the same key, the recipient device can recover the data from the encrypted form of data by decrypting with the secret key. To keep the process secret, the key is shared with communicating devices using secure communication principals.

Public Key Encryption

In public key encryption, there are two kinds of keys: a public key and a secret key. Before sending, the data are encrypted with the recipient's public key and after receiving the data are decrypted by the recipient's secret key to recover the data.

Searchable Secret Key Encryption

This mechanism is based on secret key encryption that allows searching specific data on outsourced storage encrypted data via a generated trapdoor. The data owner device needs to share the secret key with all authorized devices to generate the trapdoor.

One-Way Hash Algorithms

After communicating, it is necessary to verify the data are not modified in any way in between the sender and receiver. This verification is called integrity checking. Generally, the integrity checking is carried out by a hash function. If a publicly known hash function is applied to the data with a specific length, then the outcome is called hash value of the data. However, this process is only one-way; it cannot recover the corresponding data from the hash value. The sender sends the data with its corresponding hash value. After receiving the data, the receiver checks data integrity by the same way, applying the hash function to the received data; if both hash values are the same, then the data has been shown to be authentic.

Digital Signature

In the digital signature process, an authorized user is issued a key pair (public and secret) and a corresponding digital certificate that ensures the identity of the user or entity. By using a secret key, the sender signs the data or hash value of the data. After receiving the data and signed data, the recipient verifies both the certificate and signed data by using the sender's public key.

Background

Here, we present the related cryptographic mechanisms that are used in our proposed scheme.

Overall System Architecture

In our scheme, we consider a model of IoT data sharing and searching at the edge system that consists of four main entities.

Smart Devices

The smart devices, connected with the physical world, are entities that have a large amount of data to be shared with other devices or stored in edge/storage servers. The authorized smart devices are allowed to decrypt/download the shared data and also retrieve desired data after searching from encrypted storage.

Edge Servers

The edge servers are semitrusted and secure entities located at the proximity of smart devices that are capable of sharing data with a number of smart devices. It is responsible for security-oriented operations such as secret key generation and management, encryption, and decryption. The edge servers are maintained by clouds. Moreover, the edge servers provide data storage and processing of the smart devices.

Certificate Authority

The certificate authority is fully trusted and is responsible for issuing certificates to edge servers.

Key Generation Server

The key generation server is also a trusted third party that is responsible for generation of public and secret key pairs. As shown in Figure 1, the data owner and recipient smart devices are connected to each other by edge servers and edge servers are interconnected with each other so that data are shared and searched within the IoT scenario.

Threat Models

We focus on two types of threats during sharing and searching the data at the edge:

1. Insider threats are initiated by malicious insiders such as smart devices and/or edge servers that want to access/disclose/modify the stored/shared data.
2. Outsider threats are initiated by unauthorized outsider smart devices to access the data.

Insider threats can prove to be more devastating because they are generally launched by trusted entities. Because people trust insider entities, the research community focuses more on outsider attackers.

Trust Model and Assumptions

In our scheme, we assume that the edge servers are semitrusted and are able to achieve security over

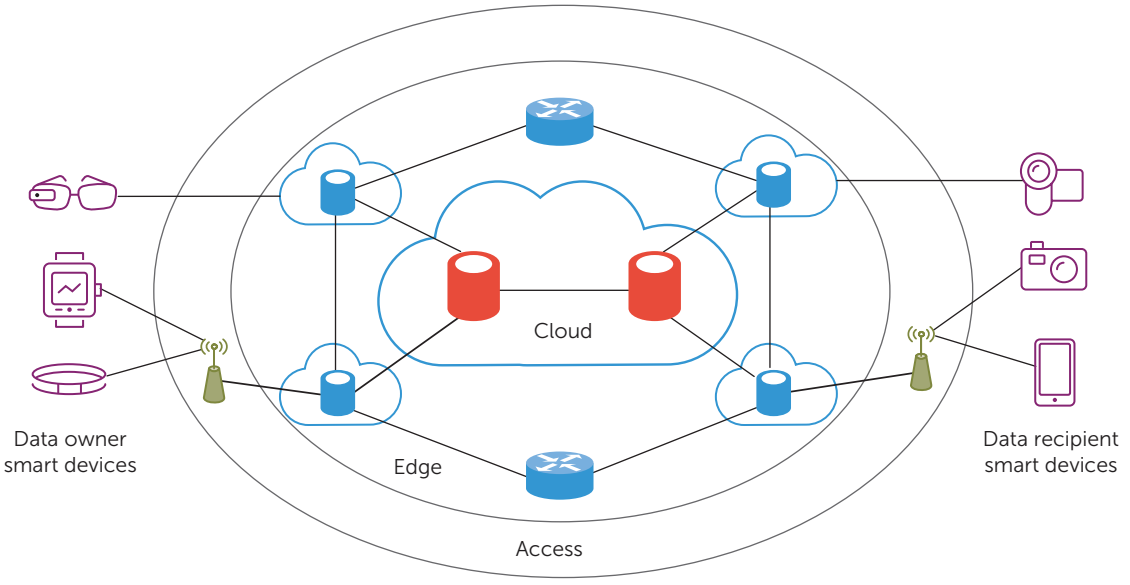

FIGURE 1. Cloud-assisted Internet of Things scenario.

shared/searched data. Therefore, the edge servers work fairly well with other entities.

We assume that in edge servers there exists a secret key generator that delivers securely these keys to other edge servers. Finally, we assume that the smart devices are not capable of key generation, encryption, or decryption and cannot make own data secure.

Secure-Data Sharing and Searching at the Edge

In this section, we present our proposed scheme that secures the sharing and searching of data at the edge of cloud-assisted IoT. Before data sharing and searching, all users need to register with edge servers by username and password to avail data sharing, downloading, desired data searching and retrieving. Our proposed scheme consists of four parts: 1) key generation, 2) data and keywords uploading, 3) data sharing and downloading and 4) data searching and retrieval.

Key Generation

In our scheme, the edge servers generate two kinds of secret keys on behalf of data owner smart devices as follows: 1) 256 bit keys are randomly generated, and 2) two kinds of keys, *Sec.Key* and *S.Sec.Key*, are assigned that are used for data-sharing and -searching purposes, respectively. With the help of the list uploaded by the data owner smart device, the edge server generates both secret keys differently and uniquely.

Data and Keywords Uploading

The data owner first puts the username and password to login into a nearby edge server from a smart device. After collecting the data from the physical systems, the data are transferred from the smart device to nearby edge servers. In addition, the data owner sends some related keywords of the data so that any authorized users can search the data and a list of recipient users that are authorized to access the data. Before uploading the data from edge server to storage, the data and its associated keywords are encrypted. And finally, to verify data integrity, the encrypted data are signed. Therefore, after receiving the data, keywords, and list, the edge server works as follows:

Encrypt the data with secret key for sharing as

$$C.Share \leftarrow Encrypt\ (Data,\ Sec.Key)$$

Next, by using secret key for searching, the keywords are encrypted as

$$C.KW.Search \leftarrow Encrypt\ (Keywords,\ S.Sec.Key)$$

The edge server receives pair of keys from key generation server such as public key *Public.Key* and private key *Private.Key* on behalf of the data owner and every recipient smart device with the help of list provided by the data owner's smart device. Moreover, the edge server is issued a digital certificate *Dig.Cert* from certificate authority that guarantees

Table 1. Data structure of uploading from smart device to edge server under a username.

Serial No.	Data	List	Keywords
.	.	.	.
.	.	.	.
N	$Data_n$	$List_n$	$Keywords_n$
.	.	.	.
.	.	.	.

Table 2. Data structure of uploading data from edge server to storage under different usernames.

Username	Encrypted data	.....	Signed hash value	Signature
..	..	..	..	..
..	..	..	..	..
Username	C.Share	..	Signed.H_1	Dig.Cert
..	..	..	..	..
..	..	..	..	..

the validity of the edge server and contains its identification information.

To share securely with authorized devices, the secret key is encrypted with authorized recipients' public keys as

$$C.Sec.Key \leftarrow Encrypt\ (Sec.Key,\ Public.Key)$$

To ensure integrity, the edge server computes the hash value of encrypted form of the data by collision-resistant hash function as

$$H_1 \leftarrow Compute\ hash\ (Data)$$

Then, the edge server signs the hash value with the data owner's private key as

$$Signed.H_1 \leftarrow Sign\ (H_1,\ Private.Key)$$

Finally, the edge server uploads the tuple (*C.Share* || *C.Sec.Key* || *C.KW.Search* || *Signed.*H_1 || *Dig.Cert*) to the edge storage or cloud as per requirements under the username. After verifying *Dig.Cert*, the tuple is stored in storage.

The data structure of uploading data from smart device to edge server under a username and from the edge server to storage under different usernames in tabular format is shown in Tables 1 and 2.

Data Sharing and Downloading

When an authorized smart device wants to access the data, it requests the nearby edge server after the login using the username and password. Then, the edge server works as follows:

The edge server downloads and stores the tuple (*C.Share* || *C.Sec.Key* || *C.KW.Search* || *Signed.*H_1 || *Dig.Cert*) under the data owner username from storage.

The edge server checks the digital certificate *Dig.Cert* as

$$Check\ (Dig.Cert)$$

Then, first it decrypts the encrypted form of secret key as

$$Sec.Key \leftarrow Decrypt\ (C.Sec.Key,\ Private.Key)$$

If the requested user is not authorized, it cannot decrypt.

After getting the secret key, the edge server decrypts the encrypted data and gets the data.

$$Data \leftarrow Decrypt\ (C.Share,\ Sec.Key)$$

To verify the integrity of decrypted data, the edge server works as

$$H_2 \leftarrow Calculate\ hash\ (Data)$$
$$H_1 \leftarrow Decrypt\ (Signed.H_1,\ Public.Key)$$
$$Check\ (H_1{=}H_2)$$

If matched, then data integrity is verified.

Finally, the data are sent to the authorized recipient.

Figure 2 illustrates the proposed secure data-sharing scheme.

Data Searching and Retrieval

To search a desired data on encrypted data on storage, the authorized user sends the keyword to the edge server after login. The edge server then works as follows:

The edge server receives the requested authorized user's secret key and generate trapdoor as

$$T_w \leftarrow Encryption\ (Keyword,\ S.Sec.Key)$$

Then T_w is uploaded to the storage server with a request to search.

Next, the storage server searches for the matched encrypted keywords under the username based on the trapdoor as

Check (C.KW.Search, T_w)

If found, the corresponding tuple (*C.Share* || *C.Sec.Key* || *C.KW.Search* || *Signed.*H_1 || *Dig.Cert*) is sent to the edge server.

The edge server checks the digital certificate *Dig.Cert* as

Check (Dig.Cert)

Then, first it decrypts the encrypted form of secret key as

Sec.Key ← *Decrypt (C.Sec.Key, Private.Key)*

If the requested user is not authorized, it cannot decrypt.

Then, the edge server decrypts to retrieve the data as

Data ← *Decrypt (C.Share, Sec.Key)*

To verify the integrity of decrypted data, the edge server works as

H_2 ← *Calculate hash (Data)*
H_1 ← *Decrypt (Signed.*H_1*, Public.Key)*
Check (H_1=H_2)

If matched, then data integrity is verified.

If verified, the data are sent to the requested authorized device.

As searchable secret keys are generated for every smart device, there is no possibility of matching any data that is not shared with the requested device or does not belong to the device. Figure 3 illustrates our proposed secure-searching scheme.

Performance Analysis

In this section, we describe the analysis of our proposed scheme.

Experimental Setup

We conducted an experimental analysis of our proposed scheme using PyCrypto (www.dlitz.net/software/pycrypto/) cryptographic toolkit. We analyzed the performance in Windows operating system (version 6.1.7600, Windows 7 32 bit) running an AMD CPU (AMD E-450 APU 1.65 GHz with 2-Gbyte RAM) and 512-Gbyte storage. We used Advanced Encryption Standard (AES), Rivest-Shamir-Adleman (RSA) algorithm, and SHA-256 for secret key encryption, public key encryption, and hash function implementations, respectively. For AES, we used the cipher block chaining mode. The performance of our

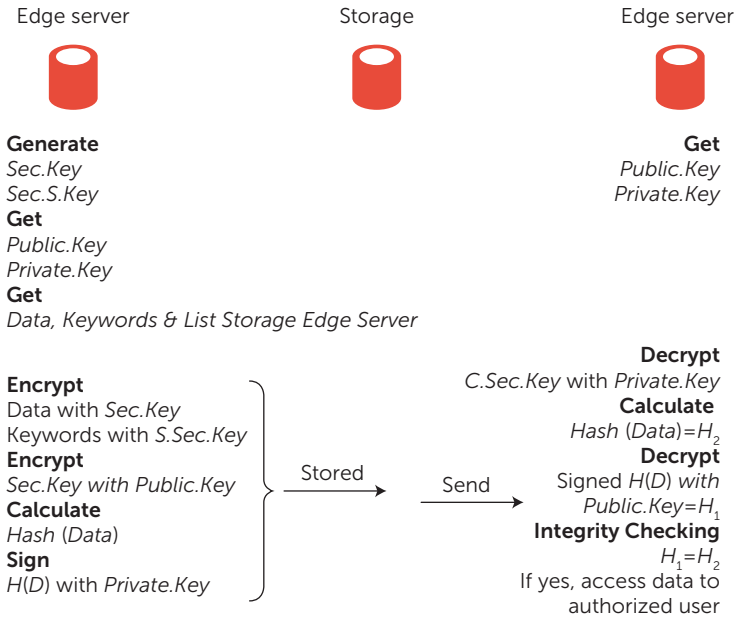

FIGURE 2. Proposed secure data-sharing scheme.

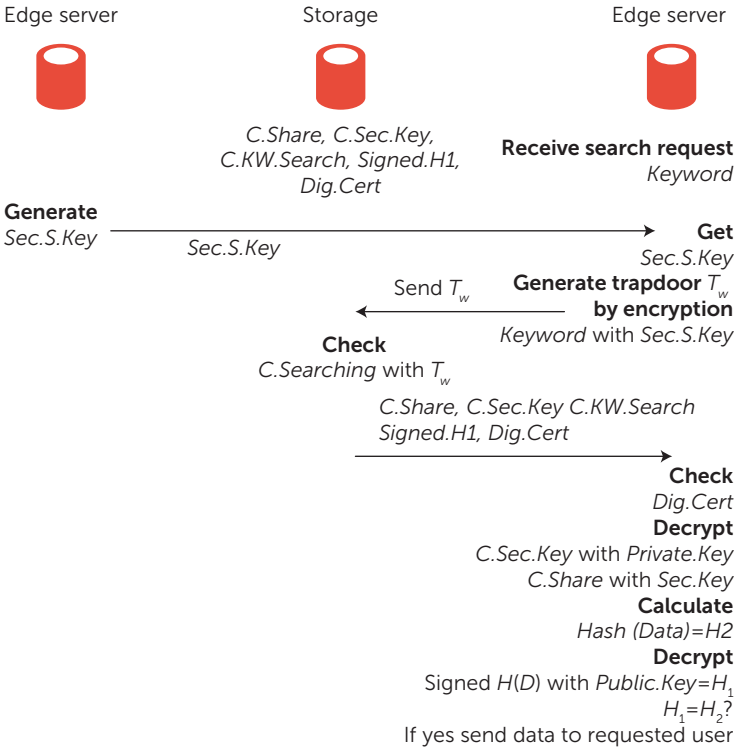

FIGURE 3. Proposed secure data-searching scheme.

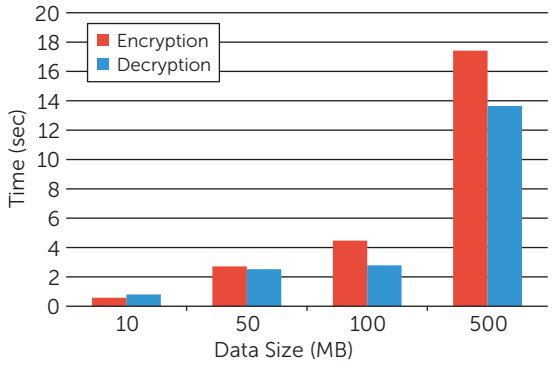

FIGURE 4. Processing time of encryption and decryption with Advanced Encryption Standard.

scheme is evaluated based on processing times. Therefore, we tested the processing time of data encryption/decryption with AES by 256-bit key size and different data size (10 to 500 Mbyte). We also tested key generation time, secret key encryption with RSA by key size of 1024 bit, hash value generation, and signing- and verification-processing times for both downloading and uploading sides. As discussed earlier, in our proposed scheme, all security-oriented operations of smart devices are executed at nearby edge servers; we focused on calculating the total processing time on edge servers.

Results

Our proposed scheme has been analyzed for the following different parts.

Key Generations

As discussed earlier, in our scheme, the edge servers generate 256-bit secret keys for both data sharing (*Sec.Key*) and searching (*S.Sec.Key*) purposes. From our analysis, we find the generation time of each secret key is 1.4 ms. This time is approximately constant and does not change with increasing the number of devices. Moreover, this time is negligible as compared with the encryption/decryption operation time.

Data Uploading

To calculate the total time consumed during data uploading, we need to analyze the processing times of encryption of the data by secret key, encryption of the secret key by the recipient's public key, calculating the hash values of data and hash value signing. These times help us to calculate the total processing time that are used to analyze the performance and compare our process with other schemes. In general, the encryption time is increased with increasing data size. Based on our analysis, we find processing times of 0.58, 2.71, 4.47, and 17.41 s during encryption of data with different data sizes such as 10, 50, 100, and 500 Mbyte, respectively. In addition, encryption of the secret key by the public key, hash value generation, and signing process take 4.8 ms to complete, and this time is almost constant.

Data Downloading

The total processing time during data downloading includes digital certificate checking, decryptions of ciphers of secret key and data, hash value generation, and sign verification. From our analysis, we find the decryption-processing time to be 0.80, 2.52, 2.79, and 13.65 s to get 10-, 50-, 100-, and 500-Mbyte data, respectively. In addition, the rest of the processing time for tasks such as cipher of secret key decryption, hash value generation, and sign verification is 3.7 ms. This time almost remains constant while varying the data size. However, we do not include digital certificate checking time. In Figure 4, we show the time consumption of encryption and decryption time with respect to data size.

Data Searching and Retrieval

To perform the desired data searching and retrieval, the sender end edge server consumes 1.4 ms to generate the secret key and 5.7 ms to encrypt searching keywords of 100-Kbyte size. On the other hand, the recipient end edge server takes about 13.01 ms to generate the trapdoor for searching encrypted keywords on the storage server. However, the remaining data retrieval time is almost identical to the data downloading times.

Related Works and Comparative Analysis

This section presents some related works of cloud-based secure data sharing and searching. We compare these works with ours in terms of processing time consumptions.

In several papers, cloud-based secure data-sharing schemes are presented whereby users can share their data with others/among a group via the cloud.[6,7,13,14] Xu et al. propose a certificateless proxy re-encryption scheme by using both symmetric key and public key encryption.[13] In this scheme, the data owner first encrypts the data with the secret key and the secret key is encrypted with data owner's public key, which is then sent to the cloud. After receiving, a proxy re-encryption agent inside, the cloud re-encrypts the encrypted form of the secret key and this re-encrypted form can be decrypted only by user's private key. However, the private-public key pairs are not associated with a certificate. The study by Seo et al. is also a certificateless scheme for data sharing but

Table 3. Comparison of total uploading time in seconds.

Data (Mbyte)	Ref. 6	Ref. 13	Ref. 7	Ref. 14	Our Scheme
10	6.43	13.05	14.95	14.59	0.5862
50	9.01	53.68	58.56	60.37	2.7162
100	17.37	99.69	112.41	155.15	4.4762
500	33.24	369.72	492.03	872.09	17.4262

Table 4. Comparison of total downloading time in seconds.

Data (Mbyte)	Ref. 6	Ref. 13	Ref. 7	Ref. 14	Our Scheme
10	6.48	9.91	9.90	10.45	0.8057
50	10.24	33.45	35.57	35.90	2.5237
100	20.68	57.14	59.14	61.59	2.7937
500	39.25	215.3	229.81	400.21	13.6537

without bilinear pairing.[7] In this scheme, the cloud is responsible for both secure data storage and public-private key pair generation. The data owner encrypts the data with public keys according to its access control policies and sends these encrypted data to the cloud. In this scheme, the decryption is performed twice by authorized public keys. At first, the cloud partially decrypts the encrypted data and then the recipient users decrypt finally to get the original data. Khan et al. utilize an incremental cryptography-based data-sharing scheme where the data are divided into several blocks and these blocks are then incrementally encrypted.[14] A trusted third party is used as a proxy for key generation, re-encryption, and access control purposes. Moreover, ElGamal cryptosystem and bilinear pairing are also used in this scheme. In the study by Ali et al., a secret key-based encryption and access control list for secure data sharing where a trusted thirst party is engaged in encryption/decryption, key management, and access control rather user's device itself is utilized.[6]

Other studies present secure data searching from encrypted data the on cloud.[11,15] In the study by Pasupuleti et al., a ranked keyword search algorithm is proposed so that the retrieved results coming back from the cloud to the requester are ranked based on the relevance scores instead of all data.[15] Another scheme has been proposed where the relevance score and k-nearest neighbor technique are used to make efficient multikeyword search that can return search results based on accuracy.[11]

However, the main drawbacks of aforesaid schemes are that they take much processing time due to computational-intensive bilinear paring, twice encryption/decryption processing, and most importantly using the cloud as we mention in introduction to this paper. Because of the limitations, IoT applications and services cannot fully use the cloud.

After calculating the total uploading and downloading times, we compare our results with several other cloud-based schemes[6,7,13,14] in Table 3 and 4. Because our trapdoor generation time is negligible compared to the retrieving time, we do not compare these times with the other cloud-based searching schemes.

In this paper, we present a proposed data-sharing and -searching scheme to share and search data securely by IoT smart devices at the edge of cloud-assisted IoT. The performance analysis demonstrates that our scheme can achieve better efficiency in terms of processing time compared with existing cloud-based systems. In future work, we plan on authenticating and accessing control challenges in this area. We hope that our proposed scheme is practical to be deployed and opens a new door in edge-oriented security research for cloud-assisted IoT applications.

References

1. M.R. Palattella, M. Dohler, A. Grieco, G. Rizzo, J. Torsner, T. Engel, et al., "Internet of Things in the 5G Era: Enablers, Architecture, and Business Models," *IEEE J. Selected Areas in Communications*, vol. 34, no. 3, 2016, pp. 510–527.

2. L. Wang and R. Ranjan, "Processing Distributed Internet of Things Data in Clouds," *IEEE Cloud Computing*, vol. 2, no. 1, 2015, pp. 76–80.
3. M. Satyanarayanan, P. Simoens, Y. Xiao, P. Pillai, Z. Chen, K. Ha, et al., "Edge Analytics in the Internet of Things," *IEEE Pervasive Computing*, vol. 14, 2015, pp. 24–31.
4. S. Yi, Z. Hao, Z. Qin, and Q. Li, "Fog Computing: Platform and Applications," *2015 3rd IEEE Workshop Hot Topics Web Systems and Technologies* (HotWeb), 2015, pp. 73–78.
5. J. Singh, T. Pasquier, J. Bacon, H. Ko, and D. Eyers, "Twenty Security Considerations for Cloud-Supported Internet of Things," *IEEE Internet of Things J.*, vol. 3, no. 3, 2016, pp. 269–284.
6. M. Ali, R. Dhamotharan, E. Khan, S. U. Khan, A.V. Vasilakos, K. Li, et al., "SeDaSC: Secure Data Sharing in Clouds," *IEEE Systems J.*, vol. 99, 2015, pp. 1–10.
7. S.-H. Seo, M. Nabeel, X. Ding, and E. Bertino, "An Efficient Certificateless Encryption for Secure Data Sharing in Public Clouds," *IEEE Trans. Knowledge and Data Engineering*, vol. 26, no. 9, 2014, pp. 2107–2119.
8. H. Kumarage, I. Khalil, A. Alabdulatif, Z. Tari, and X. Yi, "Secure Data Analytics for Cloud-Integrated Internet of Things Applications," *IEEE Cloud Computing*, vol. 3, no. 2, 2016, pp. 46–56.
9. J.B. Bernabe, J.L.H. Ramos, and A.F.S. Gomez, "TACIoT: Multidimensional Trust-Aware Access Control System for the Internet of Things," *Soft Computing*, vol. 20, no. 5, 2016, pp. 1763–1779.
10. F. Li, Y. Rahulamathavan, M. Conti, and M. Rajarajan, "Robust Access Control Framework for Mobile Cloud Computing Network," *Computer Communications*, vol. 68, 2015, pp. 61–72.
11. H. Li, D. Liu, Y. Dai, T.H. Luan, and X. Shen, "Enabling Efficient Multi-Keyword Ranked Search over Encrypted Mobile Cloud Data Through Blind Storage," *IEEE Trans. Emerging Topics in Computing*, vol. 3, no. 1, 2015, pp. 127–138.
12. H. Li, D. Liu, Y. Dai, and T.H. Luan, "Engineering Searchable Encryption of Mobile Cloud Networks: When Qoe Meets Qop," *IEEE Wireless Communications*, vol. 22, no. 4, 2015, pp. 74–80.
13. L. Xu, X. Wu, and X. Zhang, :CL-PRE: A Certificateless Proxy Re-Encryption Scheme For Secure Data Sharing with Public Cloud," *Proc. 7th ACM Symposium on Information, Computer and Communications Security*, 2012, pp. 87–88.
14. A.N. Khan, M.M. Kiah, S.A. Madani, M. Ali, and S. Shamshirband, "Incremental Proxy Re-Encryption Scheme for Mobile Cloud Computing Environment," *J. Supercomputing*, vol. 68, no. 2, 2014, pp. 624–651.
15. S.K. Pasupuleti, S. Ramalingam, and R. Buyya, "An Efficient and Secure Privacy-Preserving Approach for Outsourced Data of Resource Constrained Mobile Devices in Cloud Computing," *J. Network and Computer Applications*, vol. 64, 2016, pp. 12–22.

MUHAMMAD BAQER MOLLAH *(m.m.baqer@ieee.org) is a MSc student in the department of computer science and engineering at the Jahangirnagar University, Bangladesh. His research interests include advanced communication and security techniques for future wireless networks. He has a BSc in electrical and electronic engineering from International Islamic University Chittagong, Bangladesh. He is a Student Member of IEEE.*

MD. ABUL KALAM AZAD *(makazad@juniv.edu) is an associate professor in the department of computer science and engineering at the Jahangirnagar University, Bangladesh, and a PhD candidate in Ubiquitous Computing Lab at the University of Ulsan, Korea. His research interests include cloud computing, wireless sensor networks and information security. He has a BSc in computer science and engineering from Jahangirnagar University, Bangladesh, and MSc in computer science from KTH Royal Institute of Technology, Sweden. He is a Member of IEEE.*

ATHANASIOS VASILAKOS *(athanasios.vasilakos@ltu.se) is a professor in the department of computer science, electrical and space engineering at the Lulea University of Technology, Sweden. His research interests include networks, cloud computing, security and Internet of Things. He has a PhD in computer engineering from the University of Patras, Greece. He is a Senior Member of IEEE.*