Lightweighted Secure Searching Over Public-Key Ciphertexts for Edge-Cloud-Assisted Industrial IoT Devices

Wei Wang, *Member, IEEE*, Peng Xu, *Member, IEEE*, Dongli Liu, Laurence Tianruo Yang, *Senior Member, IEEE*, and Zheng Yan, *Senior Member, IEEE*

***Abstract*—For the industrial Internet of Things (IIoT), public-key encryption with keyword search (PEKS) is a type of applicable and promising encryption technique to maintain the data stored in clouds secure and searchable. To further improve the efficiency of searching, it is popular to introduce edge computing near the IIoT as the substitute for a cloud. However, the straightforward collaboration of the two techniques performs negatively in latency-sensitive applications since the sluggish encryption of PEKS by IIoT devices negates the instant reaction of edges. To meet this challenge, in this article, we exploit the capability of the edge-cloud architecture and propose a lightweight-designed scheme called edge-aided searchable public-key encryption (ESPE). It allows IIoT devices to delegate their costly cryptographic operations to the nearby edge for fast computing and guarantees that all outsourced ciphertexts are semantically secure. Consequently, ESPE accelerates the corresponding ciphertext procedures on edges and saves over 70% encryption cost of an IIoT device.**

***Index Terms*—Edge computing, edge-cloud, industrial devices, lightweight cryptography, public-key encryption with keyword search (PEKS).**

Manuscript received June 10, 2019; revised September 3, 2019 and September 25, 2019; accepted October 12, 2019. Date of publication October 31, 2019; date of current version February 28, 2020. This work was supported in part by the National Natural Science Foundation of China under Grant 61702206 and Grant 61872412, in part by the National Key Research and Development Plan of China under Grant 2017YFB0801804, in part by the Guangdong Provincial Key R&D Plan Project under Grant 2019B010139001, in part by the Guangdong Provincial Science and Technology Plan Project under Grant 2017B010124001, and in part by the Shenzhen Fundamental Research Program under Grant JCYJ20170413114215614. Paper no. TII-19-2437. *(Corresponding author: Wei Wang.)*

W. Wang is with the Cyber-Physical-Social Systems Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, and also with the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China (e-mail: viviawangww@gmail.com).

P. Xu is with the National Engineering Research Center for Big Data Technology and System, Services Computing Technology and System Lab, Big Data Security Engineering Research Center, School of Cyber Science and Engineering, Huazhong University of Science and Technology, Wuhan 430073, China (e-mail: xupeng@mail.hust.edu.cn).

D. Liu is with the Cyber-Physical-Social Systems Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China (e-mail: nsfldl@hust.edu.cn).

L. T. Yang is with the Cyber-Physical-Social Systems Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China, with the Department of Computer Science, St. Francis Xavier University, Antigonish, NS B2G2W5, Canada, and also with the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China (e-mail: ltyang@gmail.com).

Z. Yan is with the State Key Laboratory of Integrated Services Networks, School of Cyber Engineering, Xidian University, Xi'an 710071, China, and also with the Department of Communications and Networking, Aalto University, 02150 Espoo, Finland (e-mail: zyan@xidian.edu.cn).

This article has supplementary downloadable material available at http://ieeexplore.ieee.org provided by the authors. This includes a PDF file, which shows the proof that the IND-CKSA security of ESPE is based on the weak DBDH assumption in type-3. This material is 217 KB in size.

Digital Object Identifier 10.1109/TII.2019.2950295

I. Introduction

PLENTY of works [5] show that the edge-cloud design acts as a successful substitute for the original only-cloud platforms and compromise between powerful computation and extremely facilitated reactions in emerging applications of industrial Internet of things (IIoT) [2]–[4]. Edge computing, a popular and promising method to optimize the cloud computing systems, takes cloud services from the center of clouds to the edges of Internet, which are physically and logically closer to the IoT world. This method can provide on-demand services, such as instant caching and online analyses with low latency to meet the demands of IIoT in Industry 4.0 [1].

The other phase, the IIoT, has appeared since IoT has been widely used in the industrial field to guarantee the reliability and security of industrial processes. Cloud computing, big data analyses, and the IoTs are the fundamentals of reliability in Industry 4.0. In particular, according to big data analyses, IIoT can avoid potential risks and redundancies in producing processes. However, the goals of Industry 4.0 also include maintaining the information and the process secure [1], which indicates that data in IIoT must be securely utilized only by authorized consumers.

We implement the edge-cloud architecture in IIoT in a classic scenario, as illustrated in Fig. 1. Traditionally, IIoT devices (e.g., sensors, cameras) collect the data flow and synchronize with the cloud platforms due to limited storage. However, data managers are authorized to use the data, and they retrieve them from cloud platforms. Commonly, the communication latency during a complete data service would be considerably high since cloud platforms are physically and logically far from both IIoT devices and data managers. With the assistance of edge computing, each IIoT device stores its data flow to the closest

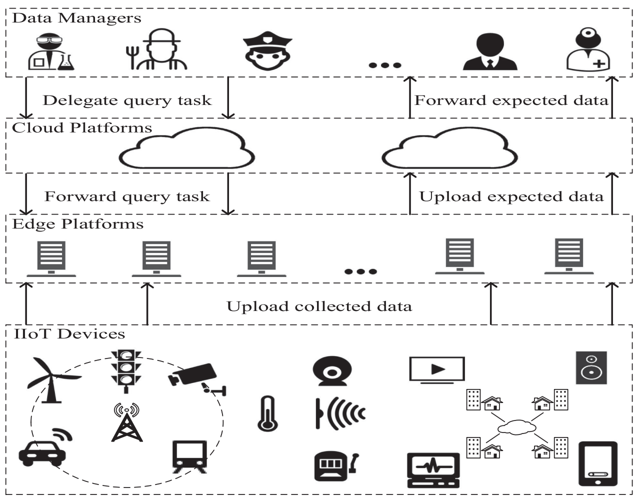

Fig. 1. Classic IIoT scenario under the edge-cloud architecture.

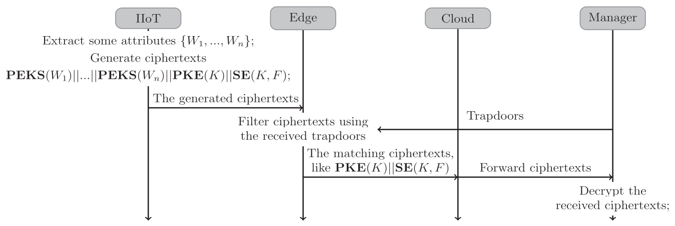
IIoT
Edge
Cloud
Manager
Extract some attributes $\{W_1, ..., W_n\}$;
Generate ciphertexts
$\mathbf{PEKS}(W_1)||...||\mathbf{PEKS}(W_n)||\mathbf{PKE}(K)||\mathbf{SE}(K, F)$;
The generated ciphertexts
Filter ciphertexts using the received trapdoors
Trapdoors
The matching ciphertexts, like $\mathbf{PKE}(K)||\mathbf{SE}(K, F)$
Forward ciphertexts
Decrypt the received ciphertexts;

Fig. 2. Application of PEKS under the edge-cloud architecture for IIoT. The application consists of four objects: the IIoT device (IIoT), the edge device (Edge), the cloud, and the data manager. Let $\mathbf{PEKS}$, $\mathbf{PKE}$, and $\mathbf{SE}$ denote the encryption algorithms of schemes PEKS, PKE, and SE, respectively. Let K be a random symmetric key. Let F be the collected data.

edge platform. Similar to the Cloud, edge platforms can retrieve these data according to the authorization of data managers, such as sorting weather records of a certain week or traffic data under a street name and a time log. Such applications reduce the transmission redundancies and communication latency of the data flow during online services compared with those of the traditional method. Edge platforms supply a win–win strategy to save communication costs by restraining the unnecessary data from uploading to cloud platforms, and they avoid slow data retrieval by IIoT devices or data managers that have low searching efficiency.

However, both cloud and edge platforms are usually assumed to be honest but curious [6]. Although traditional encryption techniques can guarantee the confidentiality of collected data, they make edge platforms hardly able to retrieve data on demand. A major challenge that exists is the dilemma of keeping both data privacy and searchability on edge platforms. Fortunately, we have a technique called public-key encryption with keyword search (PEKS) [7]. PEKS is a potential idea to address the above problem. However, the existing PEKS schemes are still inappropriate for IIoT devices due to their expensive encryption cost. Fig. 2 shows the application of normal PEKS to solve the above problem. In Fig. 2, each IIoT device extracts some attributes of its collected data, takes these attributes as keywords to generate searchable ciphertexts by PEKS, encrypts the collected data by the traditional public-key encryption (PKE) and symmetric-key encryption (SE) schemes (such as the PKE scheme RSA or ElGamal and the SE scheme AES), and sends all generated ciphertexts to a nearby edge platform. In order to retrieve the collected data with the expected attributes, a data manager generates search trapdoors for the chosen attributes and requires a cloud platform to forward the generated trapdoors to all edge platforms. Then, the following steps successively happen. Each edge platform finds the matching ciphertexts in its local storage and uploads these ciphertexts to the cloud platform; then, the cloud platform forwards the uploaded ciphertexts to the manager;and finally, the manager can read the expected data by decrypting the received ciphertexts. Clearly, PEKS allows edge platforms to retrieve IIoT data on demand even if data are encrypted, and a secure PEKS scheme can guarantee data confidentiality under the assumption that both edge and cloud platforms are honest but curious.

In the phase of efficiency, each IIoT device fails at accelerating ciphertext generation in the public-key setting. Most of the processing time is wasted on generating PEKS ciphertexts, since the $\mathbf{PEKS}$ algorithm must be run several times, and each time of running takes much more time than that of algorithms $\mathbf{PKE}$ and $\mathbf{SE}$. Hence, designing a PEKS scheme with lightweight encryption is urgently needed. Referring to previous works [8], [9], traditional protection mechanisms such as cryptographic algorithms and security protocols have proven inefficient as long as devices have several nonnegligible constrains on resources. It becomes more critical when meeting the real-time requirements of IIoT applications such as instant monitoring due to the costly operations in traditional encryption. Moreover, to the best of our knowledge regarding previous work, only [10] has introduced a lightweight searchable PKE (LSPE) to accelerate search performance, of which the encryption is still costly in the worst case. Hence, we are motivated to design a PEKS scheme with both lightweight encryption and fast search to meet the edge-cloud architecture.

We encounter two questions regarding why and how to build an accelerated searchable PKE system for an edge-cloud architectured IIoT. Our previous review on searchable encryption schemes first shows the necessity of PEKS in such a system, as well as the inspiration for our main ideas and contributions.

Our inspiration to design an edge-aided searchable PKE (ESPE) scheme is motivated by the aim to provide an excellent service experience while sharply lowering the burden on computation and communication. In this article, we exploit the capability of edge computing as well as lightweight-designed encryption and fast search. Two phases are processed simultaneously for the same goal. In the first phase, as a contrast to a searchable public-key ciphertexts with hidden structures (SPCHS) scheme [10], which is the first work that retains both semantic security and fast search, ESPE achieves not only lower encryption complexity but also dramatically reduced encryption latency by delegating the costly computation of bilinear mapping to an edge platform. In the second phase, to accelerate search performance, ESPE employs the idea of the hidden structure establishment among searchable ciphertexts in SPCHS. In addition to these high-efficiency expectations, ESPE guarantees that the honest-but-curious edge and cloud platforms cannot learn any information about the encrypted keyword in the sense of standard semantic security.

However, the aforementioned idea also introduces new challenges for the sake of securely delegating bilinear mapping. In SPCHS, generating different ciphertexts may compute bilinear mapping an unequal number of times, i.e., once or twice. Consequently, the honest-but-curious edge platform can break the security since it is easy to distinguish two independent ciphertexts with unequal times of delegated bilinear mappings if we directly transform SPCHS to ESPE. To address this problem, ESPE is elegantly designed to guarantee that every ciphertext just needs to delegate bilinear mapping only once.

Our contributions are as follows. We first define ESPE and its security. The ESPE model defines five main algorithms including the specified edge-aided encryption algorithm. The ESPE security definition extends the original security definition of SPCHS by additionally allowing any adversary to know all transferred information when delegating bilinear mapping. We construct an ESPE instance and prove its security based on a weak decisional bilinear Diffie–Hellman (DBDH) assumption [13]. The ESPE instance achieves the uniform delegation procedures to generate each ciphertext, and meanwhile, constructs hidden structures among ciphertexts to accelerate search performance. Finally, we experimentally compare ESPE with other works in terms of encryption cost and test its performance in a pollution monitoring application under the edge-cloud architecture. The results show that ESPE can save approximately 70% of the processing time for an IIoT device to generate one ciphertext, thereby saving the communication and decryption costs significantly in that application.

Section III models ESPE and its security. An ESPE instance is constructed in Section IV, and its security is proven in Section V. Section VI experimentally shows the practicality of ESPE. Section VII concludes this article.

II. Related Works

The costly encryption performance of PEKS is reminiscent of the high encryption performance of searchable symmetric-key encryption (SSE) [11]. However, when applying SSE under the edge-cloud architecture, all IIoT devices must share different symmetric keys with their data managers. Otherwise, a compromised IIoT device will leak the collected data of other IIoT devices. When there are large numbers of IIoT devices, there is a dramatically increased burden to any data manager to dynamically generate and revoke symmetric keys under SSE. Unfortunately, SSE multiplies the number of search trapdoors needed to meet multiple independent IIoT devices when PEKS needs only one for the same case. Consequently, the communication cost of the whole system to transfer those trapdoors increases as well. In contrast, IIoT devices never share secrets with their managers under PEKS, not to mention symmetric keys. Hence, PEKS cannot be substituted with SSE under the edge-cloud architecture. Moreover, all PEKS trapdoors are independent of IIoT devices.

Our main challenge in this article is to design a PEKS scheme with lightweight encryption. Hence, some related works on designing lightweight encryption will be introduced in the following. To our best knowledge, there are three kinds of ideas to design lightweight encryption. These ideas are heuristics in term of our article. However, only one of them is appropriate to be introduced in our article.

The first idea is to replace the costly cryptographic operations by some lightweight operations. This idea is frequently used in designing lightweight SE [14]. In contrast, it is much harder to be used in designing lightweight PEKS. Most of the PEKS schemes are constructed based on bilinear mapping. This operation has been widely recognized as the most costly cryptographic operation [15]. To date, no lightweight operation can replace it while keeping the same algebraic features. In addition, very few PEKS schemes are constructed without bilinear mapping, such as the PEKS schemes based on quadratic residuosity or lattice algebra [16]–[19]. However, all of them cannot achieve fast searching performance over ciphertexts, i.e., the sublinear search complexity, and many random numbers are chosen in their encryption algorithms. Our experiment shows that choosing numerous random numbers introduces equal or even high complexity compared with that of bilinear mapping.

The second idea is to divide a complete encryption into two parts that are, respectively, run offline and online. The offline part can be precomputed without plaintexts. Therefore, as the needed data arrive, the online part takes the results of the offline part as the input to finish the remainder of the encryption processes. Such kinds of lightweight encryption are called online/offline encryption [20]. Generally, the processor of the offline part might possibly be data producers, a trusted third party, or two noncollusive and honest-but-curious third parties. With the first processor that is also the data producer, the total time spent on encryption is not shortened. Further, the energy consumption is not reduced as well if the producer is an IIoT device. The first processor shows advantages only in employing the idle intervals of data producers to run the offline parts. The remaining two processors with semantic security need some stronger assumptions that are not easily realized.

The third processor, as a third-party idea, will be delegated with the costly cryptographic operations as long as there is encryption of certain required data. This idea is concerned with the assumptions on the delegated third party. In previous works [23], the third party is assumed either honest but curious or untrusted. The untrusted assumption on the third party means that it can return wrong or malicious results to delegators, which will be useful to design an encryption scheme with strong security. However, the satisfied schemes will be impractical under the edge-cloud architecture. In other words, the total delay caused by delegating a costly cryptographic operation to an untrusted outsourced platform, including the time cost to verify the results, is usually longer than that of running the operation locally. In such cases, devices need expensive precomputation or verification processes. Thus, there is no advantage for involving outsourced

TABLE I
NOTIONS FOR ESPE MODELS

Symbols	Notions
Z_q^*	The multiplicative group of integers modulo prime q
N	The set of natural numbers
$\hat{e}$	Bilinear mapping
1^k	The k bits security parameter
K	A randomly chosen symmetric key
$(\mathcal{PK}, \mathcal{SK})$	A pair of master public-and-private keys
$\mathcal{W}$	The keyword space
W_i	The ith keyword in space $\mathcal{W}$
$(\mathcal{PUB}, \mathcal{PRI})$	A pair of public-and-private parts of a hidden structure
$\mathcal{C}$	The set of keyword-searchable ciphertexts
C_i	The ith keyword searchable ciphertext in $\mathcal{C}$
T_W	A keyword-search trapdoor generated from $\mathcal{SK}$ and W

platforms. In contrast, involving the honest-but-curious assumption on the third party is more practical to balance security and practicality. Hence, our article prefers the idea to delegate the costly cryptographic operations to an honest-but-curious third party. The proposed secure data storage and searching framework in [24] gives us intuitive insight regarding our target under the edge-cloud architecture. It is also reported that in the related security analysis, no provable security was declared to support the availability of the framework yet.

III. Modeling ESPE and Its Application

This section models ESPE with its main algorithms, shows the application of ESPE under the edge-cloud architecture, analyzes the suffered attacks of ESPE, and finally, defines the security of ESPE.

Preliminarily, we introduce the main algorithms of ESPE. Most notions involved are listed in Table I.

Definition 1 (ESPE): An ESPE scheme consists of the following five algorithms.

1) Algorithm $\textbf{SystemSetup}(1^k, \mathcal{W})$ takes a security parameter and a keyword space $\mathcal{W}$ as inputs and probabilistically generates a pair of master public-and-private keys $(\mathcal{PK}, \mathcal{SK})$.
2) Algorithm $\textbf{StructureSetup}(\mathcal{PK})$ takes the master public key $\mathcal{PK}$ as input and probabilistically initializes a hidden structure by outputting a pair of public-and-private parts $(\mathcal{PUB}, \mathcal{PRI})$ of the structure.
3) Algorithm $\textbf{Encryption}(\mathcal{PK}, W, \mathcal{PRI})$ takes the master public key $\mathcal{PK}$, a keyword W, and the private part $\mathcal{PRI}$ of a hidden structure as inputs, delegates the expected cryptographic operations to an edge platform, and utilizes the responses of the edge platform to generate a keyword-searchable ciphertext C.
4) Algorithm $\textbf{Trapdoor}(\mathcal{SK}, W)$ takes the master private key $\mathcal{SK}$ and a keyword W as inputs and generates a keyword-search trapdoor T_W.
5) Algorithm $\textbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ takes the master public key $\mathcal{PK}$, the set $\mathcal{C}$ of all keyword-searchable ciphertexts, a keyword-search trapdoor T_W of keyword W, and the public part $\mathcal{PUB}$ of a hidden structure as inputs and finds all matching ciphertexts of keyword W with the hidden structure's public part $\mathcal{PUB}$ from $\mathcal{C}$.

In addition, an ESPE scheme must be consistent in the sense that given the keyword-search trapdoor T_W of any keyword W and the public part $\mathcal{PUB}$ of any hidden structure, Algorithm $\textbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ always finds all matching ciphertexts of W with $\mathcal{PUB}$, except with a negligible probability.

Generally, ESPE has a similar structure to that of the previous work SPCHS except for the employed encryption algorithms in SPCHS. SPCHS seems to be a heuristic framework of ESPE with inseparable encryption algorithms, which is also the biggest challenge before ESPE. In particular, ESPE defines an interactive encryption algorithm, which can delegate the compute-intensive cryptographic operations to an edge platform and then, use the corresponding response to generate a ciphertext. In contrast, SPCHS defines its encryption algorithm as a noninteractive one, which indicates the failure of SPCHS in our scenarios under the edge-cloud architecture.

As illustrated by Fig. 3, the tasks for the four objects in an IIoT application are changed comparing to that of PEKS. In the communication phase, when an IIoT device collects data and wishes to update data to the cloud, they first cache data to a nearby edge device. Then, the edge device relays historical data to the cloud. A data manager, who is authorized to retrieve the data, will send requests to the cloud; then, the cloud returns the data according to the requests. In the security phase, the outsourced data are searchable encrypted locally by the IIoT devices, when the searchablity is completed by the edge device. The cloud and the edge device response to the searching requests, respectively, on historical data and instant data. The corresponding ciphertexts finally will be delivered to the data manager by the cloud and decrypted out. All data flows are illuminated in Fig. 3.

The application performs in the order from the setup phase to the search phase. The operations on the vertical timelines below the IIoT devices, the edge devices, the cloud, and the data manager are implemented by sequence. The horizontal arrow lines demonstrate the information transmission. In the application, suppose that all IIoT devices know their managers' public keys for running Algorithm $\textbf{PKE}$. The application consists of three phases.

1) *Setup phase:* A manager generates a pair of master public-and-private keys $(\mathcal{PK}, \mathcal{SK})$ by running Algorithm $\textbf{SystemSetup}(1^k, \mathcal{W})$ and stores $\mathcal{PK}$ in his IIoT devices; each IIoT device runs Algorithm $\textbf{StructureSetup}(\mathcal{PK})$ to initialize its hidden structure by generating a pair of public-and-private parts $(\mathcal{PUB}, \mathcal{PRI})$ and uploads $\mathcal{PUB}$ to its nearest edge platform.
2) *Ciphertext generation phase:* Let F be the collected data of an IIoT device. The IIoT device extracts some attributes $\{W_1, \ldots, W_n\}$ from F, runs Algorithm $\textbf{Encryption}(\mathcal{PK}, W_i, \mathcal{PRI})$ for those attributes to generate searchable ciphertexts $\{C_i | i \in [1, n]\}$ with the assistance of the edge platform, encrypts F as $C_F = \textbf{PKE}(K) || \textbf{SE}(K, F)$ where K is a randomly chosen symmetric key, and finally, uploads $C_1 || \ldots || C_n || C_F$ to the closest edge platform.
3) *Search Phase:* To access the date of attribute W, a manager runs Algorithm $\textbf{Trapdoor}(\mathcal{SK}, W)$ to generate a search trapdoor T_W and sends it to all edge platforms via a cloud platform; once T_W copies are received, each edge platform runs $\textbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ to find

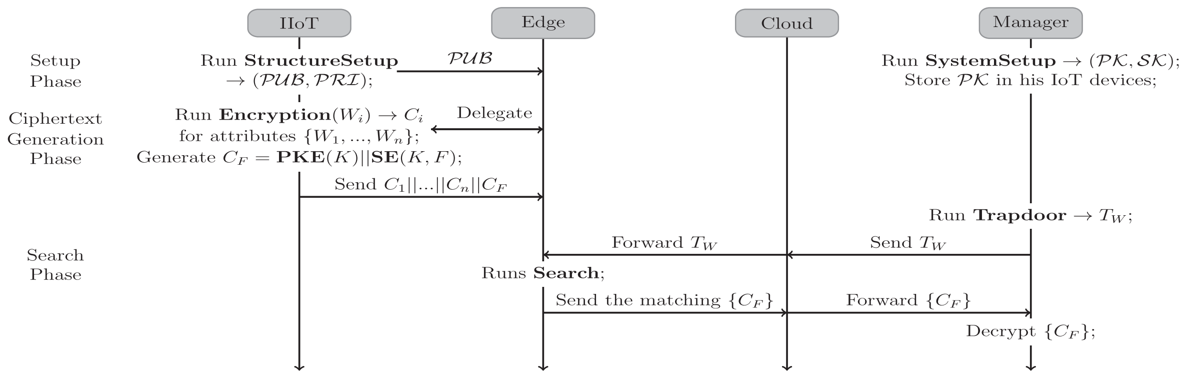
IIoT
Edge
Cloud
Manager
Setup Phase
Run **StructureSetup** $\rightarrow (\mathcal{PUB}, \mathcal{PRI})$;
$\mathcal{PUB}$
Run **SystemSetup** $\rightarrow (\mathcal{PK}, \mathcal{SK})$; Store $\mathcal{PK}$ in his IoT devices;
Ciphertext Generation Phase
Run **Encryption**$(W_i) \rightarrow C_i$ for attributes $\{W_1, ..., W_n\}$;
Delegate
Generate $C_F = \mathbf{PKE}(K)||\mathbf{SE}(K, F)$;
Send $C_1||...||C_n||C_F$
Run **Trapdoor** $\rightarrow T_W$;
Search Phase
Forward T_W
Send T_W
Runs **Search**;
Send the matching $\{C_F\}$
Forward $\{C_F\}$
Decrypt $\{C_F\}$;

Fig. 3. Application of ESPE under the edge-cloud architecture.

the matching searchable ciphertexts in its local storage and sends the PKE and SE ciphertexts that are both appended to these matching ciphertexts to the manager via the cloud platform; finally, the manager decrypts the received ciphertexts.

In the abovementioned application, ESPE resists attacks from the honest-but-curious edge and cloud platforms, the compromised IIoT devices, and eavesdroppers. The previous work SPCHS also suffers the same attacks as ESPE except the attacks from the honest-but-curious edge platform. Each edge platform is mainly used to achieve the delegated cryptographic operations from IIoT devices, and then, IIoT devices take the responses from their corresponding edge platforms as auxiliary values to generate keyword-searchable ciphertexts. Hence, edge platforms know not only the generated ciphertexts but also some important values that are used to generate these ciphertexts. In contrast, cloud platforms only know the generated ciphertext. Hence, a malicious edge platform can launch more serious attacks than a malicious cloud platform can.

ESPE resists the same attacks as does SPCHS, implying that ESPE has the same security definition as SPCHS to a certain degree. Furthermore, their major differences in terms of the security definition concern how to generate keyword-searchable ciphertexts, such as the place to generate the challenge keyword-searchable ciphertext. Specifically, the security of ESPE is defined as the indistinguishability under chosen keyword-and-structure attacks (IND-CKSA). It allows a probabilistic polynomial-time (PPT) adversary to query the keyword-search trapdoors of the expected keywords and the private parts of the expected hidden structures as defined by SPCHS. When the adversary queries any keyword-searchable ciphertext (including the challenge one) with SPCHS, the difference in semantic security between SPCHS and ESPE shows that with ESPE, the IND-CKSA security allows the adversary to know not only the expected ciphertext but also the corresponding state information. Generally, the state information of a keyword-searchable ciphertext includes all observable information of the whole delegation process during the generation of the ciphertext, which means that ESPE faces more critical challenges. The IND-CKSA definition of ESPE is as follows.

Definition 2 (IND-CKSA): Suppose there are at most $N \in \mathbb{N}$ hidden structures. An ESPE scheme is IND-CKSA secure that if any PPT adversary $\mathcal{A}$ has only a negligible advantage $Adv^{\text{IND-CKSA}}_{ESPE,\mathcal{A}}$ to win in the following IND-CKSA game.

1) *Setup phase:* This phase is used to set up an ESPE scheme. A challenger runs Algorithm $\mathbf{SystemSetup}$ to generate a pair of master public-and-secret keys $(\mathcal{PK}, \mathcal{SK})$, runs Algorithm $\mathbf{StructureSetup}$ N times to generate N hidden structures, and finally, sends $\mathcal{PK}$ and $\mathbf{PUB}$ to $\mathcal{A}$, where $\mathbf{PUB}$ is the set of the N hidden structures' public parts.
2) *Query phase 1:* This phase allows $\mathcal{A}$ to adaptively issue the following queries multiple times to the challenger, where the Trapdoor query simulates the attacks from the honest-but-curious cloud platform, the Privacy query simulates the attacks from the compromised IIoT devices, and the Encryption query simulates the attacks from both the honest-but-curious edge platform and eavesdroppers. The details are as follows.
   a) Trapdoor query $\mathcal{Q}_{\text{Trap}}(W)$: Given an issued keyword W, the challenger returns the corresponding keyword search trapdoor T_W.
   b) Privacy query $\mathcal{Q}_{\text{Pri}}(\mathcal{PUB})$: Given an issued hidden structure's public part $\mathcal{PUB} \in \mathbf{PUB}$, the challenger returns the corresponding private part $\mathcal{PRI}$.
   c) Encryption query $\mathcal{Q}_{\text{Enc}}(W, \mathcal{PUB})$: Given an issued keyword W and an issued hidden structure's public part $\mathcal{PUB} \in \mathbf{PUB}$ where $\mathcal{A}$ has never taken $\mathcal{PUB}$ as input to query $\mathcal{Q}_{\text{Pri}}(\mathcal{PUB})$ before,[1] the challenger returns the expected keyword-searchable ciphertext C and its state information $\mathcal{ST}$.
3) *Challenge phase:* $\mathcal{A}$ chooses and sends two challenge keyword-and-structure pairs $(W_0^*, \mathcal{PUB}_0^*)$ and $(W_1^*, \mathcal{PUB}_1^*)$ to the challenger, where both $\mathcal{PUB}_0^*$ and $\mathcal{PUB}_1^*$ belong to $\mathbf{PUB}$; the challenger randomly chooses

[1]If the adversary $\mathcal{A}$ has queried the $\mathcal{Q}_{\text{Pri}}(\mathcal{PUB})$, he can generate the expected keyword-searchable ciphertext of the keyword W with the public part $\mathcal{PUB}$ by himself. Hence, the adversary does not need to issue the encryption query in this case.

$d \in \{0,1\}$, generates the challenge ciphertext C_d^* of keyword W_d^* with the hidden structure $\mathcal{PUB}_d^*$, and sends C_d^* and its state information $\mathcal{ST}_d^*$ to $\mathcal{A}$.

4) *Query phase 2:* This phase is the same as Query phase 1. Note that in both Query phase 1 and Query phase 2, $\mathcal{A}$ cannot query the corresponding private parts of both $\mathcal{PUB}_0^*$ and $\mathcal{PUB}_1^*$ and the keyword search trapdoors of both W_0^* and W_1^*.

5) *Guess phase:* $\mathcal{A}$ sends a guess $d' \in \{0,1\}$ to the challenger. We say that $\mathcal{A}$ wins if $d' = d$. In addition, let $\mathrm{Adv}_{\mathrm{ESPE},\mathcal{A}}^{\mathrm{IND\text{-}CKSA}} = |Pr[d' = d] - Pr[d' \neq d]|$ be the advantage of $\mathcal{A}$ to win in the above game.

In practice, an IND-CKSA secure ESPE scheme means that: 1) all ESPE ciphertexts can keep the privacy of their keywords as well as hidden structures for someone who does not know any keyword-search trapdoor or any hidden structure's private part and 2) given a keyword-search trapdoor or a hidden structure's private part, one can only infer which ciphertexts have the same keyword with the given trapdoor or which ciphertexts are related to the given private part. In addition, the IND-CKSA definition does not consider keyword guessing attacks, since the previous work [23] proposed a general solution for PEKS including ESPE against such attacks. Guessing attack is not considered in this work although how to construct a lightweight design based on the naive combination is still a challenging work. However, we will consider the correlated works in the future.

IV. Instantiating ESPE

This section constructs an ESPE instance in which the encryption algorithm can delegate bilinear mapping to an edge platform and take the response from the edge platform as input to generate a ciphertext. Moreover, the generated ciphertext is semantically secure under the assumption that the edge platform is honest but curious. Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be three multiplicative groups, all with the prime order q. Let g_1 and g_2 be the generators of groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an efficient bilinear mapping. Equation $\hat{e}(g_1^a, g_2^b) = \hat{e}(g_1, g_2)^{ab}$ holds for any $a \in \mathbb{Z}_q^*$ and $b \in \mathbb{Z}_q^*$, and $\hat{e}(g_1, g_2)$ is a generator of group $\mathbb{G}_T$. Let $\mathcal{W} = \{0,1\}^*$ be the keyword space. Our ESPE scheme is constructed as follows.

1) Algorithm $\mathbf{SystemSetup}(1^k, \mathcal{W})$ takes a security parameter 1^k ($k \in \mathbb{N}$) and the keyword space $\mathcal{W}$ as inputs and performs the following step.
   a) Initialize a bilinear mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and choose a generator $g_2 \in \mathbb{G}_2$, where the binary size of prime order q equals k.
   b) Randomly pick $s \in \mathbb{Z}_q^*$ and set $p = g_2^s$.
   c) Pick a cryptographic hash function $\mathbf{H} : \mathcal{W} \to \mathbb{G}_1$.
   d) Return a pair of master public-and-private keys $(\mathcal{PK}, \mathcal{SK})$, where $\mathcal{PK} = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_2, p, \mathbf{H})$ and $\mathcal{SK} = s$.
2) Algorithm $\mathbf{StructureSetup}(\mathcal{PK})$ takes the master public key $\mathcal{PK}$ as input and performs the following steps.
   a) Randomly pick u and r_1 both from $\mathbb{Z}_q^*$ and compute g_2^u and $g_2^{r_1}$.
   b) Initialize a pair of public-and-private parts $(\mathcal{PUB}, \mathcal{PRI})$, where $\mathcal{PUB} = g_2^u$ and $\mathcal{PRI} = (u, r_1, g_2^{r_1})$.

   Note that $\mathcal{PRI}$ is a variable list formed as $(u, r_1, g_2^{r_1}, \{(w, Pt[u,w]) | w \in \mathcal{W}, u \in \mathbb{Z}_q^*, Pt[u,w] \in \mathbb{G}_T\})$, in which $Pt[u,w]$ denotes an element of group $\mathbb{G}_T$.
   c) Algorithm $\mathbf{Encryption}(\mathcal{PK}, W, \mathcal{PRI})$ takes the master public key $\mathcal{PK}$, a keyword W, and the private part $\mathcal{PRI}$ of a hidden structure as inputs, parses $\mathcal{PRI}$ as $(u, r_1, g_2^{r_1})$ and some records such as $(w, Pt[u,w]) \in \mathcal{W} \times \mathbb{G}_T$, and performs the following steps.
      a) Randomly pick $r_2 \in \mathbb{Z}_q^*$ and compute $g_2^{r_1 \cdot r_2}$.
      b) Compute and send $\mathbf{H}(W)^{r_2}$ to an edge platform.
      c) Receive value $\hat{e}(\mathbf{H}(W)^{r_2}, p)$ from the edge platform.
      d) Seek record $(W, Pt[u,W])$ from $\mathcal{PRI}$ according to both W and u.
      e) If existing, generate the keyword-searchable ciphertext $C = (Pt, g_2^{r_1 \cdot r_2})$ and update record $(W, Pt[u,W] = \hat{e}(\mathbf{H}(W)^{r_2}, p)^{r_1})$ into $\mathcal{PRI}$.
      f) Otherwise, generate the keyword-searchable cipertext $C = (\hat{e}(\mathbf{H}(W)^{r_2}, p)^{u/r_2}, g_2^{r_1 \cdot r_2})$ and add $(W, Pt[u,W] = \hat{e}(\mathbf{H}(W)^{r_2}, p)^{r_1})$ into $\mathcal{PRI}$.
4) Algorithm $\mathbf{Trapdoor}(\mathcal{SK}, W)$ takes the master private key $\mathcal{SK}$ and a keyword W as inputs and generates a keyword-search trapdoor $T_W = \mathbf{H}(W)^s$.
5) Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ takes the master public key $\mathcal{PK}$, the set $\mathcal{C}$ of all keyword-searchable ciphertexts, a keyword-search trapdoor T_W, and the public part $\mathcal{PUB}$ of a hidden structure as inputs, initializes an empty set $\mathcal{C}' = \emptyset \subset \mathcal{C}$, and performs the following steps.
   a) Compute $Pt = \hat{e}(T_W, \mathcal{PUB})$.
   b) Seek a keyword-searchable ciphertext $C' = (C_1', C_2') \in \mathcal{C}$ having $C_1' = Pt$.
   c) If existing, add the found ciphertext C' into $\mathcal{C}'$, set $Pt = \hat{e}(T_W, C_2')$, and go to step 2.
   d) Otherwise, return the set $\mathcal{C}'$.

The Correctness Proof of ESPE. Suppose that: 1) $\mathcal{PRI}$ is initialized as $(u, r_1, g_2^{r_1})$ by Algorithm $\mathbf{StructureSetup}(\mathcal{PK})$ and 2) there are, in total, two keyword-searchable ciphertexts (C^1, C^2), which are sequentially generated by running Algorithm $\mathbf{Encryption}(\mathcal{PK}, W, \mathcal{PRI})$ two times. Hence, the ciphertexts C^1 and C^2 can be parsed as $C^1 = (C_1^1 = \hat{e}(\mathbf{H}(W), p)^u, C_2^1 = g_2^{r_1 \cdot r_2})$ and $C^2 = (C_1^2 = \hat{e}(\mathbf{H}(W), p)^{r_1 \cdot r_2}, C_2^2 = g_2^{r_1 \cdot r_3})$, respectively, where r_2 and r_3 are the two random numbers picked from the space $\mathbb{Z}_q^*$ by the double implementations of Algorithm $\mathbf{Encryption}(\mathcal{PK}, W, \mathcal{PRI})$.

To prove the correctness of ESPE, it is without loss of generality to prove that Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ can just find the above-mentioned two ciphertexts C^1 and C^2. According to step 1 of Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$, we have $Pt = \hat{e}(T_W, \mathcal{PUB}) = \hat{e}(\mathbf{H}(W)^s, g_2^u) = \hat{e}(\mathbf{H}(W), p)^u$. The ciphertext C^1 can be found since equation $C_1^1 = Pt$ holds in step 2 of Algorithm $\mathbf{Search}$. Furthermore, we have $Pt = \hat{e}(T_W, C_2^1) = \hat{e}(\mathbf{H}(W)^s, g_2^{r_1 \cdot r_2}) = \hat{e}(\mathbf{H}(W), p)^{r_1 \cdot r_2}$ according to step 3 of Algorithm $\mathbf{Search}$. It is clear that the ciphertext C^2 can be also found by step 2 of Algorithm $\mathbf{Search}$ since equation $C_1^2 = Pt$ holds.

In addition, we need to prove that no more ciphertext can be found by Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ except the matched ones. Without loss of generality, suppose that:

1) this algorithm finds an incorrect keyword-searchable ciphertext C^i;
2) the ciphertext C^i has keyword W' and the public part $\mathcal{PUB}' = g_2^{u'}$;
3) either $W' \neq W$ or $\mathcal{PUB}' \neq \mathcal{PUB}$ holds;
4) the corresponding private part of $\mathcal{PUB}'$ is initialized as $\mathcal{PRI}' = (u', r_1', g_2^{r_1'})$.

We now prove that this assumption is false by the following two cases.

Case 1: The ciphertext C^i is the first ciphertext of both keyword W' and the public part $\mathcal{PUB}'$. In this case, we have that the ciphertext C^i can be parsed as $C^i = (C_1^i = \hat{e}(\mathbf{H}(W'), p)^{u'}, C_2^i)$, and one of the following three equations holds if Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ can find the ciphertext C^i. The three equations are $\hat{e}(\mathbf{H}(W'), p)^{u'} = \hat{e}(\mathbf{H}(W), p)^{u}$, $\hat{e}(\mathbf{H}(W'), p)^{u'} = \hat{e}(\mathbf{H}(W), p)^{r_1 \cdot r_2}$, and $\hat{e}(\mathbf{H}(W'), p)^{u'} = \hat{e}(\mathbf{H}(W), p)^{r_1 \cdot r_3}$. It is clear that none of the above equations hold in practice except with a negligible probability.

Case 2: The ciphertext C^i is not the first ciphertext of both keyword W' and the public part $\mathcal{PUB}'$. This case implies that the ciphertext C^i can be parsed as $C^i = (C_1^i = \hat{e}(\mathbf{H}(W'), p)^{r_1' \cdot r_2'}, C_2^i)$ where $r_2' \in \mathbb{Z}_q^*$ is randomly chosen, and one of the following three equations holds if Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ can find the ciphertext C^i. The three equations are $\hat{e}(\mathbf{H}(W'), p)^{r_1' \cdot r_2'} = \hat{e}(\mathbf{H}(W), p)^{u}$, $\hat{e}(\mathbf{H}(W'), p)^{r_1' \cdot r_2'} = \hat{e}(\mathbf{H}(W), p)^{r_1 \cdot r_2}$, and $\hat{e}(\mathbf{H}(W'), p)^{r_1' \cdot r_2'} = \hat{e}(\mathbf{H}(W), p)^{r_1 \cdot r_3}$. It is also clear that none of the above equations hold in practice except with a negligible probability.

In the above statements, we simplify the proof by supposing that there are only two matching ciphertexts C^1 and C^2. Indeed, Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ is correct for any given number of matching ciphertexts, since all the other matching ciphertexts can be found using the same method to find C^2. Moreover, it is easy to extend the above-mentioned Case 2 to prove that no incorrect ciphertext can be found even if there are more than two matching ciphertexts. In summary, Algorithm $\mathbf{Search}(\mathcal{PK}, \mathcal{C}, T_W, \mathcal{PUB})$ can only find all matching keyword-searchable ciphertexts of keyword W with the public part $\mathcal{PUB}$ except with a negligible probability in practice.

V. Proving the IND-CKSA Security of ESPE

Generally, bilinear mapping $\hat{e}$ in ESPE can be instantiated as per three different types in practice. The corresponding security proofs have the same essence with only minor differences in detail. Hence, we will only prove that the ESPE instance based on the type-3 bilinear mapping [13] is IND-CKSA secure. Before the proof, it is conventional to introduce a mathematical hardness assumption.

Definition 3 (The Weak DBDH Assumption in Type 3 [13]): Given the public parameters $(q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, \hat{e}, g_1, g_2)$ of a type-3 bilinear mapping, the weak DBDH problem in type-3 is defined as the advantage of any PPT Algorithm $\mathcal{B}$ to distinguish between tuples $(g_1^a, g_2^a, g_2^b, g_1^c, \hat{e}(g_1, g_2)^{abc})$ and $(g_1^a, g_2^a, g_2^b, g_1^c, Y)$, where (a, b, c) are randomly chosen in $\mathbb{Z}_q^*$, Y is randomly chosen in $\mathbb{G}_T$, and $\hat{e}$ is a type-3 one. Let $Adv_{\mathcal{B}}^{DBDH}(1^k) = |Pr[\mathcal{B}(g_1^a, g_2^a, g_2^b, g_1^c, \hat{e}(g_1, g_2)^{abc}) = 1] - Pr[\mathcal{B}(g_1^a, g_2^a, g_2^b, g_1^c, Y) = 1]|$ be the advantage of Algorithm $\mathcal{B}$ to solve the above-mentioned DBDH problem, where k equals the binary length of q. We say that the weak DBDH assumption in type-3 holds if $Adv_{\mathcal{B}}^{\text{DBDH}}(1^k)$ is negligible in the parameter k.

Theorem 1 proves that the IND-CKSA security of ESPE is based on the weak DBDH assumption in type-3 (due to page limitations, please see Supplemental Materials for the proof details). This theorem implies that since the weak DBDH assumption in type-3 holds in practice, no PPT adversary can break the IND-CKSA security of ESPE with the nonnegligible advantage.

Theorem 1: Let the hash function $\mathbf{H}$ of ESPE be modeled as a random oracle $\mathcal{Q}_H$. Suppose that: 1) there are at most $N \in \mathbb{N}$ hidden structures; 2) there is a PPT adversary $\mathcal{A}$ who can break the IND-CKSA security of ESPE with the advantage $Adv_{\text{ESPE},\mathcal{A}}^{\text{IND-CKSA}}$; and 3) $\mathcal{A}$ issues the trapdoor query $\mathcal{Q}_{\text{Trap}}$ at most q_t times and the private query $\mathcal{Q}_{\text{Pri}}$ at most q_p times. Then, there is a PPT Algorithm $\mathcal{B}$ that solves the weak DBDH problem in type-3 with the advantage of approximately $\text{Adv}_{\mathcal{B}}^{\text{DBDH}}(1^k) \approx \frac{128}{e^4 \cdot (q_t + q_p)^4} \cdot \text{Adv}_{\text{ESPE},\mathcal{A}}^{\text{IND-CKSA}}$, where e is the base of the natural logarithms.

VI. Testing ESPE

This section experimentally shows that ESPE has significant advantages in saving the encryption cost of IIoT devices and keeping the advantages of the edge-cloud architecture even in the ciphertext setting.

A. Reduction of Approximately 70% Encryption Cost

1) Experimental Environment: We choose a bilinear-mapping-friendly elliptic curve [25] from plenty of candidates. This curve has faster exponentiation operations of both groups $\mathbb{G}_1$ and $\mathbb{G}_2$ than those of other curves, causing ESPE to have the same security level as that of the widely used AES-128. We use the pairing-based cryptography (PBC) library to code the chosen elliptic curve and ESPE, while a Raspberry Pi device (a single-board computer with wireless LAN and Bluetooth connectivity) is used as an IIoT device and a laptop as an edge platform. The IIoT device is directly connected to the laptop via WiFi. More notions are shown in Table II.

2) Experimental Results: To show the advantage of ESPE via the encryption cost saving, we compare ESPE with two other schemes. One is the scheme of ESPE without delegation. It means that the bilinear mapping operation is not delegated to the edge platform in ESPE. The other is the original scheme SPCHS, which does not ever take lightweight encryption into account. Table III shows the experimental results. The time cost of IIoT devices in ESPE is approximately 68.19% and 78.66% less time, respectively, compared with that of the ESPE scheme without delegation and the original SPCHS to generate one ciphertext. Considering only the time cost of the IIoT device per se, ESPE reduces the time by 75.91% and 83.84% compared with that of the two aforementioned schemes, respectively.

TABLE II
EXPERIMENTAL ENVIRONMENT

2*Edge platform	Intel Dual-Core I5-4200M CPU @ 2.50 GHz, 8 GB of RAM, Ubuntu 16.04, and Gcc5.4.0
2*IIoT device	RASPBERRY PI 3 MODEL B 1 GB of RAM, Raspbian, and Gcc 6.3.0
Elliptic curve with embedding degree 12 [26] (Unit: Decimal)	
Equation	$y^2 = x^3 + b$
2*Base field	260532200783961536561853044153738822596223089371557168731494664303638395082391
2*Group order	260532200783961536561853044153738822595712665821175760941446444365476223949041
2*b	102741906334308373669282955219530608470979838544970689833470880386791420013557
2*Beta	102741906334308373669282955219530608470979838544970689833470880386791420013557
2*Alpha0	202674882470337347060959516991291483807567385593608563803865379340605598986836
2*Alpha1	227648511008665453826529691521323839881674239419389154433707706469782156619760

TABLE III
TIME COST TO GENERATE ONE CIPHERTEXT

Scheme	The IIoT device	The edge platform
SPCHS	798.06 ms	0 ms
ESPE without delegation	535.38 ms	0 ms
ESPE	128.99 ms	41.33 ms

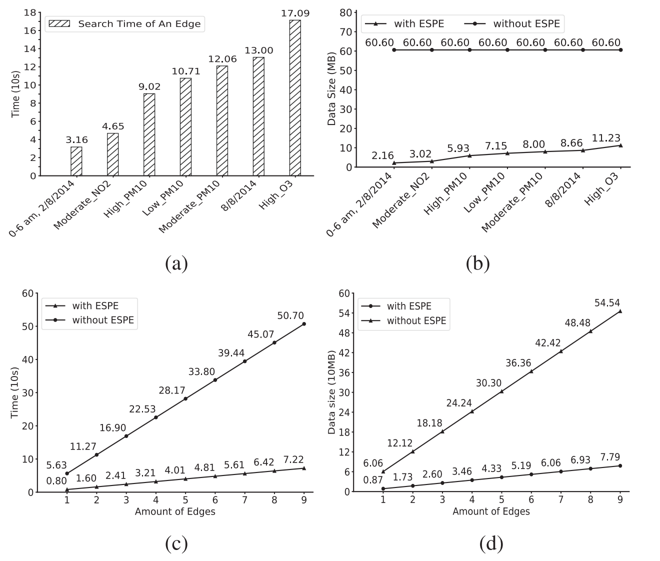

Fig. 4. Comparisons of system performances between implementing ESPE and traditionally symmetric encryption approach (named with/without ESPE). (a) Search cost. (b) Average communication cost. (c) Decryption cost. (d) Total communication cost.

B. Case Study of ESPE Under the Edge-Cloud Architecture

1) Case Instruction: Suppose ESPE is applied in a secure pollution monitoring system under the edge-cloud architecture. In it, each edge platform serves 20 IIoT sensors via the WiFi network. Each sensor detects five types of pollutants, which are *ozone*, *particulate matter*, *carbon monoxide*, *sulphur dioxide*, and *nitrogen dioxide*, and the detection period is five minutes. All pollution records are introduced by the CityPulse project [28]. With the pollution records, the sensors take the detection time, device location, and pollution levels of the aforementioned five pollutants as keywords to generate ESPE ciphertexts, where the pollution levels refer to the standards of the U.K. [27]. The pollution records are encrypted by an elliptic-curve-based ElGamal with a 256-bit group order and AES-128. IIoT devices upload all ciphertexts to the corresponding edge platforms. To query the pollution records of the expected keywords, such as keyword *High_O3*, an environment manager generates the corresponding search trapdoor by ESPE and sends it to all edge platforms via the cloud platform. Each edge platform finds the matching ciphertexts in its local storage by ESPE and sends them to the manager also via the cloud platform. Finally, the manager decrypts the received ciphertexts and obtains the queried records. Each edge platform just stores the ciphertexts generated in the last week and provides the related search services since the historical data are infrequently queried. In practice, the historical data can be uploaded to the cloud platform by the edge platform, and the search services on the historical data can be queried via the cloud platform.

2) Numerical Results: In the resulting system, each edge platform stores approximately 40 320 encrypted records and 282 240 ESPE ciphertexts. Suppose that each keyword has the same distribution in all edge platforms. Referring to Table III, Fig. 4(a) shows the search performance of an edge platform. For example, suppose that the environment protector wants to query the pollution records of keyword *High_O3*; each edge platform takes approximately 170 s to find all matching ciphertexts. In practice, the search performance can be significantly improved with more CPU cores on the edge platforms since ESPE supports parallel search.

To show the advantages of ESPE under the edge-cloud architecture, we test the performance of the pollution monitoring system with ESPE and without ESPE, respectively. In the system without ESPE, all edge platforms will send the entire ciphertext database to the environment protector, and the protector will decrypt out all plaintexts and subsequently, find the expected records. Fig. 4(b) shows that ESPE saves approximately 81.5%–96.4% of the communication cost of one edge platform when replying ciphertexts to the protector. Moreover, following the increasing of the edge amount, ESPE achieves a linearly enhanced advantage on decryption and communication, as shown in Fig. 4(c) and (d). Fig. 4(b) is the average cost of each record and Fig. 4(d) is the total cost of all records. Apparently, the advantages are not noticeable for single records; but for the big data records, the advantages of ESPE are tremendous on communication.

VII. CONCLUSION

In this article, a dilemma that happened often in the modern industry that IIoT devices need both lowered-cost and high-security requirements when retrieving private data under the edge-cloud architecture was discussed. The lightweight designed ESPE employed outsourcing PEKS and hidden structures in its models, which are promising to accelerate encryption and searching according to our analyses. It also showed the possibility that IIoT devices are free to delegate partial encryption, especially the most costly operations to the curious third party so as to save their energy and storage, which is critical. We also consider it as a novel method to build security guarantee among the trust-lacked objects in the IIoT. Summarily, ESPE under the edge-cloud architecture is over and above the given presentation as a milestone of exploring the highest capability of outsourcing secure data in the IIoT.

References

[1] L. Yang, “Industry 4.0: A survey on technologies, applications and open research issues,” *J. Ind. Inf. Integr.*, vol. 6, pp. 1–10, 2017.

[2] M. Chiang and T. Zhang, “Fog and IoT: An overview of research opportunities,” *IEEE Internet Things J.*, vol. 3, no. 6, pp. 854–864, Dec. 2016.

[3] E. Elmroth, P. Leitner, S. Schulte, and S. Venugopal, “Connecting fog and cloud computing,” *IEEE Cloud Comput.*, vol. 4, no. 2, pp. 22–25, Mar./Apr. 2017.

[4] X. Masip-Bruin, E. Marin-Tordera, G. Tashakor, A. Jukan, and G. J. Ren, “Foggy clouds and cloudy fogs: A real need for coordinated management of fog-to-cloud computing systems,” *IEEE Wireless Commun.*, vol. 23, no. 5, pp. 120–128, Oct. 2016.

[5] H. Teruo, Y. Hirozumi, H. Akihito, U. Akira, and Y. Keiichi, “Edge computing and IoT based research for building safe smart cities resistant to disasters,” in *Proc. IEEE 37th Int. Conf. Distrib. Comput. Syst.*, 2017, pp. 1729–1737.

[6] J. Zhang, B. Chen, Y. Zhao, X. Cheng, and F. Hu, “Data security and privacy-preserving in edge computing paradigm: Survey and open issues,” *IEEE Access*, vol. 6, pp. 18209–18237, 2018.

[7] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in *Proc. Int. Conf. Theory Appl. Cryptographic Techn.*, 2004, vol. 3027, pp. 506–522.

[8] A.-R. Sadeghi, C. Wachsmann, and M. Waidner, “Security and privacy challenges in industrial Internet of Things,” in *Proc. 52nd Des. Autom. Conf.*, 2015, pp. 54:1–54:6.

[9] S. Pinto, T. Gomes, J. Pereira, J. Cabral, and A. Tavares, “IIoTEED: An enhanced, trusted execution environment for industrial IoT edge devices,” *IEEE Internet Comput.*, vol. 21, no. 1, pp. 40–47, Jan./Feb. 2017.

[10] P. Xu, S. He, W. Wang, W. Susilo, and H. Jin, “Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks,” *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3712–3723, Aug. 2018.

[11] S. Kamara, C. Papamanthou, and T. Roeder, “Dynamic searchable symmetric encryption,” in *Proc. ACM Conf. Comput. Commun. Secur.*, 2012, pp. 965–976.

[12] P. Xu, Q. Wu, W. Wang, W. Susilo, J. Domingo-Ferrer, and H. Jin, “Generating searchable public-key ciphertexts with hidden structures for fast keyword Search,” *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 9, pp. 1993–2006, Sep. 2015.

[13] S. Chatterjee and A. Menezes, “On cryptographic protocols employing asymmetric pairings—The role of ψ revisited,” *Discrete Appl. Math.*, vol. 159, no. 13, pp. 1311–1322, 2011.

[14] S. Banik, S. K. Pandey, T. Peyrin, Y. Sasaki, S. M. Sim, and Y. Todo, “GIFT: A small present—Towards reaching the limit of lightweight encryption,” in *Proc. Int. Conf. Cryptographic Hardware Embedded Syst.* 2017, pp. 321–345.

[15] J. Shao and Z. Cao, “CCA-Secure proxy re-encryption without pairings,” in *Proc. Int. Workshop Public Key Cryptography*, 2009, pp. 357–376.

[16] R. Behnia, M. O. Ozmen and A. A. Yavuz. “Lattice-based public key searchable encryption from experimental perspectives,” *IEEE Trans. Dependable and Secure Comput.*, doi: 10.1109/TDSC.2018.2867462.

[17] V. Kuchta and O. Markowitch, “Identity-based threshold encryption on lattices with application to searchable encryption,” in *Proc. Int. Conf. Appl. Techn. Inf. Secur.*, 2016, pp. 117–129.

[18] Y. Yang and M. Ma, “Semantic searchable encryption scheme based on lattice in quantum-era,” *J. Inf. Sci. Eng.*, vol. 32, no. 2, pp. 425–438, 2016.

[19] G. D. Crescenzo and V. Saraswat, “Public key encryption with searchable keywords based on Jacobi symbols,” in *Proc. Int. Conf. Cryptology India*, 2007, pp. 282–296.

[20] W. Liu, J. Liu, Q. Wu, B. Qin, and K. Liang, “Online offline public-index predicate encryption for fine-grained mobile access control,” in *Proc. Eur. Symp. Res. Comput. Secur.*, 2016, vol. 9879, pp. 588–605.

[21] H. Tian, F. Zhang, and K. Ren, “Secure bilinear pairing outsourcing made more efficient and flexible,” in *Proc. 10th ACM Symp. Inf., Comput. Commun. Secur.*, 2015, pp. 417–426.

[22] S. Canard, J. Devigne, and O. Sanders, “Delegating a pairing can be both secure and efficient,” in *Proc. Int. Conf. Appl. Cryptography Netw. Secur.*, 2014, pp. 549–565.

[23] P. Xu, H. Jin, Q. Wu, and W. Wang, “Public-key encryption with fuzzy keyword search: A provably secure scheme under keyword guessing attack,” *IEEE Trans. Comput.*, vol. 62, no. 11, pp. 2266–2277, Nov. 2013.

[24] J. Fu, Y. Liu, H. Chao, B. K. Bhargava, and Z. Zhang, “Secure data storage and searching for industrial IoT by integrating fog computing and cloud computing,” *IEEE Trans. Ind. Informat.*, vol. 14, no. 10, pp. 4519–4528, Oct. 2018.

[25] P. S. L. M. Barreto and M. Naehrig, “Pairing-friendly elliptic curves of prime order,” in *Proc. Sel. Areas Cryptography*, 2005, vol. 3897, pp. 319–331.

[26] B. Lynn, “PBC Library.” Accessed: Jul. 19, 2018. [Online]. Available: https://crypto.stanford.edu/pbc/manual/ch08s08.html

[27] U.K. DEFRA, “What is the Daily Air Quality Index?” Accessed: Jul. 20, 2018. [Online]. Available: https://uk-air.defra.gov.uk/air-pollution/daqi?view=more-info

[28] CityPulse Project, “CityPulse.” Accessed: Jul. 20, 2018. [Online]. Available: http://www.ict-citypulse.eu/page/

Wei Wang (M’13) received the B.S. and Ph.D. degrees in electronic and communication engineering from the Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2011, respectively.

She was a Postdoctoral Researcher with the School of Electronic and Information Engineering, Peking University, China. She is currently an Associate Professor with the Cyber-Physical-Social Systems Lab, Huazhong University of Science and Technology. She has authored almost 20 papers in international journals and at conferences. Her research interests include cloud security, network coding, and multimedia transmission.

Peng Xu (M’13) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, Wuhan, China, in 2010.

From 2010 to 2013, he was a Postdoctoral Researcher with the Huazhong University of Science and Technology, and an Associate Research Fellow with the University of Wollongong, Wollongong, NSW, Australia, from 2018 to 2019. He is currently an Associate Professor with the School of Cyber Science and Engineering from the Huazhong University of Science and Technology. He was Principal Investigator in eight grants, including three projects national nature science foundation of China. He is also with the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China. He has authored more than 30 research papers and two books. His research interests are in the field of cryptography.

Dongli Liu received the B.E. degree in information security in 2016 from the Huazhong University of Science and Technology, Wuhan, China, where he is currently working toward the M.S. degree in cyberspace security with the School of Computer Science and Technology.

His current research interests include cryptography and secure policy.

Laurence Tianruo Yang (M'97–SM'15) received the B.E. degree in computer science and technology from Tsinghua University, Beijing, China, and the Ph.D. degree in computer science from the University of Victoria, Victoria, BC, Canada, in 1992 and 2006, respectively.

He is a currently Professor with the School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China, as well as, with Department of Computer Science, St. Francis Xavier University, Antigonish, NS, Canada. His research has been supported by the National Sciences and Engineering Research Council, Canada, and the Canada Foundation for Innovation. His research interests include parallel and distributed computing, embedded and ubiquitous/pervasive computing, and big data.

Zheng Yan (M'06–SM'14) received the Doctor of Science degree in Technology in electrical engineering from the Helsinki University of Technology in 2007.

Before joining the academia in 2011, she has been a Senior Researcher with the Nokia Research Center, Helsinki, Finland, since 2000. She is currently a Professor with Xidian University, Xi'an, China, and a Visiting Professor and a Finnish Academy Research Fellow with Aalto University, Espoo, Finland. Her research interests include trust, security, privacy, and security-related data analytics.

Prof. Yan is the recipient of several awards, including the 2017 Best Journal Paper Award by the IEEE Communication Society Technical Committee on Big Data and the Outstanding Associate Editor of 2017/2018 for IEEE ACCESS. He is an Associate Editor for the IEEE INTERNET OF THINGS JOURNAL, *Information Fusion*, *Information Sciences*, IEEE ACCESS, and the *Journal of Network and Computer Applications*. She was a General Chair or Program Chair for numerous international conferences including IEEE TrustCom 2015. She is a Founding Steering Committee Co-Chair of the IEEE Blockchain Conference.