Lightweight Searchable Public-Key Encryption for Cloud-Assisted Wireless Sensor Networks

Peng Xu, *Member, IEEE*, Shuanghong He, Wei Wang, *Member, IEEE*, Willy Susilo, *Senior Member, IEEE*, and Hai Jin, *Senior Member, IEEE*

***Abstract*—The industrial Internet of Things is flourishing, which is unprecedentedly driven by the rapid development of wireless sensor networks (WSNs) with the assistance of cloud computing. The new wave of technology will give rise to new risks to cyber security, particularly the data confidentiality in cloud-assisted WSNs (CWSNs). Searchable public-key encryption (SPE) is a promising method to address this problem. In theory, it allows sensors to upload public-key ciphertexts to the cloud, and the owner of these sensors can securely delegate a keyword search to the cloud and retrieve the intended data while maintaining data confidentiality. However, all existing and semantically secure SPE schemes have expensive costs in terms of generating ciphertexts and searching keywords. Hence, this paper proposes a lightweight SPE (LSPE) scheme with semantic security for CWSNs. LSPE reduces a large number of the computation-intensive operations that are adopted in previous works; thus, LSPE has search performance close to that of some practical searchable symmetric encryption schemes. In addition, LSPE saves considerable time and energy costs of sensors for generating ciphertexts. Finally, we experimentally test LSPE and compare the results with some previous works to quantitatively demonstrate the above advantages.**

***Index Terms*—Cloud, industrial Internet of Things (IIoT), lightweight searchable public-key encryption (LSPE), semantic security, wireless sensor networks (WSNs).**

Manuscript received October 17, 2017; revised December 1, 2017; accepted December 11, 2017. Date of publication December 18, 2017; date of current version August 1, 2018. This work was supported in part by the National Program on Key Basic Research Project (973 Program) under Grant 2014CB340600, in part by the National Natural Science Foundation of China under Grant 61472156, in part by the Shenzhen Fundamental Research Program under Grant JCYJ20170413114215614, and in part by the Fundamental Research Funds for the Central Universities under Grant 2017KFYXJJ062. Paper no. TII-17-2418. *(Corresponding author: Peng Xu.)*

P. Xu is with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430073 China, and also with the Shenzhen Huazhong University of Science and Technology Research Institute, Shenzhen 518057, China (e-mail: xupeng@mail.hust.edu.cn).

S. He and H. Jin are with the Services Computing Technology and System Lab, Cluster and Grid Computing Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430073, China (e-mail: heshuanghong@mail.hust.edu.cn; hjin@mail.hust.edu.cn).

W. Wang is with the Cyber-Physical-Social Systems Lab, School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430073, China (e-mail: viviawangwei@hust.edu.cn).

W. Susilo is with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia (e-mail: wsusilo@uow.edu.au).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TII.2017.2784395

I. Introduction

DRIVEN by the fourth industrial revolution, the industrial Internet of Things (IIoT) is rapidly emerging. As an extended concept of Internet of Things (IoT) [1], [2], IIoT is the use of IoT technologies in manufacturing. For example, IIoT takes advantage of a large volume of sensor data, machine-to-machine communications, automation technologies, and machine learning technologies to provide great potential impacts on 100% of global energy production and on 44% of energy consumption [3]. According to a new report from Grand View Research, the global IIoT market is expected to reach $933.62 billion by 2025 [4]. Undoubtedly, this new industrial wave will provide contributions to the formation of a new era for technological development and economic growth. As one of the most important components of IIoT, wireless sensor networks (WSNs) and the associated cloud technologies are playing increasingly more pivotal roles in various scenarios, such as health care, agriculture, military defense, environmental monitoring, and smart metering [5]–[7].

WSNs connect sensors to the Internet through gateways, which are responsible for the connection between the WSN and the Internet [8]. In this case, a mass of sensors deployed in the monitoring area compose a WSN, and generate a volume of sensor data that will be forwarded by gateways. Notably, the increasing adoption of WSNs, particularly cloud-assisted WSNs (CWSNs), will certainly bring some new challenges in terms of energy consumption and data confidentiality [9]–[12]. The sensors in CWSNs generally collect sensitive data and upload these data to the cloud. Hence, both eavesdroppers and the untrusted cloud are curious about these data, such as the examples shown in [13] and [14]. Therefore, the issue of data confidentiality must be carefully considered when deploying CWSNs in applications [15].

Numerous research works introduce cryptography to CWSNs to protect data confidentiality, and many cryptographic algorithms are adopted. Wang *et al.* [16] propose a secure data division scheme based on homomorphic encryption in CWSNs for health care. Since the sensors in CWSNs are generally energy intensive and computing power limited, Wang *et al.* [17] propose a group key-policy attribute-based encryption with partial outsourcing decryption in WSNs, and Elhoseny *et al.*

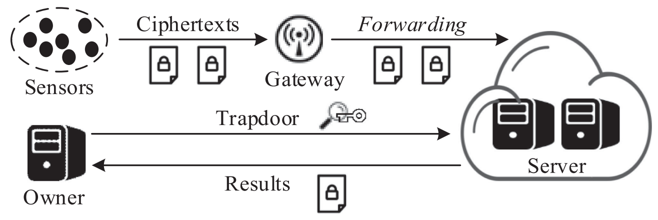

Fig. 1. Typical application of SE in CWSNs.

[18] propose an energy efficient encryption scheme for secure dynamic WSN. Additionally, there are some other encryption methods introduced to CWSNs, such as mixed encryption [19], asymmetric encryption [20], authenticated encryption [21], etc.

Searchable encryption (SE) is one of the promising cryptographic techniques for CWSNs to maintain data confidentiality [22]. When applying SE in CWSNs, as shown in Fig. 1, sensors can generate searchable ciphertexts for their data and upload them to the cloud. To retrieve the intended data, the owner of the sensors delegates a keyword search to the cloud; then, the cloud finds all matching ciphertexts and returns them to the owner; finally, the owner decrypts the intended data. In terms of security, SE guarantees that both eavesdroppers and the untrusted cloud cannot learn any information about sensors' data in some sense.

Currently, SE can be categorized into two types: the first is searchable symmetric-key encryption (SSE) [23], and the second is searchable public-key encryption (SPE) [24]. In the application of CWSNs, SSE requires that all sensors have the same symmetric key to generate ciphertexts. Hence, if one of the sensors is compromised by an adversary, then all the other sensors' data will be leaked. Fortunately, in contrast to SSE, SPE only requires that all sensors store the public key. Hence, SPE is more secure than SSE in practice. However, the existing SPE schemes are still impractical for CWSNs in terms of performance.

In CWSNs, sensors generally have limited energy, and the cloud must complete a search task as soon as possible. Hence, a practical SPE scheme should be highly efficient in generating ciphertexts and searching keywords. However, the existing SPE schemes fail to achieve the above aims. The seminal work of SPE [24] has search complexity that is linear with the total number of ciphertexts. A following work on SPE [25] accelerates the search performance such that its search complexity is sublinear with the total number of ciphertexts. To the best of our knowledge, the sublinear search complexity is the best. However, this paper is still impractical. Our experiment shows that this paper takes an average time of 1.16 ms to find one matching ciphertext. The performance is far from being practical. According to studies on SSE, such as [26], we generally think that the practical performance should be approximately 7.3 μs. In addition, the existing SPE schemes also use many computation-intensive operations to generate ciphertexts. Due to the limited energy of sensors, improving the performance to generate ciphertexts is also needed.

A. Our Ideas

Currently, it is still an interesting and challenging work to improve the search performance of SPE without sacrificing the semantic security of keywords. In theory, this paper can be achieved in two ways: The first is to reduce the search complexity such that the resulting complexity is less than sublinear, and the second is to reduce the number of computation-intensive operations in large while guaranteeing the sublinear search complexity.

According to our previous studies, the first approach mentioned above appears to be impossible when maintaining the semantic security is required. The sublinear search complexity means that the search complexity is linear with the number of matching ciphertexts. If there is a search complexity that is less than sublinear, it means that the corresponding search algorithm can find at least two matching ciphertexts in only one step. In practice, achieving this type of search algorithm requires that the cloud can decide which two ciphertexts have the same keyword even without any authorized keyword search from the owner of sensors. This clearly contradicts the semantic security. In addition, we can also find a fact that implies the impossibility from the studies on SSE. This fact is that no semantically secure SSE scheme can achieve search complexity that is less than sublinear. Hence, this paper focuses on the second approach mentioned above.

Xu *et al.* [25] proposed the first unique SPE scheme (called XW15 in this paper) with both sublinear search complexity and semantic security. We find that this scheme is constructed by many computation-intensive operations. Specifically, XW15 is constructed by a supersingular elliptic curve. Let $\mathbb{G}_1$ and $\mathbb{G}_1'$ denote the corresponding algebraic groups, and let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1' \rightarrow \mathbb{G}_2$ denote the corresponding pairing operations. If $\mathbb{G}_1 = \mathbb{G}_1'$, then we say that $\hat{e}$ is symmetric; otherwise, $\hat{e}$ is asymmetric (more related mathematical definitions will be provided in the following section). XW15 consists of many pairing and multiplication operations of $\mathbb{G}_1$ and many exponentiation operations of $\mathbb{G}_2$. These operations have considerably higher time costs than other cryptographic operations, such as the multiplication and division operations of $\mathbb{G}_2$. We experimentally test these operations by the type-A and type-D supersingular elliptic curves, which are introduced in the pairing-based cryptography (PBC) manual [27]. The two types of elliptic curves have quite different time costs when running cryptographic operations, because these two curves have different embedding degrees that greatly affect the computation complexity. Table I clearly shows the comparisons of the above operations in terms of time cost.

In addition, to generate N ciphertexts for a keyword, XW15 will execute operation $\text{Mul}_{\mathbb{G}_1}$ N times, operation $\text{Exp}_{\mathbb{G}_2}$ $N+1$ times, and operation $\hat{e}$ N times. To search a keyword, suppose that there are N matching ciphertexts also, then XW15 will execute operation $\hat{e}$ $N+1$ times. In other words, XW15 generally takes the number of the above computation-intensive operations, linear with the number of associated ciphertexts, to generate ciphertexts or search keywords.

To summarize, XW15 is impractical even if it has sublinear search complexity. We are interested in constructing a lightweight SPE (LSPE) scheme. It has the same search complexity and semantic security as XW15. In contrast to XW15, it greatly reduces the number of the computation-intensive operations.

TABLE I
TIME COSTS TO EXECUTE DIFFERENT CRYPTOGRAPHIC OPERATIONS WHEN USING DIFFERENT ELLIPTIC CURVES

Operation	Time Cost Per Execution (μs)	
	The Type-A Elliptic Curve	The Type-D Elliptic Curve
SHA-256	2.3	
$\mathrm{Mul}_{\mathbb{G}_2}$	0.9	6
$\mathrm{Div}_{\mathbb{G}_2}$	5.5	18
$\mathrm{Mul}_{\mathbb{G}_1}$	1697	607
$\mathrm{Exp}_{\mathbb{G}_2}$	156	1234
$\hat{e}$	1136	3755

Let $\mathrm{Mul}_{\mathbb{G}_2}$, $\mathrm{Div}_{\mathbb{G}_2}$, and $\mathrm{Exp}_{\mathbb{G}_2}$ denote the multiplication, division, and exponentiation operations of $\mathbb{G}_2$, respectively. Let $\mathrm{Mul}_{\mathbb{G}_1}$ denote the multiplication operation of $\mathbb{G}_1$. Note that each operation is executed 1000 times, and the average time is taken as the time cost per execution.

B. Our Work

According to the concept of searchable public-key ciphertexts with hidden structures (SPCHS) [25], we construct an LSPE scheme. This scheme generates starlike hidden structures among searchable ciphertexts as XW15 does to achieve sublinear search complexity. In contrast to XW15, LSPE has considerably more efficient time costs to generate ciphertexts and search keywords. To generate N ciphertexts for a keyword, LSPE executes one pairing operation $\hat{e}$ and one multiplication operation $\mathrm{Mul}_{\mathbb{G}_1}$. To search a keyword that has N matching ciphertexts, LSPE executes one pairing operation $\hat{e}$. In other words, LSPE has the number of the above computation-intensive operations, independent with the number of the associated ciphertexts, to generate ciphertexts or search keywords. Clearly, LSPE is considerably more practical than XW15. Hence, when applying LSPE in the scenario of CWSNs, sensors can expend less time and energy costs to generate ciphertexts, and the cloud can find all matching ciphertexts in a much shorter time.

In terms of security, LSPE has provable semantic security under the computational bilinear Diffie-Hellman (CBDH) assumption in the random oracle (RO) model. It means that without any delegated keyword search task from the owner of sensors, no one, including eavesdroppers and the cloud, can learn any information about keywords; with a delegated keyword search task, the cloud only knows which ciphertexts are matching.

We experimentally compare our LSPE with XW15. We first code these two schemes. Then, we investigate their time and energy costs to generate some ciphertexts for environmental pollution data using a single-board computer, and we investigate their time costs to search keywords using a workstation. The single-board computer is suggested by Microsoft as a popular platform to run IoT systems [28]. Hence, our experiments are sufficiently convincing in showing the significant advantages of LSPE. In addition, according to the experimental results, we find that the search performance of LSPE is very close to that of a practical SSE scheme [26]. To the best of our knowledge, LSPE is the first semantically secure SPE scheme that has this type of advantage.

C. Organization

The remainder of this paper is organized as follows. Section II reviews the concepts of SPCHS and its semantic security. Section III introduces our LSPE scheme. Section IV introduces the application of LSPE in the scenario of CWSNs. Section V experimentally compares our LSPE and XW15. Section VI introduces the other related works. Section VII concludes this paper.

II. REVIEWING SPCHS AND ITS SEMANTIC SECURITY

SPCHS is a novel and extended concept of SPE. In contrast to the traditional concept of SPE, SPCHS not only defines the search ability of ciphertexts, but also defines the hidden structures constructed by the generated ciphertexts. In other words, SPCHS additionally defines the hidden relationship among the generated ciphertexts, and the corresponding relationship can be disclosed by an authorized keyword search task. The disclosed relationship allows a search process to obtain some new properties, in addition to deciding whether a ciphertext is matching. Xu *et al.* [25] introduces three types of hidden structures: The first one is the starlike structure, which can reduce the search complexity; the second one is the ringlike structure, which allows one to verify the completeness of a search task; and the final one is the treelike structure, which can achieve the content search in some sense.

Our LSPE scheme is an instance of SPCHS. Specifically, LSPE constructs a starlike hidden structure among the generated ciphertexts to achieve the basic requirement, which is sublinear search complexity. In addition, the definition of SPCHS's semantic security is also suitable for LSPE. Hence, this section briefly reviews the concept of SPCHS and its semantic security to help readers to understand LSPE. For more formal details on SPCHS, readers can refer to [25].

SPCHS defines five algorithms, which are algorithms **Setup**, **Structure**, **Encryption**, **Trapdoor**, and **Search**. They are described, respectively, as follows.

1) Algorithm **Setup** is the most fundamental one. According to the requirement on the degree of security, it will generate some system parameters for all other algorithms. The system parameters consists of two parts: one is the master public key; another one is the master private key. In the scenario of CWSNs, this algorithm is implemented by the sensors' owner; the generated master public key is stored in all sensors; the owner secretly stores the master private key.
2) Algorithm **Structure** is used to initialize a hidden structure, and it will be used in algorithm **Encryption**. An initialized hidden structure consists of two parts: one is the public part; another one is the private part. In the scenario of CWSNs, this algorithm is implemented by a sensor before the first time to run algorithm **Encryption**; the generated public part is uploaded by the sensor to the cloud; the generated private part is secretly stored by the sensor.
3) Algorithm **Encryption** is used to generate the searchable ciphertext of an intended keyword, and the generated

ciphertext contains a hidden relationship with some previously generated ciphertexts. In the scenario of CWSNs, this algorithm is implemented by a sensor if it wants to generate keyword-searchable ciphertexts for some collected data; the generated ciphertext is uploaded by the sensor to the cloud; finally, the sensor updates the private part of its hidden structure for the follow-up ciphertexts.

4) Algorithm **Trapdoor** is used to generate the keyword search trapdoor for an intended keyword, and it must take the master private key as input. In the scenario of CWSNs, if the owner would like to retrieve the sensors' data of an intended keyword, he will run this algorithm to generate a keyword search trapdoor, and send this trapdoor to the cloud as an authorized keyword search task. Since only the owner knows the master private key, no one except the owner can delegate a keyword search to the cloud.
5) Algorithm **Search** is used to find all matching ciphertexts of an intended keyword. In the scenario of CWSNs, upon receiving a keyword search trapdoor from the owner, the cloud runs this algorithm to find all matching ciphertexts.

In practice, a secure SPCHS scheme must guarantee the confidentiality of keywords to resist the inside and outside attackers. When applying SPCHS in the scenario of CWSNs, the inside attackers include the compromised sensors and the honest-but-curious cloud. The outside attackers are eavesdroppers. Suppose that all keyword search trapdoors are securely transferred to the cloud. A secure SPCHS scheme means that the compromised sensors and eavesdroppers cannot learn any information about keywords. With a keyword search trapdoor, the cloud only knows which ciphertexts are matching. Without any keyword search trapdoor, the cloud cannot learn any information about keywords.

The above security requirements are defined by the semantic security of SPCHS. Moreover, the semantic security is a more general security definition. It is defined as the semantic security for both keywords and the hidden structures under chosen keyword and structure attacks (SS-CKSA). It models an adaptive attack game on SPCHS, and then it defines that a SPCHS scheme is SS-CKSA secure if no one can win the game with a nonnegligible advantage. The adaptive attack game implies the following.

1) An attacker can know some inside information such as the information known by the compromised sensors and the honest-but-curious cloud.
2) The attacker can choose which sensors are compromised.
3) The attacker can know all outside information such as the information known by eavesdroppers.

Specifically, the adaptive attack game on SPCHS consists of the following five phases.

1) The *setup* phase is implemented by a challenger who will challenge the capability of an attacker to compromise a SPCHS scheme. In this phase, the challenger setups the master public and private keys of SPCHS, initializes some hidden structures by generating their public parts, and publishes the master public key and the public parts to the attacker. This phase simulates the truth that all public parameters can be known by the attackers.
2) The *query 1* phase is launched by the attacker. He will adaptively chose some keywords and hidden structures, and he will query the corresponding keyword search trapdoors and private parts. The challenger will respond to these queries if their responses are not directly related to the attack targets. This phase simulates the truth in the scenario of CWSNs, such that the following statements hold.
   a) Some sensors can be compromised in practice.
   b) The honest-but-curious cloud knows some keyword search trapdoors from the owner.
   c) The transferred ciphertexts can be eavesdropped by the attackers.
3) The *challenge* phase allows the attacker to choose two pairs of keyword and hidden structure as his attack targets. The challenge will generate the challenging keyword-searchable ciphertext for one of the pairs. This phase simulates the truth that attackers can choose the attack targets.
4) The *query 2* phase is the same as the *query 1* phase. This phase simulates the truth that after choosing the attack targets, attackers still can steal some inside and outside information, and these information may be helpful for the successful attack.
5) The *guess* phase is the final phase. In this phase, the attacker will guess which of the two pairs chosen by the attacker in the *challenge* phase is used to generate the challenging keyword-searchable ciphertext. This phase defines the final step of an attack. This definition is a general one to contain many real attacks, like the attacks to learn the content of keywords, steal the private parameters, etc.

If the attacker guesses the correct result, then he wins the above game. Suppose that the probability of the attacker winning the game is $\Pr[\text{Win}]$. The advantage of the attacker winning the game is defined as $\text{Adv}_{\text{SPCHS},\mathcal{A}}^{\text{SS-CKSA}} = \Pr[\text{Win}] - \frac{1}{2}$.

III. Our LSPE Scheme

In this section, according to the concepts of SPCHS and its semantic security, we will construct our LSPE scheme and prove the correctness and semantic security of LSPE. Prior to constructing our scheme, we introduce some related mathematical definitions at first.

Let $\mathbb{G}_1$ denote an additive group with prime order q. Let $\mathbb{G}_2$ denote a multiplicative group that is also of prime order q. Let P be a generator of group $\mathbb{G}_1$. A pairing operation $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ is defined as a function with the following properties [29].

1) Efficient: Given two elements P and $Q \in \mathbb{G}_1$, there is a polynomial time algorithm to compute $\hat{e}(P, Q) \in \mathbb{G}_2$.
2) Bilinear: For any two integers a and $b \in \mathbb{Z}_q^*$, equation $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ holds.
3) Nondegenerate: If P is a generator of $\mathbb{G}_1$, then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_2$.

Let $\mathbf{BG}(1^k)$ be an efficient pairing generator that takes a security parameter 1^k as input and probabilistically outputs $(q, \mathbb{G}_1, \mathbb{G}_2, P, \hat{e})$. Let $\mathcal{W} = \{0, 1\}^*$ be the keyword space.

A. Constructing LSPE

Our LSPE scheme is constructed as follows.

1) $\textbf{Setup}(1^k,\ \mathcal{W})$: Take the security parameter 1^k and the keyword space $\mathcal{W}$ as inputs, run $\mathbf{BG}(1^k)$ to generate parameters $(q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ P,\ \hat{e})$, randomly select $a \in \mathbb{Z}_q^*$, set $Q = aP$, choose two cryptographic hash functions $\mathcal{H}_1 :\ \{0,\ 1\}^* \rightarrow \mathbb{G}_1$ and $\mathcal{H}_2 :\ \mathbb{G}_2 \rightarrow \{0,\ 1\}^{256}$, and output the master secret key $\mathcal{SK} = a$ and the master public key $\mathcal{PK} = (q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ P,\ \hat{e},\ Q,\ \mathcal{H}_1,\ \mathcal{H}_2,\ \mathcal{W})$.
2) $\textbf{Structure}(\mathcal{PK})$: Take $\mathcal{PK}$ as the input, randomly select $u \in \mathbb{Z}_q^*$, and initialize a hidden structure by outputting the public part $\mathcal{PUB} = u \cdot P$ and the private part $\mathcal{PRI} = (u)$. Note that $\mathcal{PRI}$ is a variable list formed as $(u,\ \{(W,\ Pt[u,\ W])|W \in \mathcal{W},\ Pt[u,\ W] \in \mathbb{G}_2\})$.
3) $\textbf{Encryption}(\mathcal{PK},\ W,\ \mathcal{PRI})$: Take $\mathcal{PK}$, a keyword $W \in \mathcal{W}$, and $\mathcal{PRI}$ as inputs, and perform the following steps.
   a) Retrieve $(W,\ Pt[u,\ W])$ by W from $\mathcal{PRI}$.
   b) If it is not found, randomly select $Pt[u,\ W] \in \mathbb{G}_2$, insert $(W,\ Pt[u,\ W])$ into $\mathcal{PRI}$, and output the ciphertext $C = (\mathcal{H}_2(\hat{e}(u \cdot Q,\ \mathcal{H}_1(W))),\ \hat{e}(u \cdot Q,\ \mathcal{H}_1(W)) \cdot Pt[u,\ W])$.
   c) Otherwise, randomly select $R \in \mathbb{G}_2$, set $C = (\mathcal{H}_2(Pt[u,\ W]),\ Pt[u,\ W] \cdot R)$, update $Pt[u,\ W] = R$ in $\mathcal{PRI}$, and output the ciphertext C.
4) $\textbf{Trapdoor}(\mathcal{SK},\ W)$: Take $\mathcal{SK}$ and a keyword $W \in \mathcal{W}$ as inputs and output a keyword search trapdoor $T_W = a \cdot \mathcal{H}_1(W)$ of keyword W.
5) $\textbf{Search}(\mathcal{PK},\ \mathcal{PUB},\ \mathbb{C},\ T_W)$: Take $\mathcal{PK}$, a hidden structure's public part $\mathcal{PUB}$, all keyword-searchable ciphertexts $\mathbb{C}$ (let $\mathbb{C}[i]$ denote the ith ciphertext of $\mathbb{C}$, and $\mathbb{C}[i]$ can be parsed as $\mathbb{C}[i,\ 1] \in \{0,1\}^{256}$ and $\mathbb{C}[i,\ 2]) \in \mathbb{G}_2)$ and a keyword trapdoor T_W of keyword W as inputs, set $\mathbb{C}' = \phi$, and perform the following steps.
   a) Compute $Pt' = \hat{e}(\mathcal{PUB},\ T_W)$.
   b) Seek a ciphertext $\mathbb{C}[i]$ having $\mathbb{C}[i,\ 1] = \mathcal{H}_2(Pt')$.
   c) If the ciphertext exists, add $\mathbb{C}[i]$ into $\mathbb{C}'$, compute $Pt' = Pt'^{-1} \cdot \mathbb{C}[i,\ 2]$, and go to Step 5b).
   d) If no matching ciphertext is found, output $\mathbb{C}'$.

An example of LSPE: Suppose that there are three keyword-searchable ciphertexts generated by LSPE for keyword W_1. These ciphertexts and their hidden structure are shown in Fig. 2. With the keyword search trapdoor $a \cdot \mathcal{H}_1(W_1)$ of keyword W_1, one can compute $\mathcal{H}_2(\hat{e}(\mathcal{PUB},\ a \cdot \mathcal{H}_1(W_1)))$ and find the first matching ciphertext $\mathbb{C}[1]$, since we have $\mathcal{H}_2(\hat{e}(\mathcal{PUB},\ a \cdot \mathcal{H}_1(W_1))) = \mathcal{H}_2(\hat{e}(u \cdot Q,\ \mathcal{H}_1(W_1)))$. Then, one can decrypt ciphertext $\mathbb{C}[1]$ and obtain $Pt[u,\ W_1]_1$. With the decrypted $Pt[u,\ W_1]_1$, one can find the second matching ciphertext $\mathbb{C}[2]$. By using the same method, one can decrypt $Pt[u,\ W_1]_2$ and find the third matching ciphertext $\mathbb{C}[3]$. Consequently, with the keyword search trapdoor of keyword W_1, one can find all the matching ciphertexts in LSPE.

B. Proving the Correctness

The above example intuitively shows the correctness of LSPE. Here, the correctness will be formally proven by the following Theorem 1.

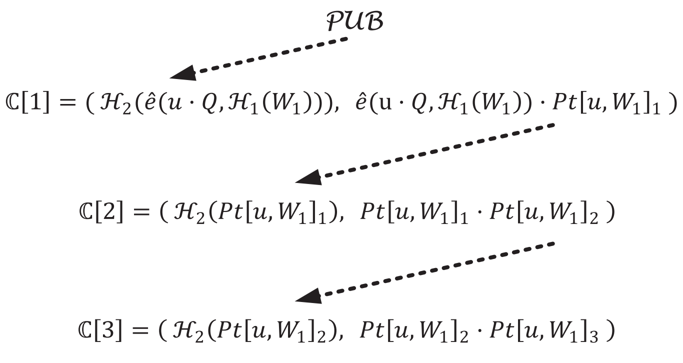
$\mathcal{PUB}$
$\mathbb{C}[1] = (\mathcal{H}_2(\hat{e}(u \cdot Q, \mathcal{H}_1(W_1))),\ \hat{e}(u \cdot Q, \mathcal{H}_1(W_1)) \cdot Pt[u, W_1]_1)$
$\mathbb{C}[2] = (\mathcal{H}_2(Pt[u, W_1]_1),\ Pt[u, W_1]_1 \cdot Pt[u, W_1]_2)$
$\mathbb{C}[3] = (\mathcal{H}_2(Pt[u, W_1]_2),\ Pt[u, W_1]_2 \cdot Pt[u, W_1]_3)$

Fig. 2. Example of LSPE. Note that the dashed arrows denote the hidden structure, parameters $Pt[u,\ W_1]_i$ for $i \in [1,\ 3]$ have different and random values, and we use $Pt[u,\ W_1]_i$ to denote the different values of variable $Pt[u,\ W_1]$ in algorithm $\textbf{Encryption}$.

Theorem 1: Suppose that the hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$ are both collision free, except with a negligible probability. LSPE is correct, also except with a negligible probability.

Proof: Without loss of generality, it is equal to prove that given the keyword search trapdoor $T_{W_i} = a \cdot \mathcal{H}_1(W_i)$ of keyword W_i and the hidden structure's public part $\mathcal{PUB} = u \cdot P$, algorithm $\textbf{Search}(\mathbf{PK},\ \mathcal{PUB},\ \mathbb{C},\ T_{W_i})$ will find all matching ciphertexts of keyword W_i with the hidden structure $\mathcal{PUB}$. The proof is as follows.

According to LSPE, algorithm $\textbf{Search}(\mathcal{PK},\ \mathcal{PUB},\ \mathbb{C},\ T_{W_i})$ first computes $Pt' = \hat{e}(\mathcal{PUB},\ T_{W_i})$. Suppose that algorithm **Encryption** has generated the first keyword-searchable ciphertext of keyword W_i. Let $\mathbb{C}[j]$ denote the ciphertext. We have $\mathbb{C}[j] = (\mathcal{H}_2(\hat{e}(u \cdot Q,\ \mathcal{H}_1(W_i))),\ \hat{e}(u \cdot Q,\ \mathcal{H}_1(W_i)) \cdot Pt[u,\ W_i])$. Since $\mathcal{H}_2(Pt') = \mathcal{H}_2(\hat{e}(u \cdot Q,\ \mathcal{H}_1(W_i)))$ holds, algorithm **Search** can find the first matching ciphertext of keyword W_i, except with a negligible probability that is caused by the collision probabilities of both hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$. In other words, algorithm **Search** could find a ciphertext that has its prefix equals to $\mathcal{H}_2(Pt')$, but the ciphertext does not belong to keyword W_i or the hidden structure $\mathcal{PUB}$. However, the probability of the exception is negligible due to the collision-free properties of both hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$.

By decrypting the first matching ciphertext $\mathbb{C}[j]$, algorithm **Search** obtains $Pt[u,\ W_i]$. Suppose that algorithm **Encryption** has generated the second keyword-searchable ciphertext of keyword W_i. Let $\mathbb{C}[j']$ denote the ciphertext. We have $\mathbb{C}[j'] = (\mathcal{H}_2(Pt[u,\ W_i]),\ Pt[u,\ W_i] \cdot R)$. Hence, algorithm **Search** can find the second matching ciphertext of keyword W_i, also except with a negligible probability that is caused by the collision probability of hash function $\mathcal{H}_2$. By using the same method, all matching ciphertexts of keyword W_i can be found, except with a negligible probability.

C. Proving the Semantic Security

The SS-CKSA security of LSPE relies on the CBDH assumption [29]. It means that if the CBDH assumption holds or the corresponding CBDH problem cannot be efficiently solved in practice, then LSPE is SS-CKSA secure. To prove the SS-CKSA security, we will prove that if there is an attacker who can break the SS-CKSA security, then we can leverage the adversary to solve the CBDH problem. Before presenting the proof, we first review the CDBH assumption.

Definition 1 (The CBDH Assumption): Given parameters $(g,\ \mathbb{G}_1,\ \mathbb{G}_2, P,\ \hat{e})$ generated by $\mathbf{BG}(1^k)$ and $(a\cdot P,\ b\cdot P, c\cdot P)$, where $(a,\ b\ , c)$ are randomly chosen in $\mathbb{Z}_q^*$, the CBDH problem in $\mathbf{BG}(1^k)$ is to compute $\hat{e}(P,\ P)^{abc}$. Let $\mathrm{Adv}_{\mathcal{B}}^{\mathrm{CBDH}}(1^k)$ be the advantage of algorithm $\mathcal{B}$ to solve the CBDH problem. We say that the CBDH assumption holds if the advantage $\mathrm{Adv}_{\mathcal{B}}^{\mathrm{CBDH}}(1^k)$ is negligible in the security parameter 1^k.

The SS-CKSA security of LSPE is proven by the following theorem. Since no probabilistic polynomial time (PPT) algorithm can solve the CBDH problem with a nonnegligible probability, Theorem 2 implies that no PPT attacker can break the SS-CKSA security of LSPE in practice.

Theorem 2: Let the hash functions $\mathcal{H}_1$ and $\mathcal{H}_2$ be modeled as the ROs $\mathcal{Q}_{\mathcal{H}_1}(\cdot)$ and $\mathcal{Q}_{\mathcal{H}_2}(\cdot)$, respectively. Let $\mathcal{Q}_{\mathcal{P}}$, $\mathcal{Q}_{\mathcal{T}}(\cdot)$, and $\mathcal{Q}_{\mathcal{E}}(\cdot)$ be three oracles to response the issues of querying the private part of a hidden structure, querying the keyword search trapdoor of a keyword, and querying the ciphertext of a keyword with a hidden structure, respectively. Suppose that there are a total of N hidden structures in practice and that a PPT attacker $\mathcal{A}$ has an advantage of $\mathrm{Adv}_{\mathrm{LSPE},\mathcal{A}}^{\text{SS-CKSA}}$ to break LSPE in the SS-CKSA game, in which $\mathcal{A}$ makes at most q_1 queries to oracle $\mathcal{Q}_{\mathcal{H}_1}(\cdot)$, at most q_2 queries to oracle $\mathcal{Q}_{\mathcal{H}_2}(\cdot)$, at most q_p queries to oracle $\mathcal{Q}_{\mathcal{P}}$, at most q_t queries to oracle $\mathcal{Q}_{\mathcal{T}}(\cdot)$, and at most q_e queries to oracle $\mathcal{Q}_{\mathcal{E}}(\cdot)$. Then, there is a PPT algorithm $\mathcal{B}$ that solves the CBDH problem in $\mathbf{BG}(1^k)$ with probability

$$\mathrm{Adv}_{\mathcal{B}}^{\mathrm{CBDH}}(1^k) \geq \frac{256}{2e^4(q_t+q_p)^4(q_2+q_e+1)}\mathrm{Adv}_{\mathrm{LSPE},\mathcal{A}}^{\text{SS-CKSA}}$$

where e is the base of the natural logarithm.

Proof: In this proof, algorithm $\mathbf{B}$ will be constructed to leverage the capability of attacker $\mathcal{A}$ to solve the CDBH problem in $\mathbf{BG}(1^k)$. Hence, algorithm $\mathcal{B}$ will simulate and play the SS-CKSA game with attacker $\mathcal{A}$ according to the CDBH problem. This game consists of five phases:

1) In the *setup* phase, algorithm $\mathcal{B}$ will simulate the master public key and all public parts of the N hidden structures, and it will initialize some data structure to store the following issues of attacker $\mathcal{A}$ and the corresponding responses of algorithm $\mathcal{B}$.
2) In the *query 1* and *2* phases, algorithm $\mathcal{B}$ will simulate the responses of the issues from attacker $\mathcal{A}$, including the issues to oracles $\mathcal{Q}_{\mathcal{H}_1}(\cdot)$, $\mathcal{Q}_{\mathcal{H}_2}(\cdot)$, $\mathcal{Q}_{\mathcal{P}}$, $\mathcal{Q}_{\mathcal{T}}(\cdot)$, and $\mathcal{Q}_{\mathcal{E}}(\cdot)$.
3) In the *challenge* phase, attacker $\mathcal{A}$ will choose two attack targets, and algorithm $\mathcal{B}$ will simulate a challenge ciphertext for one of these two targets.
4) In the *guess* phase, algorithm $\mathcal{B}$ will attempt to solve the CDBH problem according to attacker $\mathcal{A}$'s issues in the *query 1 and 2* phases.

Let Coin $\overset{\sigma}{\leftarrow}\{0,\ 1\}$ denote the operation to select Coin $\in\{0,\ 1\}$ with probability $\Pr[\mathrm{Coin}=1]=\sigma$. The specified value of σ will be decided later. The SS-CKSA game between algorithm $\mathcal{B}$ and attacker $\mathcal{A}$ is as follows.

1) *Setup* phase: Algorithm $\mathcal{B}$ takes the keyword space $\mathcal{W}$ and parameters $(q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ P,\ \hat{e},\ aP,\ bP,\ cP)$ as inputs, and it performs the following steps.
   a) Initialize four empty lists $\mathbf{Pt}\subseteq\mathcal{W}\times\mathbb{G}_1\times\mathbb{G}_2$, $\mathbf{S}\subseteq\mathbb{G}_1\times\mathbb{Z}_q^*\times\{0,\ 1\}$, $\mathbf{H1}\subseteq\mathcal{W}\times\mathbb{G}_1\times\mathbb{Z}_q^*\times\{0,\ 1\}$ and $\mathbf{H2}\subseteq\mathbb{G}_2\times\{0,\ 1\}^{256}$.
   b) Set the master public key $\mathcal{PK}=(q,\ \mathbb{G}_1,\ \mathbb{G}_2,\ P,\ \hat{e},\ Q=aP,\ \mathcal{W})$.
   c) Initialize N hidden structures through the following steps for $i\in[1,\ N]$.
      i) Select a random $u_i\in\mathbb{Z}_q^*$ and $\mathrm{Coin_i}\overset{\sigma}{\leftarrow}\{0,\ 1\}$.
      ii) If $\mathrm{Coin_i}=1$, compute $\mathcal{PUB}_i=u_i\cdot bP$.
      iii) Otherwise, compute $\mathcal{PUB}_i=u_i\cdot P$.
   d) Set $\mathbf{P}=\{\mathcal{PUB}_i|i\in[1,\ N]\}$ and $\mathbf{S}=\{\mathcal{PUB}_i,\ u_i,\ \mathrm{Coin_i}|\mathrm{i}\in[1,\ \mathrm{N}]\}$.
   e) Send $\mathcal{PK}$ and $\mathbf{P}$ to attacker $\mathcal{A}$.
2) *Query 1* phase: Attacker $\mathcal{A}$ adaptively issues the following queries multiple times under the condition that the same issue is queried only one time.
   a) Hash query $\mathcal{Q}_{\mathcal{H}_1}(W)$: In each query, attacker $\mathcal{A}$ issues a keyword $W\in\mathcal{W}$. With the issued keyword, algorithm $\mathcal{B}$ performs the following steps.
      i) Select a random $x\in\mathbb{Z}_q^*$ and Coin $\overset{\sigma}{\leftarrow}\{0,\ 1\}$.
      ii) If Coin $=0$, add $(W,\ x\cdot P,\ x,\ \mathrm{Coin})$ into $\mathbf{H1}$ and send xP to $\mathcal{A}$.
      iii) Otherwise, add $(W,\ x\cdot cP,\ x,\ \mathrm{Coin})$ into $\mathbf{H1}$ and send xcP to $\mathcal{A}$.
   b) Hash query $\mathcal{Q}_{\mathcal{H}_2}(Y)$: In each query, attacker $\mathcal{A}$ issues an element $Y\in\mathbb{G}_2$. With the issued element, algorithm $\mathcal{B}$ selects a random value $V\in\{0,\ 1\}^{256}$ as its response, and it adds $(Y,\ V)$ into $\mathbf{H2}$.
   c) Trapdoor query $\mathcal{Q}_{\mathcal{T}}(W)$: In each query, attacker $\mathcal{A}$ issues a keyword $W\in\mathcal{W}$. With the issued keyword, algorithm $\mathcal{B}$ performs the following steps.
      i) If record $(W,\ *,\ *,\ *)\notin\mathbf{H1}$, query $\mathcal{Q}_{\mathcal{H}_1}(W)$.
      ii) Retrieve $(W,\ X,\ x,\ \mathrm{Coin})$ by W from $\mathbf{H1}$.
      iii) If Coin $=0$, send $x\cdot Q$ to $\mathcal{A}$.
      iv) Otherwise, abort and output $\perp$.

      Note that if Coin $=0$, algorithm $\mathcal{B}$ will send the correct trapdoor of the issued keyword to attacker $\mathcal{A}$.
   d) Privacy query $\mathcal{Q}_{\mathcal{P}}(\mathcal{PUB})$: In each query, attacker $\mathcal{A}$ issues a public part $\mathcal{PUB}\in\mathbf{P}$. With the issued public part, algorithm $\mathcal{B}$ performs the following steps.
      i) Retrieve $(\mathcal{PUB},\ u,\ \mathrm{Coin})$ by $\mathbf{PUB}$ from $\mathbf{S}$.
      ii) If Coin $=0$, send u to $\mathcal{A}$.
      iii) Otherwise, abort and output $\perp$.
   e) Encryption query $\mathcal{Q}_{\mathcal{E}}(W,\ \mathcal{PUB})$: In each query, attacker $\mathcal{A}$ issues a keyword $W\in\mathcal{W}$ and a public part $\mathcal{PUB}\in\mathbf{P}$. With the issued parameters of a hidden structure, algorithm $\mathcal{B}$ performs the following steps.
      i) If $(W,\ *,\ *,\ *)\notin\mathbf{H1}$, query $\mathcal{Q}_{\mathcal{H}_1}(W)$.
      ii) Retrieve $(W,\ X,\ x,\ \mathrm{Coin})$ and $(\mathcal{PUB},\ u, \mathrm{Coin}')$ by W and $\mathcal{PUB}$ from $\mathbf{H1}$ and $\mathbf{S}$ respectively.
      iii) Seek $(W,\ \mathcal{PUB},\ Pt[u,\ W])$ by W and $\mathcal{PUB}$ in $\mathbf{Pt}$.

iv) If it does not exist, select a random $Pt[u, W] \in \mathbb{G}_2$, insert $(W, \mathcal{PUB}, Pt[u, W])$ into $\mathbf{Pt}$, and perform the following steps.
   A) If $\text{Coin} = 1 \wedge \text{Coin}' = 1$, randomly select $C[1] \in \{0, 1\}^{256}$ and $C[2] \in \mathbb{G}_2$.
   B) If $\text{Coin} = 1 \wedge \text{Coin}' = 0$, compute $C[0] = \hat{e}(u \cdot aP, x \cdot cP)$, $C[1] = \mathcal{Q}_{\mathcal{H}_2}(C[0])$ and $C[2] = C[0] \cdot Pt[u, W]$.
   C) If $\text{Coin} = 0 \wedge \text{Coin}' = 1$, compute $C[0] = \hat{e}(u \cdot aP, x \cdot bP)$, $C[1] = \mathcal{Q}_{\mathcal{H}_2}(C[0])$ and $C[2] = C[0] \cdot Pt[u, W]$.
   D) If $\text{Coin} = 0 \wedge \text{Coin}' = 0$, compute $C[0] = \hat{e}(u \cdot aP, x \cdot P)$, $C[1] = \mathcal{Q}_{\mathcal{H}_2}(C[0])$ and $C[2] = C[0] \cdot Pt[u, W]$.
   E) Send the ciphertext $C = (C[1], C[2])$ to $\mathcal{A}$.

v) Otherwise, randomly select $R \in \mathbb{G}_2$, send the ciphertext $C = (\mathcal{Q}_{\mathcal{H}_2}(Pt[u, W]), Pt[u, W] \cdot R)$ to $\mathcal{A}$, and update $Pt[u, W] = R$ in $\mathbf{Pt}$;

Note that algorithm $\mathcal{B}$ can generate the correct ciphertexts for attacker $\mathcal{A}$'s issues, except for the special case in step (iv)-A) under step 2e). However, if attacker $\mathcal{A}$ can find this exception, it means that attacker $\mathcal{A}$ issues hash query $\mathcal{Q}_{\mathcal{H}_2}$ with some elements of the special form $\hat{e}(P, P)^{abc \cdot z}$, where $z \in \mathbb{Z}_q^*$ is a variable. Clearly, such issues are helpful for algorithm $\mathcal{B}$ to solve the CBDH problem. Hence, in the following content, we suppose that attacker $\mathcal{A}$ cannot find that exception.

3) *Challenge* phase: Attacker $\mathcal{A}$ sends two challenge keyword–structure pairs $(W_0^*, \mathcal{PUB}_0^*)$ and $(W_1^*, \mathcal{PUB}_1^*)$ to algorithm $\mathcal{B}$. Then, $\mathcal{B}$ performs the following steps.
   a) Retrieve $(\mathcal{PUB}_0^*, u_0^*, PCoin_0^*)$ and $(\mathcal{PUB}_1^*, u_1^*, PCoin_1^*)$ by $\mathcal{PUB}_0^*$ and $\mathcal{PUB}_1^*$, respectively, from $\mathbf{S}$.
   b) If $PCoin_0^* = 0 \vee PCoin_1^* = 0$, then abort and output $\perp$.
   c) If $(W_r^*, *, *, *) \notin \mathbf{H1}$ for $r \in \{0, 1\}$, query $\mathcal{Q}_{\mathcal{H}_1}(W_r^*)$.
   d) Retrieve $(W_0^*, X_0^*, x_0^*, WCoin_0^*)$ and $(W_1^*, X_1^*, x_1^*, WCoin_1^*)$ by W_0^* and W_1^*, respectively, from $\mathbf{H1}$.
   e) If $WCoin_0^* = 0 \vee WCoin_1^* = 0$, then abort and output $\perp$.
   f) Randomly select $d \in \{0, 1\}$.
   g) Seek $(W_d^*, \mathcal{PUB}_d^*, Pt[u_d^*, W_d^*])$ by W_d^* and $\mathcal{PUB}_d^*$ in $\mathbf{Pt}$.
   h) If it does not exists, randomly select $Y \in \{0, 1\}^{256}$, $Pt[u_d^*, W_d^*] \in \mathbb{G}_2$ and $R \in \mathbb{G}_2$, insert $(W_d^*, \mathcal{PUB}_d^*, Pt[u_d^*, W_d^*])$ into $\mathbf{Pt}$, and send the challenge ciphertext $C_d = (Y, R \cdot Pt[u_d^*, W_d^*])$ to $\mathcal{A}$.
   i) Otherwise, randomly select $R \in \mathbb{G}_2$, set $C[1] = \mathcal{Q}_{\mathcal{H}_1}(Pt[u_d^*, W_d^*])$ and $C[2] = Pt[u_r^*, W_r^*] \cdot R$, update $Pt[u_d^*, W_d^*] = R$ in $\mathbf{Pt}$, and send the challenge ciphertext $C_d = (C[1], C[2])$ to $\mathcal{A}$.

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

5) *Guess* phase: Attacker $\mathcal{A}$ sends a guess d' to algorithm $\mathcal{B}$. Irrespective of whether the guess is correct, algorithm $\mathcal{B}$ randomly selects a record (Y, V) from $\mathbf{H2}$, and outputs $Y^{1/(u_d^* \cdot x_d^*)}$ as its solution for the CBDH problem.

In the following content, we will compute the advantage of algorithm $\mathcal{B}$ to solve the CBDH problem in the above SS-CKSA game. Let $\overline{\text{Abort}}$ be the event that algorithm $\mathcal{B}$ does not abort in the above game. Let Query be the event that attacker $\mathcal{A}$ issues hash query $\mathcal{Q}_{\mathcal{H}_2}$ with element $\hat{e}(P, P)^{abc \cdot u_0^* x_0^*}$ or $\hat{e}(P, P)^{abc \cdot u_1^* x_1^*}$. Let Query_d be the event that attacker $\mathcal{A}$ issues hash query $\mathcal{Q}_{\mathcal{H}_2}$ with element $\hat{e}(P, P)^{abc \cdot u_d^* x_d^*}$.

According to the above game, we have that (1) if algorithm $\mathcal{B}$ does not abort, then the above game is indistinguishable from a real SS-CKSA game in the view of attacker $\mathcal{A}$, and (2) if $Y = \hat{e}(P, P)^{abc \cdot u_d^* x_d^*}$ holds in the above *guess* phase, then algorithm $\mathcal{B}$ successfully solves the CBDH problem. Hence, we will first compute the probabilities of events $\overline{\text{Abort}}$ and Query_d. According to $\Pr[\text{Query}_d]$, it is easy to compute $\text{Adv}_{\mathcal{B}}^{\text{CBDH}}(1^k)$.

Claim 1: We have $\Pr[\overline{\text{Abort}}] \geq \frac{256}{e^4(q_t+q_p)^4}$, where e is the base of the natural logarithm.

Proof: According to the above SS-CKSA game, algorithm $\mathcal{B}$ may abort in the trapdoor query $\mathcal{Q}_{\mathcal{T}}$, privacy query $\mathcal{Q}_{\mathcal{P}}$, and challenge phases. Moreover, all cases that make algorithm $\mathcal{B}$ abort are independent. Hence, we have that $\Pr[\overline{\text{Abort}}] = (1-\sigma)^{q_t+q_p}\sigma^4$. Let $\sigma = \frac{4}{q_t+q_p+4}$. we have that

$$\Pr[\overline{\text{Abort}}] \geq \frac{256}{e^4(q_t+q_p)^4}$$

where e is the base of the natural logarithm.

Claim 2: Suppose that algorithm $\mathcal{B}$ does not abort in the above SS-CKSA game. We have that $\Pr[\text{Query}_d] \geq \frac{1}{2}\text{Adv}_{\text{LSPE},\mathcal{A}}^{\text{SS-CKSA}}$.

Proof: According to the definition of SS-CKSA security, we have that $\text{Adv}_{\text{LSPE},\mathcal{A}}^{\text{SS-CKSA}} = \Pr[d = d'] - \frac{1}{2}$. In addition, if event Query never occurs, then attacker $\mathcal{A}$ has no advantage to win the above game since the challenge ciphertext is independent from all challenge keyword-and-structures pairs. Hence, we have that

$$\begin{aligned}\Pr[d = d'] - \frac{1}{2} &= \Pr[d = d'|\text{Query}]\Pr[\text{Query}] \\ &\quad + \Pr[d = d'|\overline{\text{Query}}]\Pr[\overline{\text{Query}}] - \frac{1}{2} \\ &= \left(\Pr[d = d'|\text{Query}] - \frac{1}{2}\right)\end{aligned}$$

Furthermore, we have that $\Pr[\text{Query}] \geq \text{Adv}_{\text{LSPE},\mathcal{A}}^{\text{SS-CKSA}}$. Since attacker $\mathcal{A}$ has the same probability to issue hash query $\mathcal{Q}_{\mathcal{H}_2}$ with elements $\hat{e}(P, P)^{abc \cdot u_0^* x_0^*}$ and $\hat{e}(P, P)^{abc \cdot u_1^* x_1^*}$. We finally have that

$$\Pr[\text{Query}_d] \geq \frac{1}{2}\text{Adv}_{\text{LSPE},\mathcal{A}}^{\text{SS-CKSA}}.$$

According to the above game, there are at most $q_2 + q_e + 1$ records in $\mathbf{H2}$. Hence, under the condition that algorithm $\mathcal{B}$ does not abort in the above SS-CKSA game, *Claim 2* implies that algorithm $\mathcal{B}$ has a probability of greater than

$$\frac{1}{2(q_2+q_e+1)}\mathrm{Adv}_{\mathrm{LSPE},\mathcal{A}}^{\mathrm{SS\text{-}CKSA}}$$

to randomly select a record $(Y,\ V)$ from **H2** in the *guess* phase having $Y=\hat{e}(P,\ P)^{abc\cdot u_d^* x_d^*}$.

Finally, according to *Claim 1*, we have that

$$\mathrm{Adv}_{\mathcal{B}}^{\mathrm{CBDH}}(1^k)\geq\frac{256}{2e^4(q_t+q_p)^4(q_2+q_e+1)}\mathrm{Adv}_{\mathrm{LSPE},\mathcal{A}}^{\mathrm{SS\text{-}CKSA}}.$$

IV. Applying LSPE in CWSNs

When applying LSPE in CWSNs, LSPE must cooperate with a traditional public-key encryption (PKE) scheme and a symmetric-key encryption (SKE) scheme, such as rivest-Shamir-Adleman encryption (RSA) and advanced encryption standard (AES) schemes. Without loss of generality, a traditional PKE scheme consists of algorithms $\mathbf{Setup}_{\mathrm{PKE}}$, $\mathbf{Enc}_{\mathrm{PKE}}$, and $\mathbf{Dec}_{\mathrm{PKE}}$. Algorithm $\mathbf{Setup}_{\mathrm{PKE}}(1^k)$ takes a security parameter 1^k as input, and probabilistically outputs a pair of public and private keys $(\mathcal{PK}',\ \mathcal{SK}')$; algorithm $\mathbf{Enc}_{\mathrm{PKE}}(\mathcal{PK}',\ M)$ takes $\mathcal{PK}'$ and a plaintext M as inputs, and probabilistically outputs a ciphertext C. Algorithm $\mathbf{Dec}_{\mathrm{PKE}}(\mathcal{SK}',\ C)$ takes $\mathcal{SK}'$ and a ciphertext C as inputs and decrypts the contained plaintext M. An SKE scheme generally consists of algorithms $\mathbf{Enc}_{\mathrm{SKE}}$ and $\mathbf{Dec}_{\mathrm{SKE}}$. Algorithm $\mathbf{Enc}_{\mathrm{SKE}}(K,\ M)$ takes a symmetric key K and a plaintext M as inputs, and outputs a ciphertext C. Algorithm $\mathbf{Dec}_{\mathrm{SKE}}(K,\ C)$ takes a symmetric key K and a ciphertext C as inputs, and decrypts the contained plaintext M.

The LSPE-based CWSNs system generally consists of the following phases.

1) *Setup* phase: In this phase, the owner of sensors chooses a security parameter 1^k, runs algorithm $\mathbf{Setup}(1^k)$ of LSPE to generate $(\mathcal{PK},\ \mathcal{SK})$, runs algorithm $\mathbf{Setup}_{\mathrm{PKE}}(1^k)$ of a PKE scheme to generate $(\mathcal{PK}',\ \mathcal{SK}')$, stores $(\mathcal{PK},\ \mathcal{PK}')$ in all sensors, and deploys these sensor in the real world to collect data.
2) *Data collection* phase: Suppose that a sensor would like to upload its collected data F to the cloud. First, if it is the first time to upload data, it runs algorithm $\mathbf{Structure}(\mathcal{PK})$ to initialize a hidden structure $(\mathcal{PUB},\ \mathcal{PRI})$ and upload $\mathcal{PUB}$ to the cloud. Second, it extracts some keywords from the data F. Let $\{W_1,\ldots,W_n\}$ be the extracted keywords. Third, it runs algorithm $\mathbf{Encryption}(\mathcal{PK},\ W_i,\ \mathcal{PRI})$ for $i\in[1,\ n]$ to generate keyword-searchable ciphertexts $\{C_1,\ldots,C_n\}$, randomly chooses a symmetric key K, runs algorithm $\mathbf{Enc}_{\mathrm{PKE}}(\mathcal{PK}',\ K)$ to generate a ciphertext C_{PKE}, and runs algorithm $\mathbf{Enc}_{\mathrm{SKE}}(K,\ F)$ to generate a ciphertext C_{SKE}. Finally, it uploads all ciphertexts $\{C_1,\ldots,C_n,\ C_{\mathrm{PKE}},\ C_{\mathrm{SKE}}\}$ to the cloud.
3) *Data retrival* phase: Suppose that the owner would like to retrieve the data of keyword W_i from the cloud. It runs algorithm $\mathbf{Trapdoor}(\mathcal{SK},\ W_i)$ to generate the keyword search trapdoor T_{W_i} of keyword W_i and securely uploads T_{W_i} to the cloud. First, the cloud runs algorithm $\mathbf{Search}(\mathcal{PK},\ \mathcal{PUB},\ \mathbb{C},\ T_{W_i})$ for all hidden structures' public parts to find all matching ciphertexts. Second, the cloud sends all matching ciphertexts' PKE and SKE parts to the owner. Finally, the owner obtains the intended data by decrypting the receiver PKE and SKE parts. For example, suppose that $\{C_1,\ldots,C_n,\ C_{\mathrm{PKE}},\ C_{\mathrm{SKE}}\}$ is a matching ciphertext (it means that there is a part $C_j\in\{C_1,\ldots,C_n\}$ containing keyword W_i). Then, the cloud sends $\{C_{\mathrm{PKE}},\ C_{\mathrm{SKE}}\}$ to the owner. Finally, the owner decrypts the PKE part C_{PKE} using the private key $\mathcal{SK}'$ to obtain a symmetric key K, and then it decrypts the SKE part C_{SKE} with the symmetric key K to obtain the intended data F.

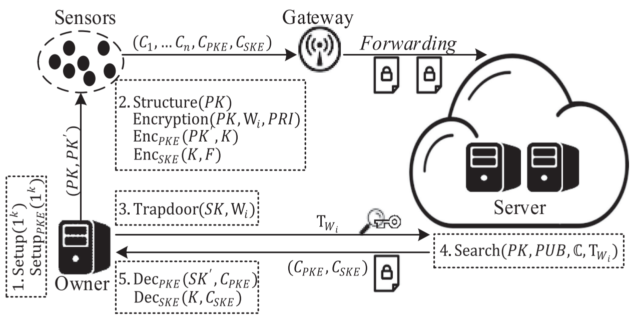
Sensors
Gateway
$(C_1,\ldots C_n,C_{PKE},C_{SKE})$
Forwarding
2. Structure(PK)
Encryption(PK,W_i,PRI)
$\mathrm{Enc}_{PKE}(PK',K)$
$\mathrm{Enc}_{SKE}(K,F)$
(PK,PK')
1. Setup(1^k)
$\mathrm{Setup}_{PKE}(1^k)$
Server
3. Trapdoor(SK,W_i)
T_{W_i}
4. Search$(PK,PUB,\mathbb{C},T_{W_i})$
Owner
5. $\mathrm{Dec}_{PKE}(SK',C_{PKE})$
$\mathrm{Dec}_{SKE}(K,C_{SKE})$
(C_{PKE},C_{SKE})

Fig. 3. Applying LSPE in CWSNs.

Fig. 3 shows the interacting processes of the above system. In this system, all data are encrypted by a PKE scheme and an SKE scheme. Hence, no attacker, including eavesdroppers and the honest-but-curious cloud, can learn any information about the data. In addition, all extracted keywords are encrypted by LSPE. According to the provable SS-CKSA security, LSPE guarantees the practical confidentiality of keywords.

V. Experiments and Comparisons

According to the construction of LSPE, it is easy to find that LSPE has the same search complexity as XW15. Hence, this section will experimentally show that LSPE is considerably more efficient than XW15 in practice since it has reduced a large number of computation-intensive operations.

A. Performance Evaluation

Suppose that N keyword-searchable ciphertexts with a hidden structure for a keyword are generated by XW15 and LSPE. Table II shows the number of computation-intensive operations that are needed by these two schemes. In XW15, the numbers of most of the computation-intensive operations are linear with N. Specially, the numbers of the three most expensive operations, namely the pairing operation, the multiplication operation in group $\mathbb{G}_1$, and the exponentiation operation in group $\mathbb{G}_2$, are all linear with N. In contrast to XW15, LSPE only needs one pairing operation and one multiplication operation in group $\mathbb{G}_1$. Although LSPE additionally needs N multiplication operations in group $\mathbb{G}_2$ and N hashing operations, these two types of operations have considerably less time costs than the other operations.

Consider finding N matching ciphertexts of a keyword. Table II shows that in XW15, the number of the pairing

TABLE II
PERFORMANCE EVALUATION

Schemes	The number of computation-intensive operations		Size per ciphertext
	Encryption algorithm	Search algorithm	
XW15	$N \cdot (\hat{e} + \mathcal{H}_1 + \mathrm{Mul}_{\mathbb{G}_1} + \mathrm{Exp}_{\mathbb{G}_2} + \mathrm{Mul}_{\mathbb{G}_2}) + 1 \cdot \mathrm{Exp}_{\mathbb{G}_2}$	$(N+1) \cdot \hat{e} + N \cdot \mathrm{Div}_{\mathbb{G}_2}$	$\lvert\mathbb{G}_1\rvert + 2 \cdot \lvert\mathbb{G}_2\rvert$
LSPE	$1 \cdot (\hat{e} + \mathcal{H}_1 + \mathrm{Mul}_{\mathbb{G}_1}) + N \cdot (\mathcal{H}_2 + \mathrm{Mul}_{\mathbb{G}_2})$	$1 \cdot \hat{e} + (N+1) \cdot \mathcal{H}_2 + N \cdot \mathrm{Div}_{\mathbb{G}_2}$	$256 + \lvert\mathbb{G}_2\rvert$

Note that $\hat{e}$ denotes one pairing operation; $\mathcal{H}_1$ denotes one hashing operation of hash function $\mathcal{H}_1$; $\mathcal{H}_2$ denotes one hashing operation of hash function $\mathcal{H}_2$; $\mathrm{Mul}_{\mathbb{G}_1}$ denotes one multiplication in group $\mathbb{G}_1$; $\mathrm{Exp}_{\mathbb{G}_2}$, $\mathrm{Mul}_{\mathbb{G}_2}$, and $\mathrm{Div}_{\mathbb{G}_2}$ denote one exponentiation, multiplication, and division in group $\mathbb{G}_2$, respectively; $\lvert\mathbb{G}_1\rvert$ and $\lvert\mathbb{G}_2\rvert$ denote the binary size of groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively.

TABLE III
SYSTEM CONFIGURATION AND ELLIPTIC CURVE

Server	Intel Xeon CPU E5-2420 v2 @ 2.20 GHz
IoT device	RASPBERRY PI 3 MODEL B
OS and compiler	Linux and gcc4.4.7
Program library	PBC
	Mathematical parameters
Elliptic curve	$y^2 = x^3 + x$
Base field	8780710799663312522437781984754049815806883199414208211028653399266475630880222957078625179422662221423155858769582317459277713367317481324925129998224791
Group order	$2^{159} + 2^{107} + 1$
	The default unit is decimal

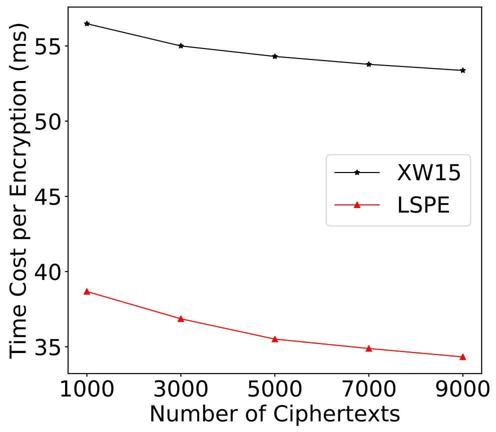

Fig. 4. Time cost of encryption.

operations is linear with N. In contrast to XW15, only one pairing operation is needed by LSPE. Although LSPE additionally needs N hashing operations, the time cost of these operations is considerably less than that of the pairing operations.

The size of a keyword-searchable ciphertext serves as important indicator to measure the communication costs of both XW15 and LSPE. In practice, we generally have $\lvert\mathbb{G}_1\rvert \in [160,\ 512]$ and $\lvert\mathbb{G}_2\rvert \approx 1024$, where $\lvert\mathbb{G}_1\rvert$ and $\lvert\mathbb{G}_2\rvert$ denote the binary sizes of groups $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Table II clearly shows that LSPE is more efficient than XW15.

From the above performance evaluation, we can conclude that LSPE is considerably more efficient than XW15 in terms of generating ciphertexts, searching keywords and transferring ciphertexts. In the following experiment, we will show that LSPE is practical.

B. Experimental Results

In this section, we will investigate the time costs of both XW15 and LSPE to generate keyword-searchable ciphertexts and search keywords, and we will investigate the energy costs of both XW15 and LSPE to generate ciphertexts. In addition to showing the advantages of LSPE, our experiment shows that if there are a large number of matching ciphertexts, the time cost of LSPE to find these ciphertexts is quite close to that of a practical SSE scheme [26].

Test environment: Table III shows the system configuration and the chosen elliptic curve of our experiments. Specifically, we code XW15 and LSPE using the PBC library (a popular cryptographic library [27]) and the chosen elliptic curve, and we use a Raspberry Pi 3 MODEL B (a single-board computer with wireless LAN and bluetooth connectivity [30]) to simulate a sensor device. The experiment to test the time and energy costs of encryption is performed using the Raspberry Pi. A high-precision USB voltage-and-current detector named *USB TESTER* is employed as the testing equipment for energy cost. The experiment to test the time cost of search is performed by a server with an Intel Xeon CPU E5-2420 v2 2.20-GHz processor and 16-GB RAM. We download a subset of the pollution data from the website of City Pulse [31] as our testing data. The subset includes 10^6 records. Each pollution record consists of the pollution data of four types of pollutants, which are *ozone*, *nitrogen dioxide*, *sulfur dioxide,* and *PM 10 particles*. According to the *air quality index* metric of the United Kingdom [32], each pollutant has four grades, which are *low*, *moderate*, *high,* and *very high*. Hence, we extract four keywords from each pollution record. Each of which consists of a pollutant name and the corresponding pollution grade. For example, keyword *ozone low* denotes the pollutant *ozone* with the *low* grade. All extracted keywords will be encrypted or searched by XW15 and LSPE. Clearly, the keyword space $\mathcal{W}$ includes 16 keywords in total.

Time cost of encryption: Raspberry Pi device is applied for running XW15 and LSPE to encrypt all extracted keywords of the above testing data. When generating a given amount of ciphertexts, Fig. 4 shows the average time cost to generate one

TABLE IV
NUMBER OF MATCHING CIPHERTEXTS OF EACH KEYWORD

Keyword	Amount
Ozone low	42 023
Ozone moderate	36 875
Ozone high	21 102
Nitrogen dioxide low	95 658
Nitrogen dioxide moderate	4342
Sulfur dioxide low	100 000
PM 10 particles low	23 232
PM 10 particles moderate	16 267
PM 10 particles high	13 525
PM 10 particles very high	46 976
The other keywords have no matching ciphertext.	

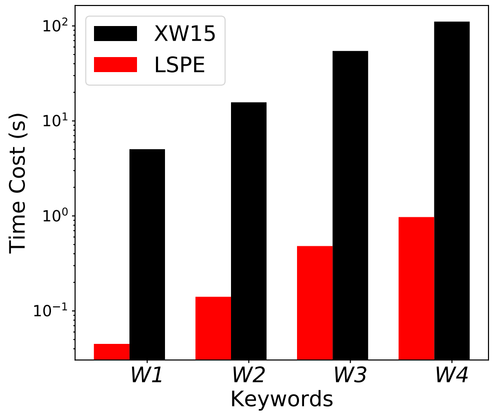

Fig. 5. Time cost of search.

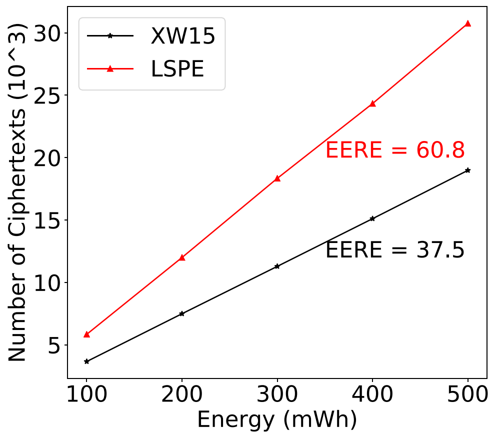

Fig. 6. Energy cost of encryption.

ciphertext by XW15 and LSPE. For example, consider the case of generating 9000 ciphertexts. XW15 takes an average time of 53.3 ms to generate one ciphertext, whereas LSPE only needs 34.3 ms. Therefore, LSPE saves approximately 35% time cost to generate one ciphertext. Moreover, following the increase in the amount of generated ciphertexts, LSPE will save more time cost compared to XW15 for generating one ciphertext.

Time cost of search: According to our testing data, there are 4×10^6 keyword-searchable ciphertexts in total. Table IV lists the number of matching ciphertexts of each keyword. Consider searching keywords *W1* = *nitrogen dioxide moderate*, *W2*=*PM 10 particles high*, *W3* = *PM 10 particles very high,* and *W4* = *nitrogen dioxide low* by XW15 and LSPE. To search keywords *tW1*, *W2*, *W3,* and *W4*, Fig. 5 shows that XW15 takes 5.04, 15.7, 54.54, and 111 s, respectively, and LSPE takes 0.045, 0.141, 0.482, and 0.973 s, respectively. It is clear that the time cost of LSPE is approximately 113 times less than that of XW15 for searching all keywords.

In addition, according to the time cost of LSPE to search keyword *W4*, we find that the average time cost to find one matching ciphertext is approximately 10 μs. Referring to an SSE scheme with practical search performance [26], the practical search performance to find one matching ciphertext should be approximately 7.3 μs. Hence, the search performance of LSPE is very close to that of the practical SSE scheme. To the best of our knowledge, LSPE is the first SPE scheme having the practical search performance.

Energy cost of encryption: Sensors generally have the limited energy in practice [33]. Therefore, the energy cost of sensors to run XW15 and LSPE is also an important index for measuring the practicality of the two schemes. We investigate the energy costs of both XW15 and LSPE's encryption algorithms, and the results are shown in Fig. 6. To explain the advantage of LSPE in more general terms, we define energy efficiency ratio of encryption (EERE) as equation EERE = Ciphertext Number/Energy Cost to measure the energy efficiency. For example, taking 500 mWh of energy (including the energy cost of the operating system of Raspberry Pi), XW15 can generate 18 976 ciphertexts, and LSPE can generate 30 748 ciphertexts, which is much more than that of XW15. In addition, our EERE results in Fig. 6 show that LSPE increases the EERE ratio by 62% compared with XW15. Hence, LSPE is considerably more efficient in terms of the energy cost to generate ciphertexts than XW15.

VI. OTHER RELATED WORKS

In 2004, the first SPE scheme was proposed by Boneh *et al.* [24], and it is called public-key encryption with keyword search (PEKS). Following this seminal work, numerous researchers have devoted efforts to constructing PEKS schemes with different properties. Currently, most PEKS schemes can be categorized into the following four types.

Standard PEKS: Abdalla *et al.* [34] redefine the correctness of PEKS, and introduce a general transformation from an anonymous identity-based encryption (IBE) scheme to a PEKS scheme. They also construct a PEKS scheme with temporary keyword search by the hierarchical IBE. To resist keyword guessing attack launched by a malicious server, Chen *et al.* [35] propose a new general framework for PEKS, which is named dual-server PEKS. However, this paper requires that keyword searches must be performed by by two servers. To avoid this requirement, Chen *et al.* [36] propose another new framework for PEKS, which is named the server-aided PEKS.

PEKS with functional search: To make PEKS versatile, numerous efforts have been devoted to constructing a PEKS scheme with functional search. Song *et al.* [37] propose an efficient conjunctive keyword search scheme without keyword fields. Wang *et al.* [38] describe a new construction for a PEKS scheme to support range search. Zhang *et al.* [39] achieve disjunctive and conjunctive keyword search. Zhu *et al.* [40] propose a fuzzy keyword search scheme. To ensure that the search results returned from a honest-but-curious server are authentic, Zheng *et al.* [41] propose a verifiable attribute-based keyword search over outsourced ciphertexts. This paper is extended to the multiowner setting by Miao *et al.* [42].

PEKS with fast keyword search: The above PEKS schemes take search time that is linear with the total number of ciphertexts. This feature makes them difficult to apply in the scenario of a large-scale database. Bellare *et al.* [43] propose a deterministic PEKS scheme to realize efficient keyword search. In their scheme, the security is formalized as the notion of "as strong as possible," which is stronger than onewayness but weaker than semantic security. The later deterministic PEKS scheme proposed by Brakerski and Segev [44] has a better security, but it still cannot guarantee semantic security. Tseng *et al.* [45] propose an interactive construction named iPEKS for fast keyword search. The more the keywords have been searched previously, the better the efficiency can be improved.

Applications of PEKS: There are several research works focusing on applying PEKS in various scenarios. Ma *et al.* [46] propose a secure channel free certificateless PEKS scheme for IIoT. Wu *et al.* [47] take into consideration the limitations of SSE and PEKS, and introduce an efficient and secure searchable encryption protocol for cloud-assisted IoT. Zhang *et al.* [48] attempt to apply SE in secure biometric authentication, and construct a secure biometric authentication scheme based on PEKS. In their scheme, the biological template is encrypted as searchable ciphertext, and the authentication process is transformed into a keyword search over encrypted database.

VII. Conclusion

In this paper, we propose a lightweight and semantically secure SPE scheme called LSPE for the scenario of CWSNs. In contrast to the previous work XW15, LSPE avoids implementing too many computation-intensive cryptographic operations. Although LSPE still has the same search complexity as XW15, LSPE is considerably more efficient in practice than XW15 in terms of the time and energy costs to generate ciphertexts and the time cost to search keywords. Compared with XW15, our experimental results show the following.

1) LSPE saves approximately 35% of the time cost to generate one ciphertext.
2) The time cost of LSPE to find one matching ciphertext is approximately 113 times less than that of XW15.
3) LSPE increases the energy efficiency to generate ciphertexts by 62%. In addition, LSPE is the first SPE scheme to obtain search performance that is as efficient as a practical SKE scheme.

References

[1] C. Cheng, N. Ganganath, and K. Fok, "Concurrent data collection trees for IoT applications," *IEEE Trans. Ind. Inform.*, vol. 13, no. 2, pp. 793–799, Apr. 2017.

[2] C. D'Orazio, K-K. R. Choo, and L. T. Yang, "Data exfiltration from Internet of Things devices: iOS devices as case studies," *IEEE Internet Things J.*, vol. 4, no. 2, pp. 524–535, Apr. 2017.

[3] GE Digital, "Everything you need to know about the industrial Internet of Things." 2017. [Online]. Available: https://www.ge.com/digital/blog/everything-you-need-know-about-industrial-internet-things

[4] Grand View Research, "Industrial IoT market size worth $933.62 Billion by 2025 | CAGR: 27.8%." 2017. [Online]. Available: http://www.grandviewresearch.com/press-release/global-industrial-internet-of-things-iiot-market

[5] L. Mainetti, L. Patrono, and A. Vilei, "Evolution of wireless sensor networks towards the Internet of Things: A survey," in *Proc. Int. Conf. Softw., TeleCommun. Comput. Netw.*, 2011, pp. 1–6.

[6] A. Aburumman and K-K. R. Choo, "A domain-based multi-cluster SIP solution for mobile ad hoc network," in *Proc. Int. Conf. Security Privacy Commun. Syst.*, 2015, vol. 153, pp. 267–281.

[7] A. Aburumman, W. J. Seo, R. Islam, M. K. Khan, and K-K. R. Choo, "A secure cross-domain SIP solution for mobile ad hoc network using dynamic clustering," in *Proc. Int. Conf. Security Privacy Commun. Netw.*, 2016, vol. 164, pp. 649–664.

[8] N. Khalil, M. R. Abid, D. Benhaddou, and M. Gerndt, "Wireless sensors networks for Internet of Things," in *Proc. IEEE 9th Int. Conf. Intell. Sensors, Sensor Netw. Inf. Process.*, 2014, pp. 1–6.

[9] X. Jian, G. Yang, Z. Chen, and Q. Wang, "A survey on the privacy-preserving data aggregation in wireless sensor networks," *China Commun.*, vol. 12, no. 9, pp. 162–180, 2015.

[10] C. Huang, R. Lu, and K-K. R. Choo, "Vehicular fog computing: architecture, use case, and security and forensic challenges," *IEEE Commun. Mag.*, vol. 55, no. 11, pp. 105–111, Nov. 2017.

[11] X. Chen, J. Li, J. Weng, J. Ma, and W. Lou, "Verifiable computation over large database with incremental updates," *IEEE Trans. Comput.*, vol. 65, no. 10, pp. 3184–3195, Oct. 2016.

[12] J. Wang, X. Chen, X. Huang, I. You, and Y. Xiang, "Verifiable auditing for outsourced database in cloud computing," *IEEE Trans. Comput.*, vol. 64, no. 11, pp. 3293–3303, Nov. 2015.

[13] H. Xiong, Z. Chen, and F. Li, "Efficient and multi-level privacy-preserving communication protocol for VANET," *Comput. Elect. Eng.*, vol. 38, no. 3, pp. 573–581, May 2012.

[14] Y. Fan, H. Chen, and X. Zhang, "Data privacy preservation in wireless sensor networks," *Chin. J. Comput.*, vol. 35, no. 6, pp. 1131–1146, 2012.

[15] X. Zhao, J. Zhu, X. Liang, S. Jiang, and Q. Chen, "Lightweight and integrity-protecting oriented data aggregation scheme for wireless sensor networks," *IET Inf. Security*, vol. 11, no. 2, pp. 82–88, 2017.

[16] X. Wang and Z. Zhang, "Data division scheme based on homomorphic encryption in WSNs for health care," *J. Med. Syst.*, vol. 39, no. 12, pp. 188:1–188:7, 2015.

[17] Q. Wang, C. Yu, F. Li, H. Wang, and L. Cao, "A group key-policy attribute-based encryption with partial outsourcing decryption in wireless sensor networks," *Security Commun. Netw.*, vol. 9, no. 17, pp. 4138–4150, 2016.

[18] M. Elhoseny, X. Yuan, H. K. El-Minir, and A. M. Riad, "An energy efficient encryption method for secure dynamic WSN," *Security Commun. Netw.*, vol. 9, no. 13, pp. 2024–2031, 2016.

[19] Y. Lu, J. Zhai, R. Zhu, and J. Qin, "Study of wireless authentication center with mixed encryption in WSN," *J. Sensors*, vol. 2016, 2016, Art. no. 9297562. doi: 10.1155/2016/9297562.

[20] S. Chen, M. Tuan, H-Y. Lee, and T. Lin, "VLSI implementation of a cost-efficient micro control unit with an asymmetric encryption for wireless body sensor networks," *IEEE Access*, vol. 5, pp. 4077–4086, 2017.

[21] T. Hwang and P. Gope, "Robust stream-cipher mode of authenticated encryption for secure communication in wireless sensor network," *Security Commun. Netw.*, vol. 9, no. 7, pp. 667–679, 2016.

[22] G. S. Poh, J-J. Chin, W-C. Yau, K-K. R. Choo, and M. S. Mohamad, "Searchable symmetric encryption: Designs and challenges," *ACM Comput. Surv.*, vol. 50, no. 3, 2017.

[23] P. Xu, S. Liang, W. Wang, W. Susilo, Q. Wu, and H. Jin, "Dynamic searchable symmetric encryption with physical deletion and small leakage," in *Proc. Aust. Conf. Inf. Security Privacy*, 2017, vol. 10342, pp. 207–226.

[24] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *Proc. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, vol. 3027, pp. 506–522.
[25] 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 Security*, vol. 10, no. 9, pp. 1993–2006, Sep. 2015.
[26] S. Kamara, C. Papamanthou, and T. Roeder, "Dynamic searchable symmetric encryption," in *Proc. ACM Conf. Comput. Commun. Security*, 2012, pp. 965–976.
[27] B. Lynn, "PBC library." 2006. [Online]. Available: https://crypto.stanford.edu/pbc/
[28] Microsoft, "Suggested boards and SoCs." 2017. [Online]. Available: https://docs.microsoft.com/en-us/windows/iot-core/learn-about-hardware/suggestedboards
[29] Y. Wang, J. Wang, and X. Chen, "Secure searchable encryption: A survey," *J. Commun. Inf. Netw.*, vol. 1, no. 4, pp. 52–65, 2016.
[30] Raspberry Pi Foundation, "RASPBERRY PI 3 MODEL B." 2017. [Online]. Available: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/
[31] CityPulse, "Dataset collection." 2017. [Online]. Available: http://iot.ee.surrey.ac.uk:8080/index.html
[32] Wikipedia, "Air quality index." 2017. [Online]. Available: https://en.wikipedia.org/wiki/Air_quality_index
[33] F. Luo, C. Jiang, H. Zhang, X. Wang, L. Zhang, and Y. Ren, "Node energy consumption analysis in wireless sensor networks," in *Proc. IEEE Veh. Technol. Conf.*, 2014, pp. 1–5.
[34] M. Abdalla *et al.*, "Searchable encryption revisited: Consistency properties, relation to anonymous IBE, and extensions," in *Proc. Int. Cryptol. Conf.*, 2005, vol. 3621, pp. 205–222.
[35] R. Chen, Y. Mu, G. Yang, F. Guo, and X. Wang, "A new general framework for secure public key encryption with keyword search," in *Proc. Aust. Conf. Inf. Security Privacy*, 2015, vol. 9144, pp. 59–76.
[36] R. Chen *et al.*, "Server-aided public key encryption with keyword search," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2833–2842, Dec. 2016.
[37] C. Song, X. Liu, and Y. Yan, "Efficient public key encryption with field-free conjunctive keywords search," in *Proc. Int. Conf. Trusted Syst.*, 2014, vol. 9473, pp. 394–406.
[38] B. Wang, Y. Hou, M. Li, H. Wang, and H. Li, "Scalable multi-dimensional range search over encrypted cloud data with tree-based index," in *Proc. ACM Symp. Inf., Comput. Commun. Security*, 2014, pp. 111–122.
[39] Y. Zhang and S. Lu, "Efficient method for disjunctive and conjunctive keyword search over encrypted data," in *Proc. ACM Conf. Comput. Commun. Security*, 2014, pp. 1535–1537.
[40] H. Zhu, Z. Mei, B. Wu, H. Li, and Z. Cui, "Fuzzy keyword search and access control over ciphertexts in cloud computing," in *Proc. Aust. Conf. Inf. Security Privacy*, 2017, vol. 10342, pp. 248–265.
[41] Q. Zheng, S. Xu, and G. Ateniese, "VABKS: Verifiable attribute-based keyword search over outsourced encrypted data," in *Proc. IEEE Conf. Comput. Commun.*, 2014, pp. 522–530.
[42] Y. Miao, J. Ma, X. Liu, J. Zhang, and Z. Liu, "VKSE-MO: Verifiable keyword search over encrypted data in multi-owner settings," *Sci. China Inf. Sci.*, vol. 60, no. 12, pp. 122105:1–122105:15, 2017.
[43] M. Bellare, A. Boldyreva, and A. ONeill, "Deterministic and efficiently searchable encryption," in *Proc. Int. Cryptol. Conf.*, 2007, vol. 4622, pp. 535–552.
[44] Z. Brakerski and G. Segev, "Better security for deterministic public key encryption: The auxiliary-input setting," in *Proc. Int. Cryptol. Conf.*, 2011, vol. 6841, pp. 543–560.
[45] F. Tseng, R. Chen, and B. P. Lin, "iPEKS fast and secure cloud data retrieval from the public-key encryption with keyword search," in *Proc. IEEE Int. Conf. Trust, Security Privacy Comput. Commun.*, 2013, pp. 452–458.
[46] M. Ma, D. He, N. Kumar, K-K. R. Choo, and J. Chen, "Certificateless searchable public key encryption scheme for industrial Internet of Things," *IEEE Trans. Ind. Informat.*, to be published. doi: 10.1109/TII.2017.2703922.
[47] L. Wu, B. Chen, K-K. R. Choo, and D. He, "Efficient and secure searchable encryption protocol for cloud-based Internet of Things," *J. Parallel Distrib. Comput.*, vol. 111, pp. 152–161, 2018.
[48] Y. Zhang, J. Qin, and L. Du, "A secure biometric authentication based on PEKS," *Concurrency Comput., Pract. Exp.*, vol. 28, no. 4, pp. 1111–1123, 2016.

Peng Xu (M'13) received the B.E. degree in computer science from Wuhan University of Science and Technology, Wuhan, China, in 2003, the Master's and Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2006 and 2010, respectively.

Since 2010, he has been a Postdoctor with Huazhong University of Science and Technology. He was a PI in four grants from National Natural Science Foundation of China, China Postdoctoral Science Foundation, and Shenzhen Fundamental Research Program, respectively. He also was a key member in several projects supported by 973 Program. He has authored more than 30 research papers.

Dr. Xu is a member of ACM.

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

His research interests include cloud security and cryptography.

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

She is currently a Researcher with the Cyber-Physical-Social Systems Lab, Huazhong University of Science and Technology, Wuhan, China. She has authored more than ten papers in international journals and conferences. Her research interests include cloud security, network coding, and multimedia transmission.

Willy Susilo (SM'01) received the Ph.D. degree in computer science from the University of Wollongong, Wollongong, NSW, Australia, in 2001.

He is currently a Professor and the Head of the School of Computing and Information Technology, University of Wollongong, where he is also the Director of the Centre for Computer and Information Security Research. His research interests include cryptography and information security. He was the author or coauthor of numerous publications in the area of digital signature schemes and encryption schemes.

Prof. Susilo was the recipient of the prestigious ARC Future Fellowship awarded by the Australian Research Council. He was a program committee member in dozens of international conferences.

Hai Jin (SM'01) received the Ph.D. degree in computer engineering from Huazhong University of Science and Technology, Wuhan, China, in 1994.

He was with The University of Hong Kong from 1998 to 2000, and a Visiting Scholar with the University of Southern California from 1999 to 2000. He is the Chief Scientist of National 973 Basic Research Program Project of Virtualization Technology of Computing System. He has coauthored 15 books and authored or coauthored more than 400 research papers.

Prof. Jin was the recipient of the Excellent Youth Award from the National Science Foundation of China in 2001. In 1996, he received the German Academic Exchange Service Fellowship to visit the Technical University of Chemnitz, Chemnitz, Germany. He is a member of the ACM.