A smart contracts-based searchable encryption scheme with forward privacy

Junjian Yan*, Zixuan Fang, Qizhi He, Yanfei Lu
College of Computer Science and Engineering, Northwest Normal University, Lanzhou 730070, Gansu, China

ABSTRACT

As an important link of the Internet of Things, wireless sensor network has great potential in industry, agriculture and daily use of residents. To ensure cloud data security, data is encrypted before being uploaded to the cloud. However, how to ensure the searchability of ciphertext in cloud is challenging. To solve the above problems, a searchable encryption scheme for sensor networks is proposed. The sensor node collects the data, encrypts it and uploads it to the smart contract after collecting the data from the local gateway. In this scheme, the concept of version information is introduced to authenticate ciphertext version so as to realize access control of trap gates of different versions. In order to avoid data leakage caused by the information storage center, the searchable encryption technology is combined with the block chain technology, and the smart contract is used as the data storage center, so as to ensure efficient ciphertext retrieval and eliminate malicious behavior on the server side. Through security analysis, it is proved that the scheme satisfies ciphertext indistinguishability and forward security based on DBDH hard problems. Experimental efficiency analysis shows that the scheme has certain advantages in computational efficiency.

Keywords: Searchable encryption, smart contracts, forward safety, public key encryption

1. INTRODUCTION

With the advancement of Industry 4.0, Internet of Things (IoT) technology is rapidly developing. The use of wireless sensor detection to generate large amounts of data and automated processing using pre-written script codes can significantly reduce labor costs, and this new wave of industry will contribute to the formation of a new era of technological development and economic growth. Among others, sensor networks play an important role in an increasing number of areas, such as the Internet of Vehicles[1], the Industrial Internet of Things[2], smart medical systems[3], smart homes[4] environmental analysis, and other aspects.

In the industrial IoT environment, sensitive information may be generated due to the work of Wireless Sensor Networks (WSNs). The need for commercial confidentiality requires encryption of some information transmission processes in sensor networks. Numerous studies have introduced cryptographic algorithms into WSNs for protecting the privacy of the information transmission process[5-6]. When the number of sensor nodes in WSNs is too large, the symmetric cryptographic regime suffers from serious key management problems. Therefore, some studies have applied public-key cryptography to industrial IoT environments where there are a large number of sensor nodes. However, most current public-key cryptography algorithms have the problem of large cryptographic operation overhead in practical use, which is difficult to operate efficiently for resource-constrained WSNs.

In the public-key WSNs model, data are generated by numerous sensor nodes, encrypted using the public key and uploaded to the information hub, and the data are searched by the data users by generating trapdoors. In the Public-key Encryption with Keyword Search (PEKS) algorithm model, a cloud server is usually used as a storage ciphertext information hub, and cloud servers are usually considered to be honest and curious. Therefore, searchable encryption schemes using non-distributed cloud servers cannot resist from internal keyword guessing attacks. To solve the above problem, DU et al.[7] proposed blockchain-based searchable cryptosystem using smart contracts as ciphertext storage servers, which can be used as trusted third parties for retrieval work in PEKS schemes due to their immutability and transparency. Stefanov et al.[8] and Hoang et al.[9] proposed schemes in the reference to satisfy forward security Song et al.[10] propose forward-privacy and

* 458838374@qq.com

Third International Conference on Computer Science and Communication Technology (ICCSCT 2022)
edited by Yingfa Lu, Changbo Cheng, Proc. of SPIE Vol. 12506, 125061O
© 2022 SPIE · 0277-786X · doi: 10.1117/12.2662013

efficient scalable schemes with high I/O efficiency, but there is a risk of being guessed by internally colluding cloud servers because they do not use a distributed third party to store ciphertexts for them.

Our contributions: On this basis, we propose a searchable encryption scheme for smart contract-based wireless sensor networks to satisfy forward security. Distributed smart contracts are used as information storage institutions, and data is processed and uploaded by automatically executed scripts. The scheme meets forward security and ensures that the old trapdoor cannot search the database after data update. The implicit structure[11] is used to optimize the query efficiency, and the forward security is realized by using the characteristics of pseudo-random substitution function[12], so that the algorithm has efficient I/O performance, the load of the algorithm is optimized in a large number of queries, and the sublinear search complexity is realized.

In terms of security, our scheme satisfies ciphertext indistinguishable security under the Decisional Bilinear Diffie-Hellman (DBDH) assumption in the random oracle model. The experimental comparison and analysis with the schemes of scheme[13] and scheme[14] show that the scheme is more efficient in terms of search and update operations compared to the comparison scheme.

2. PRELIMINARIES

2.1 Symbol description

In this section, we will introduce the relevant background knowledge, DBDH problem, and other knowledge. Some of the required identifiers and their meanings in this paper are shown in Table 1.

Table 1. Summary of notations.

Notations	Descriptions
λ	The system security parameter
G	An elliptic curve group with order p
g	A generator of G
PK_{DO}, SK_{DO}	Public and private key pairs of the data owner
PK_{DU}, SK_{DU}	Public and private key pairs of the data user
$ind_{\omega i}$	The index of i-th files containing ω
ED_{ω}	The encrypted DB(ω)
CV_{ω}	The ciphertext of version information of ω
ψ	Mapping of key-value

2.2 Bilinear pairing

Let G_1 and G_2 be two cyclic bilinear groups of the same order q, and e be a computable bilinear map $e: G_1 \times G_1 \to G_2$ with the following properties:

Bilinearity: For any $x, y \in G_1$ and $a, b \in Z_q^*$, we have $e(x^a, y^b) = e(x, y)^{ab}$.

Non-degeneracy: For any generator $x, y \in G_1$, $e(x, y) \neq 1$.

Computability: For any $x, y \in G_1$, $e(x, y)$ can be efficiently computed.

2.3 Ethernet-based smart contracts

Ethernet-based smart contracts: Ethernet is a decentralized computing platform whose security is maintained by encrypted blocks. A distributed computing center with decentralized characteristics must have a consensus mechanism, i.e. the entire network agrees on the rules for each transaction and block. Once a message is created and deployed to Ether, the code of the contract is retained permanently and cannot be changed[15].

2.4 Hidden structure

The concept of hidden structure is proposed in XU et al.[11]. The hidden structure is an open node that can point to a set of keyword ciphertext indexes, as shown in Figure 1. To some extent, the hidden structure can improve search efficiency.

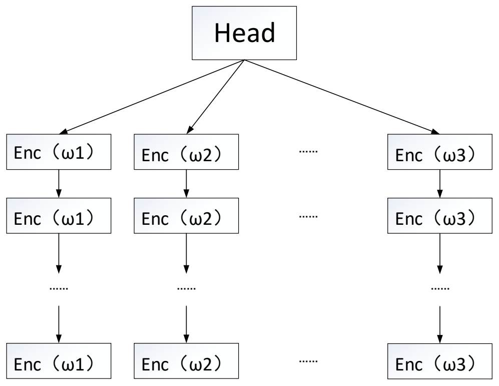

Figure 1. Hidden structure.

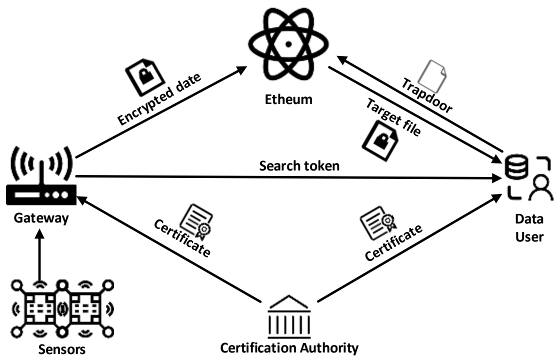

Figure 2. System model.

Definition 1: DBDH problem: The input tuple $(g, g^a, g^b, g^c) \in G_1$, where g is the generator of G_1, (a,b,c) and are chosen randomly by Z_q^*. One needs to distinguish $e(g,g)^{abc}$ from a random element T of G_2.

Definition 2: DBDH assumption: The problem assumes that any PPT algorithm A cannot solve the DBDH problem with a non-negligible advantage: $\Pr[A(g, g^a, g^b, g^c, e(g,g)^{abc}) = 1] - \Pr[A(g, g^a, g^b, g^c, T) = 1] | \le \varepsilon$,where $g \in G_1$, $(a,b,c) \in Z_q^*$, $T \in G_2$, All of the above elements are chosen at random. Let $Adv_A^{DBDH}(\lambda)$ be the advantage of algorithm A solving the DBDH problem, and if the DBDH assumption holds, then $Adv_A^{DBDH}(\lambda) \le \varepsilon$, where ε is a negligible probability.

3. PROPOSED SCHEME

In this section, we present the detailed algorithm of the article, using Ether as a cloud server for storing information, which consists of the following four main entities with the model diagram in Figure 2.

• Data owner (DO): The smart gateway acts as the data owner, aggregates the information of each sensor node, executes a predefined script for processing and encryption, sends the cipher text to the smart contract, and sends the version information to the corresponding user.

• Data User (DU): Data users who are authorized to search for the corresponding keywords can generate a trapdoor to submit search queries to the system.

• Smart Contract (SC): A cloud platform that stores ciphertext files and retrieves trapdoors for data users, with fairness and verifiability.

• Certificate Authority (CA): responsible for generating public-private key pairs and issuing integers for trusted users.

• Wireless Sensors (WS): senses information and sends the captured information to the local smart gateway using the intranet.

3.1 Setup

System input the security parameters λ, and output the system public parameter(PP) $PP=\{q,G_1,G_2,g,e,h_1,h_2,h_3,h_4,h_5,h_6,F/F^{-1}\}$, where G_1, G_2 is a bilinear cyclic group with the order q. h_1-h_6 are cryptographically secure hash functions, which $h_1:\{0,1\}^{\lambda}\leftarrow\{0,1\}^{\lambda}$, $h_2:\{0,1\}^{3\lambda}\leftarrow\{0,1\}^{\lambda}$, $h_3:Z_q^*\leftarrow\{0,1\}^{\lambda}$, $h_4:\{0,1\}^{2\lambda}\leftarrow G_2$, $h_5:\{0,1\}^{\lambda}\leftarrow G_2$, $h_6:\{0,1\}^{\lambda}\leftarrow G_1$. F/F^{-1} are pseudo-random permutation and its inverse that satisfies $F:(K\times X)\rightarrow Y$, $F:(K\times Y)\rightarrow X$, which K is a secret key. Initialize the smart contract, the data owner DO sets the retrieval unit price \$offer, the data user DU registers the account \$user and makes a deposit, and the smart contract system sets the deposit account \$deposit.

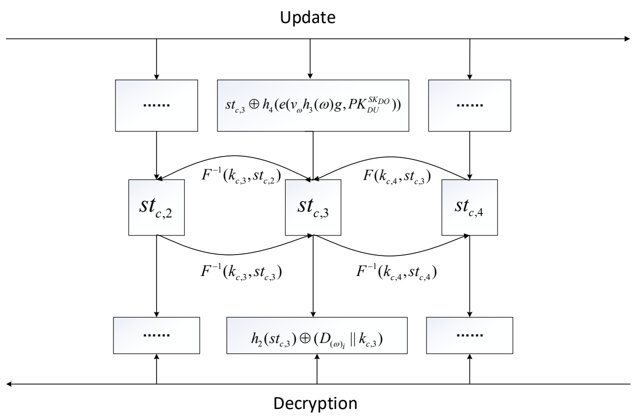
Update
......
$st_{c,3}\oplus h_4(e(v_\omega h_3(\omega)g,PK_{DU}^{SK_{DO}}))$
......
$F^{-1}(k_{c,3},st_{c,2})$
$F(k_{c,4},st_{c,3})$
$st_{c,2}$
$st_{c,3}$
$st_{c,4}$
$F^{-1}(k_{c,3},st_{c,3})$
$F^{-1}(k_{c,4},st_{c,4})$
......
$h_2(st_{c,3})\oplus(D_{(\omega)_i}\,||\,k_{c,3})$
......
Decryption

Figure 3. Update and decryption process diagram.

3.2 Key generation

Input the public parameters and randomly select $a,b\in Z_q^*$, to generate the DO's public key/private key pairs (PK_{DO},SK_{DO}) and DU's public key/private key pairs (PK_{DU},SK_{DU}):

$$PK_{DO}=g^a,SK_{DO}=a$$

$$PK_{DU}=g^b,SK_{DU}=b$$

3.3 Data encryption

The system inputs the public parameters PP, DO's private key SK_{DO}, DU's public key PK_{DU}, keywords ω, index set $D(\omega)$ containing the keywords ω, where $l=|D(\omega)|$, the update and decryption process with hidden structure is shown in Figure 3, and the following algorithm is executed.

(1) Initialization status information $st_0\leftarrow\{0,1\}^{\lambda}$, $C=0$.

(2) For each $D(\omega)_i$, $(i=1,\ldots,l)$, select $k_{c,i}\in\{0,1\}^{\lambda}$, compute the state information $st_{c,i}=F(k_{c,i},st_{c,i-1})$, $e_{c,i}=h_2(st_{c,i-1})\oplus(D(\omega)_i\|k_{c,i})$ and $u_{c,i}=h_1(st_{c,i})$.

(3) Compute first-node information $FI = (fi_1, fi_2)$, $fi_1 = st_{c,l} \oplus h_4\left(e\left(v_\omega h_3(\omega) g, PK_{DU}^{SK_{DO}}\right)\right)$, $fi_2 = h_5\left(e\left(v_\omega h_3(\omega) g, PK_{DU}^{SK_{DO}}\right)\right)$, which $v_\omega \xleftarrow{R} Z_q^*$.

(4) DO compute the version information $CV_\omega = (CV_{\omega 1}, CV_{\omega 2})$ of keyword ω , which $CV_{\omega 1} = g^m$, $CV_{\omega 2} = (\omega \| v_\omega) \oplus h_6(PK_{DU}^m)$, $m \xleftarrow{R} Z_q^*$. Then DO sends CV_ω to DO.

(5) DO send encryption data $ED_\omega = \{(st_{c,i}, e_{c,i}), FI\}$ to smart contract, which $i = 1, \ldots, l$.

3.4 Trapdoor

DO receives CV_ω and computes $h_6(SK_{DU} CV_{\omega 1}) \oplus CV_{\omega 2}$ to get keyword ω and version information v_ω .Then, DO computes trapdoor $T_\omega = \{h_3^{SK_{DU}}(\omega) v_\omega g\}$,which $t \xleftarrow{R} Z_q^*$. The DU sends the trapdoor T_ω to the smart contract and pays the deposit \$user=\$user -\$deposit. The smart contract system receives the trapdoor T_ω and accepts the deposit.

3.5 Test

DU takes the public parameters PP , encryption date ED_ω, trapdoor T_ω ω of as inputs, compute keyword results R_ω. The test process is as follows:

(1) Compute $h_{i_2} = h_5\left(e\left(v_\omega h_3^{SK_{DU}}(\omega) g, PK_{DO}\right)\right) = h_5\left(e\left(v_\omega h_3(\omega) g, PK_{DU}^{SK_{DO}}\right)\right)$ .Then, inpute h_{i_2} into mapping ψ to obtain $\psi(h_{i_2}) = h_{i_1}$.

(2) If h_{i_1} does not exist, the algorithm is terminated, otherwise compute $st = h_4\left(e\left(v_\omega h_3(\omega) g, PK_{DU}^{SK_{DO}}\right)\right) \oplus h_{i_1}$.

(3) DU computes $u^* = h_1(st)$, and $\psi(u^*) = e_{c,i}^* = h_2(st_{c,i}) \oplus \left(D(\omega)_i \| k_{c,i}\right)$.

(4) If $e_{c,i}^*$ exists, then compute $e_{c,i}^* \oplus h_2(st_{c,i})$ to obtain k and $D(\omega)_i$ .Put $D(\omega)_i$ into result set R_ω, then compute $st_{c,i-1} = F^{-1}(k, st_{c,i})$ and return step 3), Send R_ω to DU until the keyword is completely retrieved.

The smart contract will perform the retrieval operation of the next keyword in the retrieved files and deduct the corresponding search unit price \$offer from the deposit account until the query is completed or the amount of the deposit account \$deposit is insufficient for a search. The smart contract system will return the information of the remaining deposit and insufficient deposit to the user: \$deposit← \$deposit − \$offer.

4. ANALYSIS

4.1 Security Analysis

Game 1:

Setup: Challenger *C* inputs the secure parameter, outputs public parameter PP ; ψ is a mapping of key-value pair; *C* sends PP to adversary *A*.

KeyGen: Challenger *C* generates the public and private key pair of the data owner (PK_{DO}, SK_{DO}) and data user (PK_{DU}, SK_{DU}) , and sends their public keys to Adversary *A*. Public and private key is generated as follows: $PK_{DO} = g^a$, $SK_{DO} = a$; $PK_{Du} = g^b$, $SK_{Du} = b$,which $a, b \in Z_q^*$.

Phase 1: The adversary *A* makes the following multiple quiries, each key can only be queried once.

Update Oracle: A sends the query of keyword ω, C performs the following steps:

C retrieves the status (st_c, c), if (st_c, c) is null, set $C=0$, C randomly selects $st_{c,0} \xleftarrow{R} \{0,1\}^{\lambda}$. For the current database version, C randomly selects $k_{c,i} \xleftarrow{R} \{0,1\}^{\lambda}$, $i=\{1,...,l\}$, and computes the next status value $st_{c,1}=F(k_{c,1}, st_{c,0}),...,st_l=F(k_{c,l}, st_{c,l-1})$, $e_{c,j}=(D_{\omega,j} \| st_{c,j}) \oplus h_2(k_c, j)$, $u_{c,j}=h_2(k_{c,j})$, sets $\psi(u_{c,j})=e_{c,j}$, $ED_{\omega}=\{u_{c,j}, e_{c,j}\}$.

Inorder to computes first-node information $FI=(fi_1, fi_2)$, C first randomly selects $V_{\omega} \xleftarrow{R} Z_q^*$, then computes $fi_1=uh_{\omega}=h_5\{e(PK_{DU}^{SK_{DO}}, PV_{\omega})^{h_3(\omega)}\}$, $fi_2=eh_{\omega}=\{(st_{c,l}) \oplus h_4(e(PK_{DU}^{SK_{DO}}, PV_{\omega})^{h_3(\omega)})\}$. Finally, C sets $\psi[fi_1]=fi_2$.

C randomly selects $m \xleftarrow{R} Z_q^*$, and compute $CV_1=(\omega \| V_{\omega}) \oplus h_6(\mathrm{PK}_{DU}^m)$, $CV_2=P^m$, then C can get $CV=(CV_1, CV_2)$. C sends (ED_{ω}, FI, CV) to smart contract.

Trapdoor Oracle: A sends the keyword ω query to C, and C generates the trapdoor T_{ω} which is return to A.

Query Oracle: A issues a trapdoor T_{ω}, C executes the search algorithm and returns the result set to A.

Challenge: At the end of phase 1, challenger A selects two sets of key-value pairs consisting of data indexes and keywords $(D_{(\omega)_0}, \omega_0)$, $(D_{(\omega)_1}, \omega_1)$, and $D_{(\omega)_0}$, $D_{(\omega)_1}$ are equal in size. Send them to challenger C. Challenger C randomly selects $r \in \{0,1\}$, runs the update algorithm to encrypt $(D_{(\omega)_r}, \omega_r)$, and returns ($ED_{(\omega)_r}$, $HI_{(\omega)_r}$, CV) to A.

Phase 2: Adversary A continues to send queries to the random oracle machine, but the queries cannot include the content of $(D_{(\omega)_0}, \omega_0)$ and $(D_{(\omega)_1}, \omega_1)$.

Guess: Adversary A chooses $r' \in \{0,1\}$, and if $r'=r$, then adversary A wins the ciphertext indistinguishability game; otherwise A loses. Where the probability of A winning in the game is $Adv_A^{GAME1}(\lambda)$.

Game 2:

Challenger C randomly selectes, then computes $T=e(P,P)^{V_{\omega_r}}$, Head Information $HI(\omega_r)=(h_5(T), st^* \oplus h_4(T))$. Assuming that the DBDH problem holds, this round is indistinguishable from Game 1 for attacker A. In other words:

$$| Adv_{game_2}^{\mathrm{A}}(\lambda) - Adv_{game_1}^{A}(\lambda) | \leq Adv_{DBDH}^{A}(\lambda)$$

where $Adv_{DBDH}^{A}(\lambda)$ is a negligible probability.

Proof: If the tuplet $(P, \mathrm{x}P, yP, zP, T)$ satisfies that $(x,y,z) \xleftarrow{R} Z_{\mathrm{q}}^*$, and P is the generator of group G_1, then it is said to be tuplet of DBDH. In Game 1, If $PK_{DO}=xP$, $PK_{DU}=yP$, $h_3(\omega_r)V_{\omega_r}P=zP$, then $e(SK_{DO}PK_{DU}, h_3(\omega_r)V_{\omega_r}P)=e(P,P)^{xyz}$, where $(P, PK_{DO}, PK_{DU}, h_3(\omega_r)V_{\omega_r}P, e(SK_{DO}PK_{DU}, h_3(\omega_r)V_{\omega_r}P))$ is a DBDH tuplet.

Forward Privacy: From the above proof, it is clear that the correct trapdoor can search for the matching keyword sequence, and the algorithm with forward security can guarantee that the update algorithm will not leak the newly inserted information, and in the following we will prove that a keyword is inserted in the above algorithm.

First, system runs the update algorithm, and he encryption scheme as described in the algorithm description. $k_7 \xleftarrow{R} \{0,1\}^{\lambda}$, $st_7=F(k_7, st_6)$, $u_7=h_3(st_7)$, $fi_1=uh_{\omega}=h_5\{e(PK_{DU}^{SK_{DO}}, g^{V_{\omega}^*})^{h_3(\omega)}\}$,

$fi_2 = eh_\omega = \{(st_{c,l}) \oplus h_4\{e(PK_{DU}^{SK_{DO}}, g^{V_\omega^*})^{h_3(\omega)}\}$, $V_\omega^* \xleftarrow{R} Z_q^*$. Then algorithm computes the value of CV : $CV_1 = (\omega \| V_\omega) \oplus h_6(\mathrm{PK}_{DU}^{m^*})$, $CV_2 = g^{m^*}$, $CV = (CV_1, CV_2)$, $m^* \xleftarrow{R} Z_q^*$.

The adversary cannot obtain any information about the updated state st_7 from the previous ciphertext message, as demonstrated in the previous subsection. Therefore, the adversary cannot obtain information about the newly inserted index $D(\omega)$, thus proving the forward security of this scheme.

4.2 Efficiency analysis

In the following, the scheme of this paper is compared with Li et al. [13] and Ma et al.[14] in four aspects: data encryption, trapdoor generation, search keywords, and overall efficiency. In general, bilinear pairs, scalar multiplication operations, and exponential operations have a greater impact on the efficiency of the algorithm, while additive operations, and hash operations have a relatively small impact on the algorithm as a whole due to their extremely low operation time. Table 2 describes the main operational overheads of each scheme, where T_p denotes the bilinear pair operation time, T_h denotes the hash operation time, T_e denotes the exponential operation time, $T_F / T_{F^{-1}}$ denotes the the operation time of the pseudo-random permutation function and its inverse operation time, T_m denotes the multiplication operation and N denotes the time of update.

Table 2. Performance of the compared schemes.

Scheme	Encryption	Trapdoor	Test
Li *et al.* [13]	$N(2T_p + 5T_h + 2T_e + T_{F^{-1}})$	$2T_h + 3T_e$	$5T_p + 4T_e$
Ma *et al.*[14]	$3T_p + 4T_m + 3T_e + T_m$	$2T_m + T_e + T_m$	$6T_p + 2T_e + 4T_m$
Ours	$N(2T_p + 4T_h + 2T_e + T_F)$	$T_h + T_e + T_m$	$2T_p + 5T_h + 2T_e + T_{F^{-1}}$

To verify the actual efficiency of the solution, the following environment is used for simulation and simulation experiments: the experimental environment is Windows 10 operating system (64-bit), Intel® Core(TM) i5-2400 CPU @ 3.10GHz, using a local virtual machine VMware (4GB RAM) running the open source project OpenStack for performance testing, using C++ language for programming and using the Ethernet Rinkeby test network as a testbed to simulate the smart contract environment. In the static encryption phase, we start the encryption keyword from 1 and gradually increase it to 10, and test the efficiency of each of the three schemes in the encryption, trapdoor generation and search phases. In the dynamic update phase, we test the change in performance overhead of the two schemes starting with 50 keywords and in increments of 50, and test the performance change resulting from performing six keyword updates in the same case, respectively.

We tested the time overhead of the three schemes in data encryption, trapdoor generation and keyword search, and the results are shown in Figures 4-6. From Figure 4, it can be seen that our scheme outperforms the schemes in Li *et al.* [13] and Ma *et al.*[14] in data encryption update. When the number of keywords is 10, our scheme takes about 33% less time than Li *et al.* [13] scheme, and almost the same time as Ma *et al.*[14]. As can be seen from Figure 5, our scheme takes much less time in the trapdoor generation phase than Li *et al.* [13] and Ma *et al.*[14], and for generating 10 keywords trapdoors, for example, our time required is only 30% of Ma *et al.*[14] and Li *et al.* [13]scheme. As can be seen from Figure 6, our scheme is about 60% more efficient in the search phase than the Li *et al.* [13], but less efficient than Ma *et al.*[14] scheme. However, since our scheme supports dynamic updates and satisfies forward security, the Ma *et al.*[14] scheme does not support dynamic updates. In summary, our scheme improves the efficiency to some extent and supports dynamic updates to make up for the shortcomings of the scheme in reference[13].

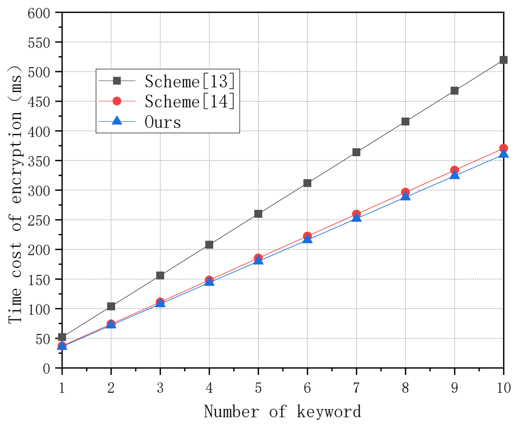

Figure 4. Encryption efficiency diagram.

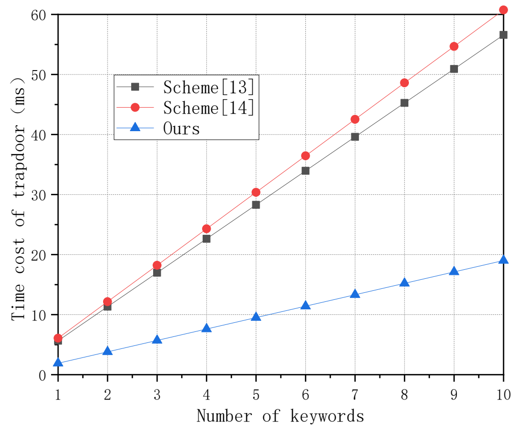

Figure 5. Trapdoor generation efficiency diagram.

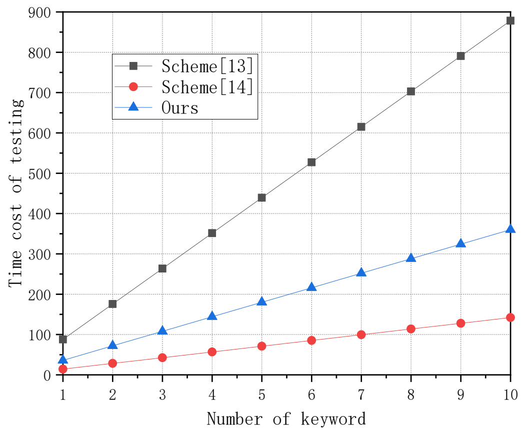

Figure 6. Search efficiency diagram.

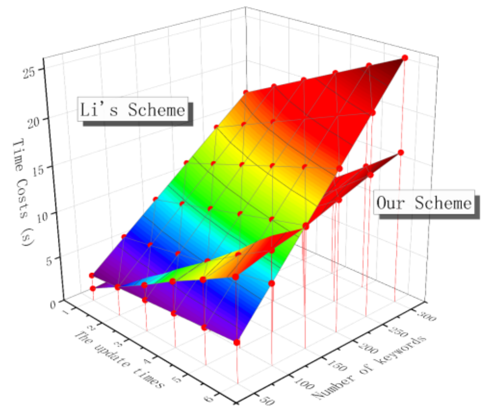

Figure 7. Time cost of encryption algorithm.

During the dynamic update phase, our algorithm is also affected by the updates, but we can complete the bilinear pair computation in the algorithm building phase, where the update operation has less impact on our scheme and the additional overhead incurred does not significantly affect the actual work of the algorithm. We achieve higher efficiency than in the Ma et al.[14] by using a pre-computation approach.

Since the scheme of Li et al.[13] does not support online update, the update test only compares the scheme of Ma et al.[14] with the scheme of this paper. As can be seen from Figure 7, our scheme occupies a relatively lower position after the keywords are 150 in the figure, which indicates that the scheme has a gradual decrease in overhead advantage when the number of keywords is larger. And the efficiency is relatively high when the number of keywords is below 150 and the number of updates is greater than 2. For the same number of additions (6), the improvement is about 975.36 ms compared to the scheme of Ma et al.[14] for encrypting 300 keywords, for example. as the number of keywords and the number of encryptions increase, the overhead of the scheme of Ma et al.[14] will then increase linearly, while our growth is sublinear. Therefore, this scheme is suitable for cases where the number of updates is high and the number of keywords is large.

5. CONCLUSION

In this paper, a lightweight dynamic searchable encryption scheme based on intelligent contract is constructed in wireless sensor network environment, and supports forward security. The scheme improves the retrieval efficiency by introducing the hidden structure, and deploying the ciphertext on the distributed intelligent contract can effectively resist the keyword guessing attack initiated by the internal adversary. the scheme avoids a large number of bilinear pairing operations in the process of construction. and the running cost of the system is maintained at a low level after many updates. Through the experimental analysis, the efficiency of encryption and update is slightly higher than that of the contrast scheme, which improves the practicability of the scheme. The next step is to consider the implementation of multi-keyword queries and searchable encryption schemes that satisfy forward security.

ACKNOWLEDGMENTS

This research was funded by the National Natural Science Foundation of China (Grant No.61662071 and 61662069). Thanks to supervisor Cao Suzhen and colleagues for their support and encouragement.

REFERENCES

[1] Chen, B., He, D., Kumar, N., et al., "A blockchain-based proxy re-encryption with equality test for vehicular communication systems," IEEE Transactions on Network Science and Engineering, 8(3), 2048-2059(2020).
[2] Chen, B., Wu, L., Kumar, N., et al., "Lightweight searchable public-key encryption with forward privacy over IIoT outsourced data," IEEE Transactions on Emerging Topics in Computing, 9(4), 1753-1764(2019).
[3] Khujamatov, K., Reypnazarov, E., Akhmedov, N., et al., "Blockchain for 5G Healthcare architecture," 2020 International Conference on Information Science and Communications Technologies (ICISCT), 1-5(2020).
[4] Allifah, N. M. and Zualkernan, I. A., "Ranking security of IoT-based smart home consumer devices," IEEE Access, 10, 18352-18369(2020).
[5] Liu, H., Chen, G. and Huang, Y., "Smart hardware hybrid secure searchable encryption in cloud with IoT privacy management for smart home system," Cluster Computing, 22(1), 1125-1135(2020).
[6] Xu, P., He, S., Wang, W., et al. "Lightweight searchable public-key encryption for cloud-assisted wireless sensor networks," IEEE Transactions on Industrial Informatics, 14(8), 3712-3723(2017).
[7] Du, R., Tan, A., et al., "Public key searchable encryption scheme based on blockchain," Journal on Communications, 41(4), 114-122(2020).
[8] Stefanov, E., Papamanthou, C. and Shi, E., "Practical dynamic searchable encryption with small leakage," Cryptology, 152-263(2013).
[9] Hoang, T., Yavuz, A. and Guajardo, J. "Practical and secure dynamic searchable encryption via oblivious access on distributed data structure," Proceedings of the 32nd Annual Conference on Computer Security Applications, 302-313(2016).
[10] Song, X., Dong, C., Yuan, D., et al., "Forward private searchable symmetric encryption with optimized I/O efficiency," IEEE Transactions on Dependable and Secure Computing, 17(5), 912-927(2017).
[11] Xu, P., Wu, Q., Wang, W., et al., "Generating searchable public-key ciphertexts with hidden structures for fast keyword search," IEEE Transactions on Information Forensics and Security, 10(9), 1993-2006(2020).
[12] Boneh, D. and Waters, B., "Constrained pseudorandom functions and their applications," International Conference on the Theory and Application of Cryptology and Information Security, 280-300(2013).
[13] Li, H., Wang, T., Qiao, Z., et al., "Blockchain-based searchable encryption with efficient result verification and fair payment," Journal of Information Security and Applications, 58, 102791(2021).
[14] Ma, M., He, D., Kumar, N., et al., "Certificateless searchable public key encryption scheme for industrial internet of things," IEEE Transactions on Industrial Informatics, 14(2), 759-767(2017).
[15] Yazdinejad, A., Srivastava, G., Parizi, R. M., et al., "Decentralized authentication of distributed patients in hospital networks using blockchain," IEEE Journal of Biomedical AND Health Informatics, 24(8), 2146-2156(2020).