Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Identity-based searchable attribute signcryption in lattice for a blockchain-based medical system*

Huifang YU†‡1,2, Xiaoping BAI1

1School of Cyberspace Security, Xi'an University of Posts & Telecommunications, Xi'an 710121, China

2School of Information Engineering, Qinghai Communications Technical College, Xining 810003, China

†E-mail: yuhuifang@xupt.edu.cn

Received Apr. 10, 2023; Revision accepted Aug. 17, 2023; Crosschecked Mar. 5, 2024

Abstract: Electronic healthcare systems can offer convenience but face the risk of data forgery and information leakage. To solve these issues, we propose an identity-based searchable attribute signcryption in lattice for a blockchain-based medical system (BCMS-LIDSASC). BCMS-LIDSASC achieves decentralization and anti-quantum security in the blockchain environment, and provides fine-grained access control and searchability. Furthermore, smart contracts are used to replace traditional trusted third parties, and the interplanetary file system (IPFS) is used for ciphertext storage to alleviate storage pressure on the blockchain. Compared to other schemes, BCMS-LIDSASC requires smaller key size and less storage, and has lower computation cost. It contributes to secure and efficient management of medical data and can protect patient privacy and ensure the integrity of electronic healthcare systems.

Key words: Blockchain; Identity-based searchable attribute signcryption; Distributed storage; NTRU lattice
https://doi.org/10.1631/FITEE.2300248 **CLC number:** TP309

1 Introduction

Traditional healthcare industry has undergone great improvement and now has stricter requirements for data transmission and storage (Price and Cohen, 2019). Electronic medical records (EMRs) are the focus of attention in modern healthcare, but historical records of patient information and treatment plans are vulnerable to attacks by hackers. In recent years, how to securely transmit and store medical data has become a research hotspot.

Compared with cloud storage technology (Ali et al., 2020), blockchain technology (Agbo et al., 2019) has the merits of decentralization and anti-interference. In addition, the blockchain can be integrated with Web 3.0 to return the ownership and control of data to the producers and users. To handle the low storage capacity of block nodes, Lu and Fu (2021) used the interplanetary file system (IPFS) to store the data, and the encrypted information is stored in IPFS, which returns the hash address. To overcome the storage bottleneck of the blockchain system, the hash address and access control information are stored on the blockchain. Kumar and Tripathi (2021) solved the problem of privacy protection in the Internet of Medical Things (IoMT) by deploying smart contracts to confirm the authentication of patients and medical devices. IPFS cluster nodes can ensure data confidentiality and device security.

Protection of the privacy of personal information is crucial in the medical field. As a cryptographic primitive, searchable encryption (Zhang AQ and Lin,

‡ Corresponding author

* Project supported by the Special Project of Kunlun Talent Teaching Master of Qinghai Province, China (No. [2020]18)

ORCID: Huifang YU, https://orcid.org/0000-0003-4711-3128; Xiaoping BAI, https://orcid.org/0000-0002-7056-8211

© Zhejiang University Press 2024

2018; Zhou et al., 2020; Chen et al., 2021; How and Heng, 2022) allows a server to find the corresponding file, but the server unknows the concrete content of the file. Therefore, privacy protection in the medical field can effectively prevent the leakage of medical data. Lattice-based searchable encryption for cloud computing (Zhang XJ and Xu, 2018) specifies only one cloud server and has low maintainability. Improved searchable encryption (Cheng et al., 2019) in the NTRU lattice reduces the calculation cost.

Cloud storage (Dohare et al., 2022) is one solution to handle a mass of medical data, but it faces the risk of illegal access. Attribute-based encryption can achieve fine-grained access control and ensure data confidentiality. Ciphertext-policy attribute based encryption can provide fine-grained access control. This scheme associates the private keys of the user with the attributes, and the ciphertext is associated with access policies. To prevent the leakage of user identity and data information, Chinnasamy et al. (2022) used the hash algorithm to hide the access policy and signature verification to provide inside security. The blockchain-based electronic health record (EHR) system (Li FQ et al., 2022) has low calculation overhead and is better than attribute based encryption with homomorphism. Multi-authority and revocable ciphertext-policy attribute based encryption from lattice (Yang et al., 2022) can resist quantum computing attacks and allow multiple authorities to participate in the required allocation. Attribute-based signcryption can better ensure the confidentiality and authenticity of data than attribute-based encryption.

Attribute-based searchable encryption (Miao et al., 2020; Wang HJ et al., 2020; Li XY, 2022) can ensure fine-grained access control and searchable ciphertext. Ciphertext-policy attribute based searchable encryption from lattice (Varri et al., 2021) realizes ciphertext-policy attribute based searchable encryption and has anti-quantum security. Attribute-based searchable encryption (Guo et al., 2022) realizes the sharing of different domains on the blockchain and uses off-chain storage to reduce the load on the blockchain. This scheme can resist quantum computing attacks and auxiliary attacks. Attribute-based keyword searchable encryption from the NTRU lattice (Li CY et al., 2022) uses attribute-based access policies and attributes for encryption to generate the private keys of users, but this scheme is subject to limitations in medical scenarios. The above schemes are vulnerable to man-in-the-middle attacks and key generation center leakage attacks (Wang WZ et al., 2022). A searchable signcryption based on multi-keyword attributes (Varri et al., 2023) cannot resist quantum computing attacks and has no unique identifiable identity. How to construct identity-based searchable attribute signcryption from the NTRU lattice to ensure anti-quantum security and low calculation cost in the blockchain medical field is a research hotspot.

In this study, we devise identity-based searchable attribute signcryption in lattice for the blockchain-based medical system (BCMS-LIDSASC). Main contributions are as follows:

1. In BCMS-LIDSASC, the blockchain and smart contract are used to achieve decentralization with no intermediary. Our scheme uses the IPFS technology for distributed storage, and thus it is highly suitable for medical application scenarios. Users have complete control over their data; thus, there is no data leakage or loss, and no incorrect storage from data custodians. Each user can actively take part in building their own EMR system.

2. BCMS-LIDSASC uses attributed-based searchable signcryption to ensure confidentiality, integrity, and authenticity relevant to the data. It can ensure searchability and fine-grained access control.

3. BCMS-LIDSASC has anti-quantum security and can effectively prevent the leakage of patient information.

2 Preliminaries

2.1 Lattice

Assume that the integer $N = 2^t>8$ is a security parameter, $\mathbb{R}$ the real space, and $\mathbb{Z}$ the integer space. BCMS-LIDSASC works over the rings $R=Z[x]/(x^N+1)$ and $R_q=Z_q[x]/(x^N+1)$, where x^N+1 can be split into k_q irreducible factors modulo q (prime $q>5$). For $x\in\mathbb{R}^N$, $\|x\|$ is the Euclidean norm of vector x. Let $f=\sum_{i=0}^{N-1} f_i x^i$ and $g=\sum_{i=0}^{N-1} g_i x^i$ be the polynomials over $\mathbb{R}$.

Definition 1 Choose an integer N (a power of 2) and two polynomials $f, g\in\mathbb{R}$, $h=gf^{-1} \bmod q$. The NTRU lattice determined by the polynomials h, q is $\Lambda_{h,q}=\{(u,v)\in\mathbb{R}^2\text{: } u + vh = 0 \bmod q\}$, $\Lambda_{h,q} = \begin{bmatrix} -C_N(h) & I_N \\ qI_N & 0_N \end{bmatrix}$ is a $2N\times 2N$ matrix, I_N is an identity matrix, 0_N is a zero matrix, $-C_N(h)$ is a Toeplitz matrix, and $\Lambda_{h,q}\in\mathbb{Z}^{2N}$ is a full-rank lattice.

2.2 TrapGen generation

Definition 2 The trapdoor generation algorithm over the NTRU lattice (TrapGen(N, q, σ)) is as given in Algorithm 1.

Algorithm 1 TrapGen(N, q, σ)

```
Input: (N, q, σ), where σ is an error scalar factor
Output: (B, h)
1. Choose $f, g \leftarrow D_{N,\sigma_f}$, where $\sigma_f \leftarrow 1.17\sqrt{\frac{q}{2N}}$
2. Compute Gram-Schmidt norm $\|\overline{B}_{f,g}\|$, where Norm $\leftarrow$
   $\max\left(\|g-f\|, \left\|\left(\frac{g\bar{f}}{f\bar{f}+g\bar{g}}, \frac{g\bar{g}}{f\bar{f}+g\bar{g}}\right)\right\|\right)$
3. If Norm$\le 1.17\sqrt{q}$, continue; otherwise, goto line 1
4. Use the extended Euclidean algorithm to compute $\rho_f, \rho_g\in\mathbb{R}$,
   $R_f, R_g\in\mathbb{Z}$ such that $\rho_f f=R_f \bmod (x_N + 1)$ and $\rho_g g = R_g \bmod (x_N + 1)$
5. If $\gcd(R_f, R_g) \ne 1$ or $\gcd(R_f, g) \ne 1$, goto line 1
6. Choose $\xi, \psi\in\mathbb{Z}$ satisfying $\xi R_f + \psi R_g=1$
7. Compute $F = q\xi\rho_g$, $G = q\psi\rho_f$, $\alpha = \left[\frac{F\bar{g} + G\bar{g}}{f\bar{f} + g\bar{g}}\right]\in\mathbb{R}$
8. Reduce $F, G$ as $F \leftarrow F - \alpha f$, $G \leftarrow G - \alpha g$
9. Return $h = gf^{-1} \bmod q$, $B = \begin{pmatrix} A(g) - A(f) \\ A(G) - A(F) \end{pmatrix}$, where $h\in R_q$,
   $B\in Z_q^{2N\times 2N}$
```

2.3 Discrete Gaussian and sampling algorithms

Definition 3 For any $\sigma > 0$, $c\in\mathbb{R}^N$, the n-dimensional discrete Gaussian function $\rho_{\sigma,c}\text{: }\mathbb{R}^N \to (0, 1]$ is defined as follows: $\rho_{\sigma,c}(x) \triangleq \exp\left(-\frac{\|x-c\|^2}{2\sigma^2}\right)$, for any $\Lambda\subset\mathbb{R}^N$, $\rho_{\sigma,c}(\Lambda) \triangleq \sum_{x\in\Lambda}\rho_{\sigma,c}(x)$.

Definition 4 The Gaussian sampling algorithm in the NTRU lattice (GSample(B, σ, c)) is as given in Algorithm 2.

Algorithm 2 GSample(B, σ, c)

```
Input: a basis $B$ of $N$-dimensional lattice $\Lambda$, standard deviation $\sigma > 0$, and center $c \in \mathbb{Z}^N$
Output: $v$ sampled in $\mathcal{D}_{\Lambda,\sigma,c}$
1. $v_N \leftarrow 0$, $c_N \leftarrow c$
2. for $i \leftarrow N, \cdots, 2, 1$ do
3.     $c_i' \leftarrow \langle c_i, b_i\rangle / \|\tilde{b}\|^2$, $\sigma_i' \leftarrow \sigma / \|\tilde{b}\|^2$
       // $\tilde{B} = (\tilde{b})_{i\in\mathbb{N}}$ is Gram-Schmidt orthogonalization of $B$
4.     $z_i' \leftarrow$ Sample_Z$(\sigma_i', c_i')$
       // Sampling a one-dimensional vector from the Gaussian
       // distribution $\mathcal{D}_{z,\sigma',c'} c_{i-1} \leftarrow c_i - z_i b_i$ and $v_{i-1} \leftarrow v_i + z_i b_i$
5. end for
6. return $v_0$
```

2.4 Hard assumptions

Definition 5 ψ_ξ is a Gaussian distribution over $\mathbb{R}$ with mean 0 and standard deviation $\xi/\sqrt{2\pi}$. ψ_ξ^N is the spherical Gaussian distribution of vector $(v_1, v_2, \cdots, v_N)$ over $\mathbb{R}^N$, where each coordinate is independently distributed in ψ_ξ. $\omega\in R_q$ is a ring with a uniform distribution, and the distribution of ω obeys ψ_ξ^N. $(\eta_i, y_i = \eta_i\omega + e_i)\in R_q \times R_q$ is a noise equation, where η_i is a random term and e_i is an error term. Distribution $A_{\xi,\psi}$ of learning with errors over ring (RLWE) is as follows: choose $\eta_i\in R_q$ and error vector e_i based on the distribution ψ_ξ^N, and then output $(\eta_i, y_i=\eta_i\omega+e_i)\in R_q\times R_q$.

Definition 6 For any $\omega\in R_q$, the decision-RLWE problem is to distinguish the RLWE distribution $A_{\xi,\psi}$ from the uniform distribution $R_q\times R_q$ with a non-negligible probability.

Definition 7 (Li RN et al., 2022) Given a prime q, a positive real number ϱ, a positive integer ς, and polynomials f, g, h ($f, g\in R_q$, $h=gf^{-1} \bmod q$), the small integer solution problem in ring (RSIS) in the NTRU lattice is to find $(z_1, z_2)\in\Lambda_{h,q}$ such that $\|(z_1, z_2)\|\le\varrho$.

2.5 Linear secret sharing scheme

Access structures are used to share the secrets between different users. U acts as the set of universal attributes. A secret sharing scheme π is linear if it satisfies the following conditions:

1. Choose $r\in Z_q$ as a shared secret value.

2. Each element in the shared vector is corresponding to an attribute in U, and all elements in the shared vector are in field Z_q.

3. For each access structure, there exists a matrix M with l rows and n columns, called the shared generation matrix of π. For all i=1, 2, $\cdots$, l, the ith row of M is M_i, the mapping function ρ defines the ith row of participant as $\rho(i)$, and the function ρ maps the ith

row to attribute Attr in U, written as $\rho(i)$→Attr, Attr∈U, i=1, 2, ⋯, l. Column vector v=(r, s_2, ⋯, s_n), where shared secret value $r \in Z_q$ and $s_2, s_3, \cdots, s_n \in Z_q$. Then, Mv is the vector of l secret factors, with each secret factor $\lambda_i = M_i v$ corresponding to attribute $\rho(i)$. S is an authorization attribute set, I={ i: $\rho(i) \in S$} (i=1, 2, ⋯, l). A constant set $\{\theta_i \in Z_q\}_{i \in I}$ satisfies $\sum_{i \in I} \theta_i M_i = (1, 0, \cdots, 0)$. If λ_i is the effective secret factor of the access structure, the secret value r can be calculated by $r = \sum_{i \in I} \theta_i \lambda_i$.

3 System model

BCMS-LIDSASC includes four main entities: data owner, data user, blockchain, and IPFS distribution storage system. Fig. 1 shows the system structure model of BCMS-LIDSASC.

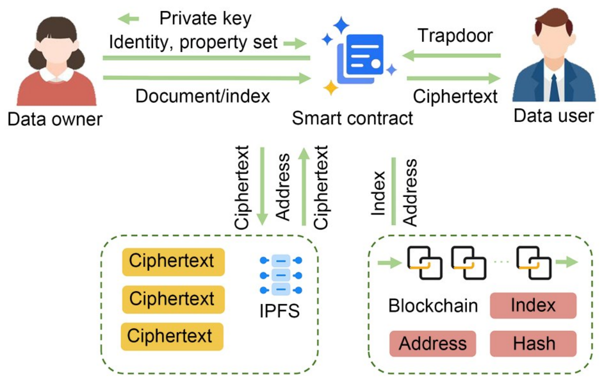

Fig. 1 System structure model of BCMS-LIDSASC

Data owner (DO): DO can be either a patient who owns the electronic medical records or a doctor who issues the prescriptions for patients. DO sends their identity and attribute set to a smart contract on the blockchain to obtain their public and private keys. Then, they generate an index using the keywords, and upload the index and ciphertext to the blockchain.

Data user (DU): DU can be either a doctor who needs to access the medical records of the patient or a patient who needs to access the prescriptions. DU uses the keywords to generate a trapdoor and uploads it to the smart contract. If the smart contract successfully matches the trapdoor, it uses the storage address in the blockchain to download the ciphertext from IPFS and returns the ciphertext to the user.

Blockchain: The blockchain stores the storage address, ciphertext hash, and index value generated by IPFS. The smart contract generates the user key in the initialization phase, and matches the indexes and trapdoors in the search phase. If the matching is successful, the ciphertext is downloaded to the user based on the address in the blockchain.

IPFS: IPFS is a peer-to-peer distribution file storage system. After a file is successfully stored, a hash value will be generated. The hash value generated by IPFS is stored in the blockchain, and thus the storage pressure on the blockchain is greatly reduced.

4 BCMS-LIDSASC

4.1 Setup

The administrator carries out the following operations to initialize the smart contract:

1. The administrator selects two system parameters (N, q), a small positive integer p, and a global attribute set U={$u_1, u_2, \cdots, u_n$}.

2. The administrator runs TrapGen(N, q, σ) to obtain the system master public key h and master private key B.

3. Hash functions are follows: H_1, H_4: $\{0, 1\}^* \to Z_q^N$, H_2: $\{0, 1\}^N \to Z_q^N$, H_3: $\{0, 1\}^N \times \{0, 1\}^N \to Z_q^N$, and H_5: $\{0, 1\}^N \times Z_q^N \to Z_q^N$.

4. Master private key Msk = B and public system parameters ψ={$h, p, H_1, H_2, H_3, H_4, H_5$}.

4.2 KeyGen

Assume that ID$_i$ is the user identity. The user sends its identity ID$_i$ and attribute set $U_i \in U$ to a smart contract. The smart contract carries out the calculations as follows:

1. Calculate $Q_i = H_1(\mathrm{ID}_i)$ as the public key of ID$_i$.

2. Obtain $(e_i, x_i) \leftarrow \mathrm{GSample}(B, \sigma, (Q_i, 0))$ and $e_i + x_i h = Q_i$.

3. Calculate $a_i = H_2\left(\sum_{u_j \in U_i} u_j\right)$ based on the user attribute set U_i.

4. Obtain $(e_i', d_i) \leftarrow \mathrm{GSample}(B, \sigma, (a_i, 0))$ such that $e_i' + hd_i = a_i$, where $(e_i', d_i) \in \Lambda_h^q$.

5. The private key of identity ID$_i$ is SK$_i$={e_i, x_i, d_i}. The public key of the data owner is Q_A, and the corresponding private key is SK$_A$={e_A, x_A, d_A}. The

public key of the data receiver is Q_B, and the corresponding private key is $\mathrm{SK}_B=\{e_B, x_B, d_B\}$.

4.3 Signcrypt

Input the system public key h, private key SK_A of DU, attribute set $U_A \in U$, and message $m \in \{0,1\}^N$. DO uses the access structure (M, ρ), where M is an $l \times n$ matrix, and ρ maps each row of this matrix to its attribute. DO chooses $r \in R_q$ and a random vector $s = (r, s_2, \cdots, s_n) \in R_q$. Then, for $j \in \{1, 2, \cdots, l\}$, DO calculates $\varphi_j = sM_j$ (M_j is the j^{th} row of matrix M). Signcrypter (DO) carries out the calculations as follows:

1. Choose $u, v_1, v_2 \leftarrow \{-1, 0, 1\}^N$.
2. Calculate $C_0 = Q_B ur + a_A + m + v_1$ and $C_{1,j} = uh\varphi_j + v_2 p$.
3. Calculate $\delta = H_3(v_1 + hv_2, m)$.
4. Calculate $z_1 = v_1 + e_A \delta$ and $z_2 = v_2 + x_A \delta$.
5. Return $\mathrm{CF} = \{C_0, \{C_{1,j}\}_{j \in \{1,2,\cdots,l\}}, \delta, z_1, z_2\}$ as the ciphertext.

4.4 IndexGen

DO chooses the keyword $k \in \{0, 1\}^*$ and carries out the calculations as follows:

1. Calculate $t = H_4(k)$.
2. Choose $r_1, e_1, e_2 \leftarrow \{-1, 0, 1\}$ and $w \leftarrow \{0, 1\}^N$.
3. Calculate $I_1 = r_1 h + e_1 \in R_q$ and $I_2 = r_1 t + e_2 + \left\lfloor \frac{q}{2} \right\rfloor w \in R_q$.
4. Calculate $I_3 = H_5(w, a_A)$.
5. Set the timestamp T_s.
6. Upload the index $I = \{I_1, I_2, I_3, T_s\}$ and CF to a smart contract.

4.5 Trapdoor

DU chooses the keyword $k' \in \{0, 1\}^*$ to search and carries out the operations as follows:

1. Calculate $t' = H_4(k')$.
2. Obtain $(e, T_1) \leftarrow \mathrm{GSample}(B, \sigma, (t', 0))$.
3. Set $T_2 = a_B$.
4. Upload the trapdoor $T = \{T_1, T_2\}$ to a smart contract.

4.6 Search

In this algorithm, the smart contract receives the trapdoor T and executes the search algorithm.

First, the smart contract verifies the validity of timestamp T_s. If $|T_s - T_{s-\mathrm{cur}}| \le \triangle T_s$ ($T_{s-\mathrm{cur}}$ is the current timestamp and $\triangle T_s$ is the maximum valid time interval), the smart contract performs the following search operations; otherwise, the search fails.

1. Calculate $w' = \left\lceil \frac{I_2 - I_1 T_1}{q/2} \right\rfloor$.
2. The smart contract verifies whether $H_5(w', T_2) = I_3$. Search the ciphertext by the storage address in the blockchain and return it to DU if the ciphertext exists; otherwise, return $\perp$.

4.7 Unsigncrypt

Input the private key SK_B of the receiver, public key Q_A of DU, and $\mathrm{CF}=\{C_0, \{C_{1,j}\}_{j \in \{1,2,\cdots,l\}}, \delta, z_1, z_2\}$. If the user attribute U_B satisfies (M, ρ) and φ_A is based on the secret r of matrix M, there must exist a set $\{\theta_A \in Z_q\}_{A \in l}$ of constants according to the properties of the linear secret sharing matrix such that $\sum_{A \in l} \varphi_A \theta_A = r$. The receiver calculates $m' = C_0 - x_B \sum_{j \in l} C_{1,j} \theta_j - d_B h$ and $m = m' \bmod p$. Then, the receiver verifies whether $\delta = H_3(hz_2 + z_1 - Q_A \delta, m)$. The receiver accepts m if yes and rejects it otherwise.

5 Correctness analysis

5.1 Correctness of the search phase

Assuming that the keywords match, then $t = t'$. Assuming that the attributes match, then $a_A = T_2 = a_B$.

$$\begin{aligned} & I_2 - I_1 T_1 \\ & = r_1 t + e_2 + \left\lfloor \frac{q}{2} \right\rfloor w - (r_1 h + e_1) T_1 \\ & = r_1 t + e_2 + \left\lfloor \frac{q}{2} \right\rfloor w - r_1 (t' - e) - e_1 T_1 \\ & = e_2 + \left\lfloor \frac{q}{2} \right\rfloor w + r_1 e - e_1 T_1, \end{aligned}$$

where r_1, e, e_1, e_2, T_1 are short vectors which can be disregarded and omitted in computations. Based on the properties of the NTRU lattice, the coefficients of $e_2 + r_1 e - e_1 T_1$ will be in $(-q/4, q/4)$. Then, we can obtain $w' = \left\lceil \frac{I_2 - I_1 T_1}{q/2} \right\rfloor = w$, so $H_3(w', T_2) = I_3$ holds, and the smart contract successfully completes the matching process.

5.2 Correctness of the unsigncryption phase

If user attributes meet the requirement specified by the signcryption, we have

$$\begin{aligned}
&\textstyle\sum_{j\in I}\varphi_j\theta_j = r,\ a_A = a_B,\\
&m' = C_0 - x_B\sum_{j\in I}C_{1,j}\theta_j - d_B h\\
&\quad = C_0 - x_B\sum_{j\in I}\left(uh\varphi_j + v_2 p\right)\theta_j - d_B h\\
&\quad = Q_B ur + a_A + m + v_1 - x_B uhr - x_B v_2 p\sum_{j\in I}\theta_j\\
&\qquad - \left(e'_B + hd'_B\right)\\
&\quad = \left(e_B + x_B h\right)ur + m + v_1 - x_B uhr - x_B v_2 p\sum_{j\in I}\theta_j - e'_B\\
&\quad = e_B ur + m + v_1 - x_B p v_2\sum_{j\in I}\theta_i + e'_B,
\end{aligned}$$

where u, v_1, v_2, e_B, and e'_B are all short vectors. Therefore, $m = m' \bmod p$.

Next, we verify the authenticity of message m:

$$\begin{aligned}
&hz_2 + z_1 - Q_A\delta\\
&= h(v_2 + x_A\delta) + v_1 + e_A\delta - (e_A + x_A h)\delta\\
&= v_1 + hv_2.
\end{aligned}$$

Hence, $\delta = H_3\left(hz_2 + z_1 - Q_A\delta, m\right)$ holds, showing that message m is valid.

6 Security analysis

Theorem 1 If an adversary $\mathcal{A}$ cannot break the keyword trapdoor indistinguishability of the underlying identity-based searchable attribute signcryption from the lattice in a random oracle model, BCMS-LIDSASC also has keyword trapdoor indistinguishability.

Proof Assume that the challenger Γ receives a random instance of the RLWE problem. It wants to determine whether $(\eta_i, y_i = \eta_i\omega + e_i) \in R_q \times R_q$ is sampled from $A_{\xi,\psi}$ or the uniform distribution over $R_q \times R_q$. $\mathcal{A}$ is a subroutine of Γ in the whole game. Initially empty lists $(L_1, L_2, L_3, L_4, L_5, L_k)$ track the different oracles. ID_γ is a challenge identity and δ is the probability that $\mathrm{ID}_i = \mathrm{ID}_\gamma$, where $(\gamma, \mathrm{ID}_\gamma)$ $(\gamma \in \{1, 2, \cdots, q_1\}$ and q_1 is the time of querying the H_1 oracle) are unknown to $\mathcal{A}$.

Γ carries out the setup algorithm and returns $\psi = \{h, p, H_1, H_2, H_3, H_4, H_5\}$ to $\mathcal{A}$, but the system master key Msk is kept secret. In the first phase, $\mathcal{A}$ issues adaptive queries as follows:

H_2 queries: $\mathcal{A}$ queries the hash value a_i of the attribute set. Γ selects the attribute set $U_i \in U$ to calculate $a_i \leftarrow H_2\left(\sum_{u_j\in U_i} u_j\right)$, and then returns it to $\mathcal{A}$ and stores (U_i, a_i) in L_2.

H_1 queries: $\mathcal{A}$ queries the hash value of identity ID_i. If $\mathrm{ID}_i = \mathrm{ID}_\gamma$, Γ fails and aborts; otherwise, Γ selects $Q_i \in Z_q^N$ and calls the H_2 oracle to obtain a_i, and sends (Q_i, a_i) to $\mathcal{A}$ and stores $(\mathrm{ID}_i, Q_i, a_i)$ in L_1.

H_3 queries: $\mathcal{A}$ queries the hash value for (v_1, v_2, m). Γ checks whether L_3 has (v_1, v_2, m, δ). If yes, Γ sends δ to $\mathcal{A}$; otherwise, Γ returns a random $\delta \in Z_q^N$ to $\mathcal{A}$ and records (v_1, v_2, m, δ) in L_3.

H_4 queries: $\mathcal{A}$ queries the hash value t_i of keyword k_i. Γ sends t_i to $\mathcal{A}$ if k_i exists in L_4; otherwise, Γ returns a random $t_i \leftarrow H_4(k_i) \in Z_q^N$ to $\mathcal{A}$ and stores (k_i, t_i) in L_4.

H_5 queries: $\mathcal{A}$ queries the hash value of $I_{3,i}$. Γ returns $I_{3,i}$ to $\mathcal{A}$ if there is a relevant tuple; otherwise, Γ selects $w_i \leftarrow \{0, 1\}^N$, calls the H_2 oracle to obtain a_i, calculates $I_{3,i} \leftarrow H_5(w_i, a_i)$, returns $I_{3,i}$ to $\mathcal{A}$, and stores $(w_i, I_{3,i})$ in L_5.

Index queries: Γ receives an index query for $(k, \mathrm{ID}_A, \mathrm{ID}_B)$. Γ runs the index generation algorithm to return I to $\mathcal{A}$ if $\mathrm{ID}_B \neq \mathrm{ID}_\gamma$; otherwise, Γ calls the H_4, H_5 oracles to obtain t_i, I_3, chooses $r_1, e_1, e_2 \leftarrow \{-1, 0, 1\}$ to calculate $I_1 = r_1 h + e_1$, $I_2 = r_1 t_i + e_2 + \lfloor q/2 \rfloor w_i$, and returns $I = \{I_1, I_2, I_3\}$ to $\mathcal{A}$.

Trapdoor queries: Γ receives a trapdoor query for $(T, \mathrm{ID}_A, \mathrm{ID}_B)$. Γ runs the trapdoor generation algorithm to return T to $\mathcal{A}$ if $\mathrm{ID}_B \neq \mathrm{ID}_\gamma$; otherwise, Γ obtains $(e, T_1) \leftarrow \mathrm{GSample}(B, \sigma, (t_i, 0))$, $T_2 = a_i$, and returns $T = \{T_1, T_2\}$, where a_i, t_i are from the H_2, H_4 oracle queries.

$\mathcal{A}$ submits a challenge query after the above queries are over. $\mathcal{A}$ selects keywords (k_0, k_1) with the same length about identities $(\mathrm{ID}_A^*, \mathrm{ID}_B^*)$. Γ fails and stops if $\mathrm{ID}_B^* \neq \mathrm{ID}_\gamma$; otherwise, Γ selects the random $\mu \in \{0, 1\}$. If $\mu = 1$, Γ returns the ciphertext to $\mathcal{A}$. If $\mu = 0$, Γ runs the index and trapdoor algorithms to obtain a trapdoor T_μ for keyword k_μ and returns to $\mathcal{A}$.

$\mathcal{A}$ finally outputs a guess μ'. $\mathcal{A}$ wins in the whole game if $\mu' = \mu$. Assume that the success advantage of $\mathcal{A}$ is $\varepsilon = \Pr\left(\left|[\mu' = \mu] - 1/2\right|\right)$. Then the probability of Γ in solving the RLWE problem is $\varepsilon' \geq \varepsilon/(\mathrm{e}q_1)$, where e is a constant and q_1 is the query time to the H_1 oracle (Li CY et al., 2022).

Theorem 2 If an adversary $\mathcal{A}$ cannot break the ciphertext indistinguishability of the underlying identity-based searchable attribute signcryption from the lattice in a random oracle model, BCMS-LIDSASC also has ciphertext indistinguishability.

Proof Assume that Γ receives a random instance of the RLWE problem. Its aim is to determine whether $(\eta_i, y_i = \eta_i\omega+e_i) \in R_q\times R_q$ is sampled from $A_{\xi,\psi}$ or a uniform distribution over $R_q\times R_q$. Γ is a challenger of $\mathcal{A}$ in the whole game.

Γ carries out the setup algorithm to return $\psi=\{h, p, H_1, H_2, H_3, H_4, H_5\}$ to $\mathcal{A}$, but the system master key Msk is kept secret by Γ. $\mathcal{A}$ then submits adaptive queries to Γ in the first phase. Hash queries are the same as in the first phase in Theorem 1.

Private key queries: $\mathcal{A}$ queries the private key of ID_i. Γ fails and stops if $\mathrm{ID}_i = \mathrm{ID}_\gamma$; otherwise, Γ obtains $(e_i, x_i) \leftarrow \mathrm{GSample}\big(B, \sigma, (Q_i, 0)\big), (e_i', d_i) \leftarrow \mathrm{GSample}\big(B, \sigma, (a_i, 0)\big)$, where Q_i, a_i are from the hash oracle queries, and Γ returns the private key (e_i, x_i, d_i) to $\mathcal{A}$ and records $(\mathrm{ID}_i, Q_i, e_i, x_i, d_i)$ in L_k.

Signcryption queries: $\mathcal{A}$ issues a signcryption query for $(m, \mathrm{ID}_A, \mathrm{ID}_B)$. Γ runs the actual signcryption algorithm to return a ciphertext CF if $\mathrm{ID}_A \neq \mathrm{ID}_\gamma$. Otherwise, Γ answers as follows:

1. Choose the corresponding access policy (M, ρ), where M is an $l \times n$ matrix, and ρ maps each row of the matrix to its attribute.
2. Choose vector $s = (r, s_2, \cdots, s_n) \in Z_q^*$ and calculate $\varphi_i = sM_i$ for each $i \in\{1, 2, \cdots, l\}$.
3. Choose $u, v_1, v_2 \leftarrow \{0,1\}^N$.
4. Calculate $C_0 = Q_B ur + a_B + m + v_1$.
5. Calculate $C_{1,j} = u\varphi_j + v_2 p$, $\delta = H_3(hv_2, m)$.
6. Calculate $z_1 = Q_A\delta$, $z_2 = v_2$.
7. Output $\mathrm{CF} = \{C_0, \{C_{1,j}\}_{j\in\{1,2,\cdots,l\}}, \delta, z_1, z_2\}$ as the ciphertext.

CF is verified by the adversary $\mathcal{A}$ as follows:

$$hz_2 + z_1 - Q_A\delta = hz_2 + Q_A\delta - Q_A\delta = hv_2.$$

Unsigncryption queries: $\mathcal{A}$ issues an unsigncryption query for $(\mathrm{CF}, \mathrm{ID}_A, \mathrm{ID}_B)$. Γ runs the actual unsigncryption algorithm to return a result if $\mathrm{ID}_B\neq\mathrm{ID}_\gamma$. Otherwise, Γ seeks different attribute sets U_i from L_2 and calculates

$$m = C_0 - Q_B\sum_{j\in l} C_{1,j}\theta_j - a_B\big(\mathrm{mod}\, p\big).$$

Γ outputs m to $\mathcal{A}$ if $\delta = H_3\big(hz_2 + z_1 - Q_A\delta, m\big)$, and $\perp$ otherwise.

$\mathcal{A}$ submits a challenge query after the above queries are over. $\mathcal{A}$ selects two messages (m_0, m_1) with the same length and two identities $(\mathrm{ID}_A^*, \mathrm{ID}_B^*)$, where ID_A^* is the identity of the data owner and ID_B^* is the identity of the data user. Γ fails and aborts if $\mathrm{ID}_B^* \neq \mathrm{ID}_\gamma$; otherwise, Γ selects a random $\mu\in\{0, 1\}$, sets $Q_B^*\in Z_q^N$, obtains the attribute set $a_i^*\neq a_B$ from the H_2 oracle, and continues to respond as follows:

1. Choose the access policy (M^*, ρ^*), where M^* is an $l\times n$ matrix and ρ^* maps each row of the matrix to its attribute.
2. Choose vector $s=(r, s_2, \cdots, s_n) \in Z_q^*$ and calculate $\varphi_i^* = sM^*$ for each $i\in\{1, 2, \cdots, l\}$.
3. Calculate $C_0^*=Q_B ur+a_A+m_\mu+v_1$ and $C_{1,j}^*=uh\varphi_j^*+v_2 p$.
4. Calculate $\delta^*=H_3(v_1+hv_2, m_\mu)$, $z_1^*=v_1+e_A\delta^*$, and $z_2^*=v_2+x_A\delta^*$.
5. Output $\mathrm{CF}^*=\{C_0^*, \{C_{1,j}^*\}_{j\in\{1,2,\cdots,l\}}, \delta^*, z_1^*, z_2^*\}$.

In the second phase, $\mathcal{A}$ makes another series of adaptive queries as in the first phase. $\mathcal{A}$ cannot query the private key oracle about ID_B^* or unsigncryption oracle about CF^*.

$\mathcal{A}$ finally outputs a guess μ'. $\mathcal{A}$ wins in the whole game if $\mu'=\mu$. Γ can distinguish whether $(\eta_i, y_i=\eta_i\omega+e_i)$ comes from $A_{\xi,\psi}$ or a uniform distribution $R_q\times R_q$. Assume that the success advantage of $\mathcal{A}$ is $\varepsilon=\Pr(|[\mu'=\mu]-1/2|)$. Then the probability that Γ obtains the decision-RLWE problem solution is $\varepsilon'\leq\varepsilon/(\mathrm{e}q_1 q_k)$ (Yu et al., 2022), where q_1 is the query time to the H_1 oracle and q_k is the query time to the private key oracle.

Theorem 3 If the underlying identity-based searchable attribute signcryption from the lattice has unforgeability, BCMS-LIDSASC also has unforgeability.

Proof Assume that Γ receives a random instance of the RSIS problem and tries to find $(z_1, z_2) \in \Lambda_{h,q}$ such that $\|(z_1, z_2)\|\leq\beta$. $\mathcal{F}$ is the subroutine of challenger Γ in the whole game. ID_γ is the challenge identity and $(\gamma, \mathrm{ID}_\gamma)$ are unknown to $\mathcal{F}$. Γ first returns ψ from the setup algorithm but Msk is unknown to $\mathcal{F}$. $\mathcal{F}$ issues adaptive queries as in the first phase in Theorem 2.

$\mathcal{F}$ finally outputs a forgery $(\mathrm{ID}_A^*, \mathrm{ID}_B^*, \mathrm{CF}^*)$ to Γ. $\mathcal{F}$ cannot query the private key of ID_A^*. Γ fails and aborts if $\mathrm{ID}_A^* \neq \mathrm{ID}_\gamma$; otherwise, $\mathcal{F}$ forges another ciphertext

$\mathrm{CF}' = \{ C_0', \{ C_{1,j}' \}_{j \in \{1,2,\cdots,l\}}, \delta', z_1', z_2' \}$. By using a forking lemma (Yu et al., 2022), we can obtain

$$\begin{aligned} & z_1^* - z_1' + h(z_2^* - z_2') + Q_A(\delta' - \delta^*) = 0 \\ & \Rightarrow z_1^* - z_1' + h(z_2^* - z_2') + (e_A^* + x_A^* h)(\delta' - \delta^*) = 0 \\ & \Rightarrow z_1^* - z_1' + e_A^*(\delta' - \delta^*) + h((z_2^* - z_2') \\ & \quad + x_A^*(\delta' - \delta^*)) = 0, \end{aligned}$$

where $\| z_1' \|, \| z_1^* \|, \| z_2' \|, \| z_2^* \| \le 2\sigma\sqrt{N}$, $\| e_A^* \|, \| x_A^* \| \le \sigma\sqrt{N}$, and $\| \delta' - \delta^* \| \le \| \delta' \| + \| \delta^* \| \le 2N$.

Therefore, the following inequalities hold:

$$\begin{aligned} & \left\| z_1^* - z_1' + e_A^*(\delta' - \delta^*) \right\| \\ & \le \| z_1^* \| + \| z_1' \| + \| e_A^* \| \left(\| \delta' \| + \| \delta^* \| \right) \\ & \le 2\sigma^2 N + 4\sigma\sqrt{N}, \\ & \left\| z_2^* - z_2' + x_A^*(\delta' - \delta^*) \right\| \\ & \le \| z_2^* \| + \| z_2' \| + \| x_A^* \| \left(\| \delta' \| + \| \delta^* \| \right) \\ & \le 2\sigma^2 N + 4\sigma\sqrt{N}. \end{aligned}$$

Γ cannot determine which set of private keys $\mathcal{F}$ chooses to forge the ciphertext, so Γ randomly selects a set of private keys such that $z_1^* - z_1' + h(z_2^* - z_2') + Q_A(\delta' - \delta^*)(\neq 0)$ is at least 1/2. Assume that a forger $\mathcal{F}$ can break the unforgeability with advantage ε. Then the probability that Γ solves the RSIS problem is $\varepsilon' \ge \varepsilon(1 - 2^{-\omega \log_2 N})$ (Yu et al., 2022).

7 Potential threats analysis

BCMS-LIDSASC may face some potential threats during the practical implementation.

1. Smart contract attacks: In our scheme, the smart contracts are used to generate the user keys in the initialization phase and match the indexes and trapdoors in the search phase. Hence, the security of smart contracts is important. In our scheme, smart contracts replace the traditional KGC in the ethereum blockchain. To launch an attack on smart contracts, an attacker must control more than 51% of the accounts in the blockchain. Current proof-of-stake (PoS) consensus mechanism is used by ethereum blockchain (2.0). In this scenario, the attackers would have to spend a substantial amount of funds to obtain system control. The cost of conducting 51% of the attacks on the ethereum blockchain is prohibitively high; i.e., the cost far exceeds the potential benefits that an attacker could obtain. Thus, it is highly unlikely that an attacker would successfully carry out an attack or disrupt the whole system. Hence, our scheme can resist attacks on smart contracts.

2. Privilege-insider attack: Once a smart contract is deployed, even the author of the contract cannot modify the deployed contract code but to destroy it. Hence, even if the system master key of BCMS-LIDSASC is known, it is impossible to modify the logic of the deployed smart contract; thus, the privilege-insider attacks cannot occur.

3. Replay attack: After the smart contract receives a search request, it first checks its validity by verifying whether $|T_s - T_{s-\mathrm{cur}}| \le \triangle T_s$. If this inequality does not hold, the smart contract rejects the search, and thus the attacker cannot intercept or replay with the new timestamp. Hence, BCMS-LIDSASC can resist the replay attacks.

8 Performance analysis

Table 1 gives a feature comparison of our scheme, VPK1 (Varri et al., 2021), GHWL (Guo et al., 2022), LDL (Li CY et al., 2022), and VPK2 (Varri et al.,

Table 1 Comparison of security features

Feature	VPK2	VPK1	GHWL	LDL	BCMS-LIDSASC
Fine-grained access control	√	√	√	√	√
Index verification	√	√	√	√	√
Security assumption	DBDH	LWE	RLWE	RLWE	RLWE/RSIS
Confidentiality	√	√	√	√	√
Unforgeability	√	×	×	×	√
No key generation center	×	×	×	×	√
Applied to blockchain	×	×	×	√	√

2023). The experiment platform is Win10 OS with 2.60 GHz Intel® Core™ i7-10750H and 32.0 GB memory. The security of our scheme is related mainly to values of q and N. We set N=128, q=2^{18} and N=192, q=2^{25}. Table 2 lists the communication cost including the IndexGen size, trapdoor size, ciphertext size, and signature size. VPK1 relies on the LWE problem, and parameter M should satisfy $M \geq 2N\lceil \log_2 q \rceil$. Note that $k = n$ is the number of keywords. GHWL relies on the RLWE problem, assuming $q_1 = q$. LDL relies on the RLWE problem in the NTRU lattice, but its encryption uses only one user attribute set for calculation, so it has low security. Table 3 lists the concrete instance comparison among several schemes.

Our scheme increases the communication cost of signature compared to other schemes but ensures data authenticity, so this increase is justifiable. Our scheme uses the NTRU lattice in trapdoor generation to obtain better security and lower communication cost; it can achieve the same security effect with fewer bits in real applications. Our scheme has higher ciphertext communication cost than LDL due to its more detailed attribute partitioning scheme, but it can better protect the privacy of users and data in practical scenarios, so this increase is worthwhile.

Table 4 lists the average time of each operation based on the PBC library. Table 5 lists the total calculation overhead of several schemes.

Matlab software deals with the simulations, where n is the number of chosen attributes. From Fig. 2, with an increasing number of attributes, the time consumption of several schemes increases linearly, but the computation time of BCMS-LIDSASC is the least.

Table 2 Communication cost comparison

Scheme	IndexGen size (bit)	Trapdoor size (bit)	Ciphertext size (bit)	Signature size (bit)
VPK1	$(2M+n)N\log_2 q$	$MN\log_2 q$	$(kM+1)N\log_2 q$	
GHWL	$(n+\log_2 q_1)N\log_2 q_2$	$(n+1)N\log_2 q_2$	$(n+1)N\log_2 q_1$	
LDL	$(2N+1)\log_2 q$	$(2N^2+1)\log_2 q$	$(2N+1)\log_2 q$	
BCMS-LIDSASC	$(2N+1)\log_2 q$	$2N^2\log_2 q$	$(n+1)N\log_2 q$	$3\log_2 q$

Table 3 Concrete instance comparison of several schemes

Parameter	Value			
	Instance 1	Instance 2	Instance 3	Instance 4
N	128	128	192	192
$q(q_1)$	2^{18}	2^{18}	2^{25}	2^{25}
q_2	2^{25}	2^{25}	2^{26}	2^{26}
M	4808	4808	9800	9800
n	5	10	10	20
IndexGen size in VPK1 (bit)	22 166 784	22 178 304	94 128 000	94 176 000
IndexGen size in GHWL (bit)	73 600	89 600	174 720	224 640
IndexGen size in LDL (bit)	4626	4626	9625	9625
IndexGen size in BCMS-LIDSASC (bit)	4626	4626	9625	9625
Trapdoor size in VPK1 (bit)	11 077 632	11 077 632	47 040 000	47 040 000
Trapdoor size in GHWL (bit)	19 200	35 200	54 912	104 832
Trapdoor size in LDL (bit)	589 842	589 842	1 843 225	1 843 225
Trapdoor size in BCMS-LIDSASC (bit)	589 824	589 824	1 843 200	1 843 200
Ciphertext size in VPK1 (bit)	55 390 464	110 778 624	470 400 000	940 804 800
Ciphertext size in GHWL (bit)	13 824	25 344	52 800	100 800
Ciphertext size in LDL (bit)	4626	4626	9625	9625
Ciphertext size in BCMS-LIDSASC (bit)	13 824	25 344	52 800	100 800
Signature size in BCMS-LIDSASC (bit)	54	54	54	54

Table 4 Time cost for different operations

Operation type	Time (μs)
Hash operation	$T_H = 36.74$
Matrix or vector modular multiplication	$T_M = 107.27$
Gaussian sampling algorithm operation	$T_s = 9.53$
Polynomial modular multiplication	$T_{mul} = 2.35$

Table 5 Time comparison of computation cost

Scheme	Total cost
VPK1	$3T_M + 3T_s + 5nT_{mul} + nT_H$
GHWL	$T_M + (13n + 7)T_{mul}$
LDL	$3T_s + (3n + 5)T_{mul} + 8T_H$
BCMS-LIDSASC	$2T_s + (2n + 4)T_{mul} + 6T_H$

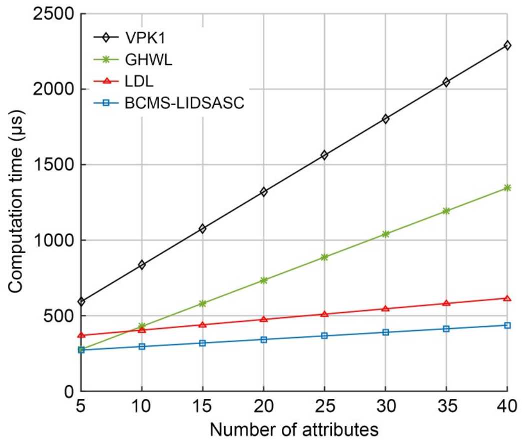

Fig. 2 Total time comparison of several schemes

9 Conclusions

BCMS-LIDSASC uses the blockchain, IPFS, and smart contracts to store sensitive data. It can effectively solve the data leakage and forgery problems in electronic healthcare systems and ensure data searchability and fine-grained access control. Malicious third parties or compromised servers cannot access the user data. Analysis shows that our scheme outperforms similar schemes.

In the future, we will continue to focus on security problems in the electronic healthcare field and investigate storage challenges facing blockchain technology.

Contributors

Huifang YU and Xiaoping BAI devised BCMS-LIDSASC, analyzed its security and performance, and drafted, revised, and finalized the paper.

Conflict of interest

Both authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Agbo CC, Mahmoud QH, Eklund JM, 2019. Blockchain technology in healthcare: a systematic review. *Healthcare*, 7(2): 56. https://doi.org/10.3390/healthcare7020056

Ali I, Lawrence T, Omala AA, et al., 2020. An efficient hybrid signcryption scheme with conditional privacy-preservation for heterogeneous vehicular communication in VANETs. *IEEE Trans Veh Technol*, 69(10):11266-11280. https://doi.org/10.1109/TVT.2020.3008781

Chen ZW, Wu AX, Li YF, et al., 2021. Blockchain-enabled public key encryption with multi-keyword search in cloud computing. *Secur Commun Netw*, 2021:6619689. https://doi.org/10.1155/2021/6619689

Cheng SP, Li MD, Duan YW, 2019. An improved scheme of searchable encryption algorithm based on NTRU. *J Phys Conf Ser*, 1345:042008. https://doi.org/10.1088/1742-6596/1345/4/042008

Chinnasamy P, Deepalakshmi P, Dutta AK, et al., 2022. Ciphertext-policy attribute-based encryption for cloud storage: toward data privacy and authentication in AI-enabled IoT system. *Mathematics*, 10(1):68. https://doi.org/10.3390/math10010068

Dohare I, Singh K, Ahmadian A, et al., 2022. Certificateless aggregated signcryption scheme (CLASS) for cloud-fog centric Industry 4.0. *IEEE Trans Ind Inform*, 18(9):6349-6357. https://doi.org/10.1109/TII.2022.3142306

Guo KY, Han YL, Wu RM, et al., 2022. CD-ABSE: attribute-based searchable encryption scheme supporting cross-domain sharing on blockchain. *Wirel Commun Mob Comput*, 2022:6719302. https://doi.org/10.1155/2022/6719302

How HB, Heng SH, 2022. Blockchain-enabled searchable encryption in clouds: a review. *J Inform Secur Appl*, 67:103183. https://doi.org/10.1016/j.jisa.2022.103183

Kumar R, Tripathi R, 2021. Towards design and implementation of security and privacy framework for Internet of Medical Things (IoMT) by leveraging blockchain and IPFS technology. *J Supercomput*, 77(8):7916-7955. https://doi.org/10.1007/S11227-020-03570-X

Li CY, Dong MX, Li J, et al., 2022. Efficient medical big data management with keyword-searchable encryption in healthchain. *IEEE Syst J*, 16(4):5521-5532. https://doi.org/10.1109/JSYST.2022.3173538

Li FQ, Liu KM, Zhang LP, et al., 2022. EHRChain: a blockchain-based EHR system using attribute-based and homomorphic cryptosystem. *IEEE Trans Serv Comput*, 15(5):2755-2765. https://doi.org/10.1109/TSC.2021.3078119

Li RN, Wang FQ, Zhang RJ, et al., 2022. NTRU-based fully homomorphic signature. *Secur Commun Netw*, 2022:9942717. https://doi.org/10.1155/2022/9942717

Li XY, 2022. Attribute-based encryption scheme on NTRU lattice. *Mod Electron Tech*, 45(19):77-82 (in Chinese).

https://doi.org/10.16652/j.issn.1004-373x.2022.19.015

Lu XF, Fu SB, 2021. A trusted data access control scheme combining attribute-based encryption and blockchain. *Netinfo Secur*, 21(3):7-14 (in Chinese). https://doi.org/10.3969/j.issn.1671-1122.2021.03.002

Miao YB, Ma JF, Liu XM, et al., 2020. Attribute-based keyword search over hierarchical data in cloud computing. *IEEE Trans Serv Comput*, 13(6):985-998. https://doi.org/10.1109/TSC.2017.2757467

Price WN, Cohen IG, 2019. Privacy in the age of medical big data. *Nat Med*, 25(1):37-43. https://doi.org/10.1038/s41591-018-0272-7

Varri US, Pasupuleti SK, Kadambari KV, 2021. CP-ABSEL: ciphertext-policy attribute-based searchable encryption from lattice in cloud storage. *Peer-to-Peer Netw Appl*, 14(3): 1290-1302. https://doi.org/10.1007/s12083-020-01057-3

Varri US, Pasupuleti SK, Kadambari KV, 2023. Practical verifiable multi-keyword attribute-based searchable signcryption in cloud storage. *J Amb Intell Human Comput*, 14(9): 11455-11467. https://doi.org/10.1007/s12652-022-03715-1

Wang HJ, Dong XL, Cao ZF, 2020. Multi-value-independent ciphertext-policy attribute based encryption with fast keyword search. *IEEE Trans Serv Comput*, 13(6):1142-1151. https://doi.org/10.1109/TSC.2017.2753231

Wang WZ, Xu H, Alazab M, et al., 2022. Blockchain-based reliable and efficient certificateless signature for IIoT devices. *IEEE Trans Ind Inform*, 18(10):7059-7067. https://doi.org/10.1109/TII.2021.3084753

Yang Y, Sun JG, Liu ZC, et al., 2022. Practical revocable and multi-authority CP-ABE scheme from RLWE for cloud computing. *J Inform Secur Appl*, 65:103108. https://doi.org/10.1016/j.jisa.2022.103108

Yu HF, Wang WK, Zhang Q, 2022. Certificateless anti-quantum ring signcryption for network coding. *Knowl-Based Syst*, 235:107655. https://doi.org/10.1016/j.knosys.2021.107655

Zhang AQ, Lin XD, 2018. Towards secure and privacy-preserving data sharing in e-health systems via consortium blockchain. *J Med Syst*, 42(8):140. https://doi.org/10.1007/s10916-018-0995-5

Zhang XJ, Xu CX, 2018. Trapdoor security lattice-based public-key searchable encryption with a designated cloud server. *Wirel Pers Commun*, 100(3):907-921. https://doi.org/10.1007/s11277-018-5357-6

Zhou YH, Li N, Tian YM, et al., 2020. Public key encryption with keyword search in cloud: a survey. *Entropy*, 22(4): 421. https://doi.org/10.3390/e22040421