Huifang YU, Xiaoping BAI, 2024. Identity-based searchable attribute signcryption in lattice for a blockchain-based medical system. *Frontiers of Information Technology & Electronic Engineering*, 25(3):461-471.
https://doi.org/10.1631/FITEE.2300248

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

Key words: Blockchain; Identity-based searchable attribute signcryption; Distributed storage; NTRU lattice

Huifang YU
E-mail: yuhuifang@xupt.edu.cn
ORCID: https://orcid.org/0000-0003-4711-3128

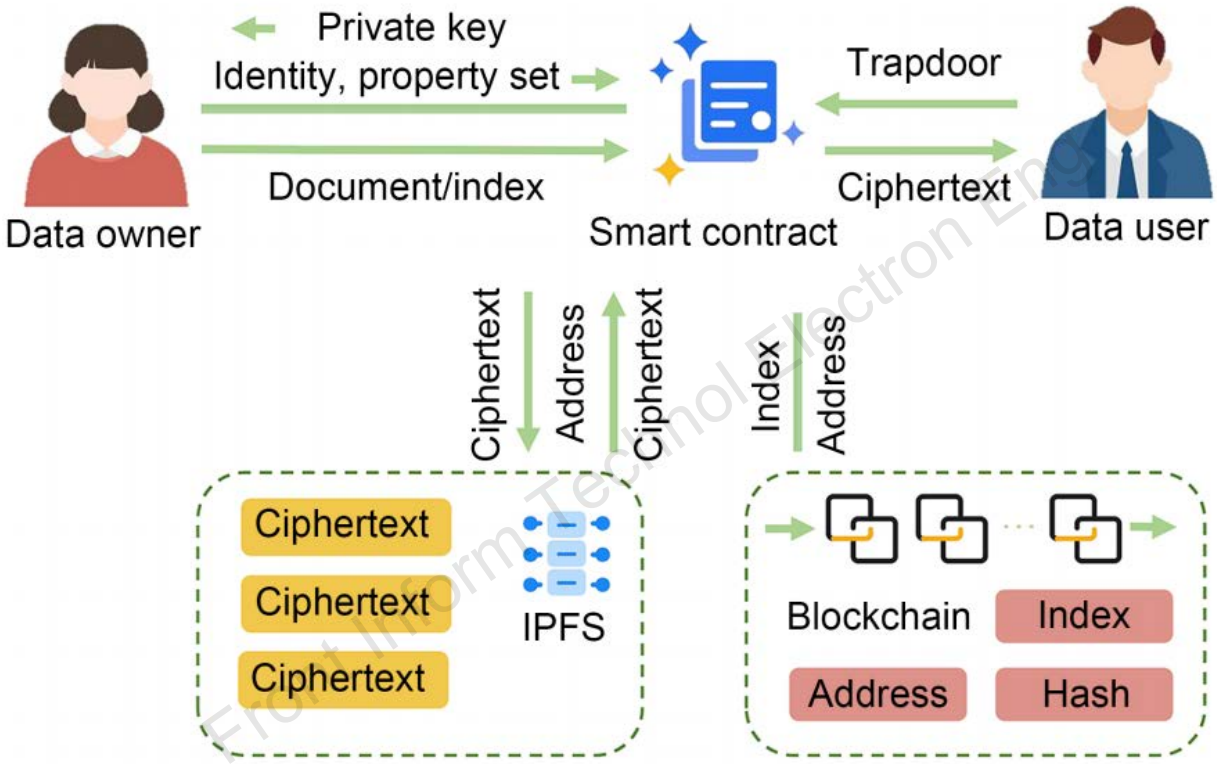
Motivation

- Security in data transmission and storage processes within the medical field is particularly crucial. Attribute-based searchable encryption ensures the searchability of user data and fine-grained access control, yet it does not guarantee data authenticity. Establishing a secure, convenient, and efficient data transmission scheme remains a challenge.
- Existing solutions often rely on trusted third-party entities to issue user keys and certificates. However, such centralized institutions are susceptible to various attacks. Integrating the blockchain technology, smart contracts, and interplanetary file system (IPFS) technology to achieve decentralization and distributed storage is crucial for user data protection.

Main idea

- We construct identity-based searchable attribute signcryption in lattice for a blockchain-based medical system (BCMS-LIDSASC), in which the blockchain and smart contract are used to achieve decentralization with no intermediary. IPFS technology is used for distributed storage. Users can complete the control over their data, and 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 electronic medical record (EMR) system.
- Our scheme has anti-quantum security and can effectively prevent the leakage of patient information.
- Attributed-based searchable signcryption is used to ensure confidentiality, integrity, and authenticity relevant to the data. Our scheme can also ensure searchability and fine-grained access control.

Framework



BCMS-LIDSASC includes four main entities: data owner, data user, blockchain, and IPFS distribution storage system.

Framework

- Data owner (DO) can be either a patient who owns the EMRs 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) 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.

Framework

- Blockchain stores the storage address, ciphertext hash, and index value generated by IPFS. 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 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.

Method

1. Setup

The administrator carries out the following operations to initialize the smart contract: The administrator selects the parameters and runs $TrapGen(N, q, \sigma)$ to obtain system public key h and master private key B. System parameters $\psi = \{h, p, H_1, H_2, H_3, H_4, H_5\}$.

2. KeyGen

The user sends its identity and attribute set to a smart contract. Smart contract uses the Gaussian sampling algorithm and carries out the calculations to obtain the private key and public key.

3. Signcrypt

DO uses access structure (M, ρ), system public key, and private key of DU to carry out a series of calculations and obtains ciphertext CF.

Method

4. IndexGen

DO chooses keyword k to calculate index I. Afterwards, DO uploads CF and I to the smart contract.

5. Trapdoor

DU chooses keyword k' and uploads trapdoor T to the smart contract.

6. Search

Smart contract receives trapdoor T and executes the search algorithm. Every search command has a time limit, and the smart contract searches for matching. If the limit is exceeded, the search will be terminated.

7. Unsigncrypt

DU unlocks the received ciphertext CF and verifies the correctness of the signature. If the verification passes, DU confirms the receipt of message m.

Major results

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

Major results

Table 2 Communication cost comparison

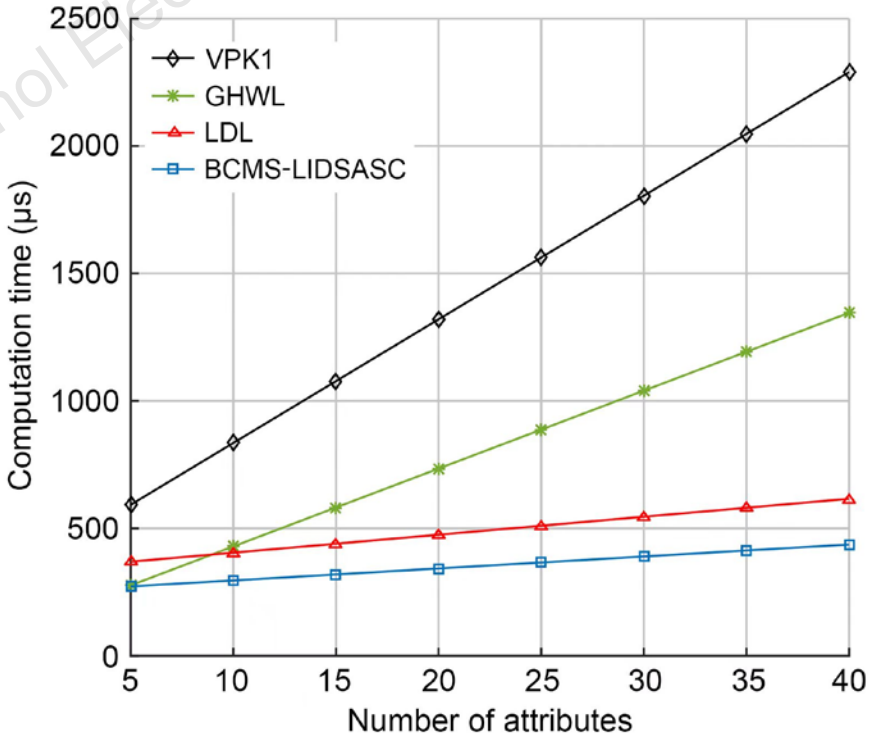
Scheme	IndexGen size (bit)	Trapdoor size (bit)	Ciphertext size (bit)	Signature size (bit)
VPK1	$(2M+n)N\log_2 q$	$MN\log_2 q$	$(kM+1)N\log_2 q$	
GHWL	$(n+\log_2 q_1)N\log_2 q_2$	$(n+1)N\log_2 q_2$	$(n+1)N\log_2 q_1$	
LDL	$(2N+1)\log_2 q$	$(2N^2+1)\log_2 q$	$(2N+1)\log_2 q$	
BCMS-LIDSASC	$(2N+1)\log_2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

Table 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	$2T_s + (2n+4)T_{mul} + 6T_H$



Conclusions

BCMS-LIDSASC uses the blockchain, IPFS, and smart contracts to store the sensitive data, and effectively solves the problems of data leakage and forgery in electronic healthcare systems. Our scheme ensures data searchability and fine-grained access control. Malicious third parties or compromised servers cannot access the user data. Our scheme outperforms similar schemes.

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