Article | Received 14 November 2024; Accepted 13 February 2025; Published 19 February 2025
https://doi.org/10.55092/blockchain20250004

A blockchain-enabled secure searchable-encryption-based data transaction protocol

Xingyun Hu[1], Yidan Chen[1], Haopeng Fan[1], Hualin Xu[2], Haoyuan Xue[1], Qifeng Tang[3], Siqi Lu[1]* and Yongjuan Wang[1]

[1] Henan Key Laboratory of Network Cryptography Technology, Information Engineering University, Zhengzhou, China
[2] Faculty of Information Science and Engineering, Ocean University of China, Qingdao, China
[3] National Engineering Laboratory for Big Data Distribution and Exchange Technologies, Shanghai Data Exchange Corporation, Shanghai, China

* Correspondence author; E-mail: 080lusiqi@sina.com

Highlights:

- Keyword-based encrypted data retrieval – data demanders efficiently search encrypted datasets using keywords without decryption, enhancing flexibility and reducing computational overhead.
- Tamper-proof transaction integrity – leveraging blockchain's immutability, the searchable-encryption-based data transaction protocol (SDTP) records all transactions on-chain to prevent denial-of-transaction and data tampering, fostering trust in decentralized ecosystems.
- Formal security guarantees – the protocol's security is rigorously validated via the Tamarin verification tool, ensuring resistance against impersonation, data leakage, and man-in-the-middle attacks.

Abstract: With the development of artificial intelligence and big data, data has become an important part of production factors, and the sharing and transaction of data have a very high importance. Through the storage service of the cloud data transaction platform, users can send data to the cloud platform remotely, and flexibly access and transmit data through the Internet anytime and anywhere. However, this approach faces growing data security concerns. When users transmit data to the cloud, they will not have full control over their data. Data stored in the cloud may be altered, deleted, leaked, or misappropriated, especially in public cloud environments. Many current data transaction platforms simply adopt a decentralized model to avoid this problem, but still perform poorly in the face of massive transaction scenarios. In addition, when the data demander receives the required data, there is the problem of denying the transaction, which challenges the availability of the data transaction platform and affects the trust of the data transaction participants in the transaction platform. This paper proposes a secure searchable-encryption-based data transaction protocol (SDTP) utilizing blockchain technology and searchable encryption. In the proposed protocol, the transaction platform does not gain access to the provider's raw data, and the data provider has all decision-making rights over the data. The data demander can search for the target encrypted data using only keywords before receiving the original data authorized by the data provider. In addition, blockchain technology, with its decentralized and tamper-proof characteristics, has

made important contributions to the transformation of traditional centralized data transaction platforms, and the entire data transaction process is recorded on the blockchain, effectively preventing problems such as demander denial and data tampering. In this paper, a formal verification tool is used to ensure that the proposed protocol meets the ideal security standard expected by the secure data transaction protocol, and the security of the protocol against attacks is proved from the perspective of non-formal theoretical analysis.

Keywords: searchable encryption; blockchain; data transaction; data sharing; formal analysis

1. Introduction

Traditional cloud service providers (CSPS) typically use encrypted channels to upload the data owner's data to cloud storage servers, and then utilize access control technologies and intrusion prevention systems to secure the storage of big data in the cloud. While data is encrypted both during the upload process and during its storage in the cloud, encryption keys are typically created or negotiated by cloud service providers, making it relatively easy for them to decrypt the data. In June 2020, Japanese centralized exchange coincheck was hacked. More than 2000 customers' personally identifiable information fell into the hands of the attackers. In 2022, the database of PayBito, a data exchange in the United States, was attacked by ransomware, and the personal information of more than 120,000 users was illegally stolen. Many privacy breaches show that market forces cannot protect cloud service providers from risks such as potential data loss and privacy breaches. Although service providers will provide services according to user needs and regulations, they still try to understand or collect users' privacy. Therefore, it is objectively believed that cloud storage service providers (platforms) are honest and curious.

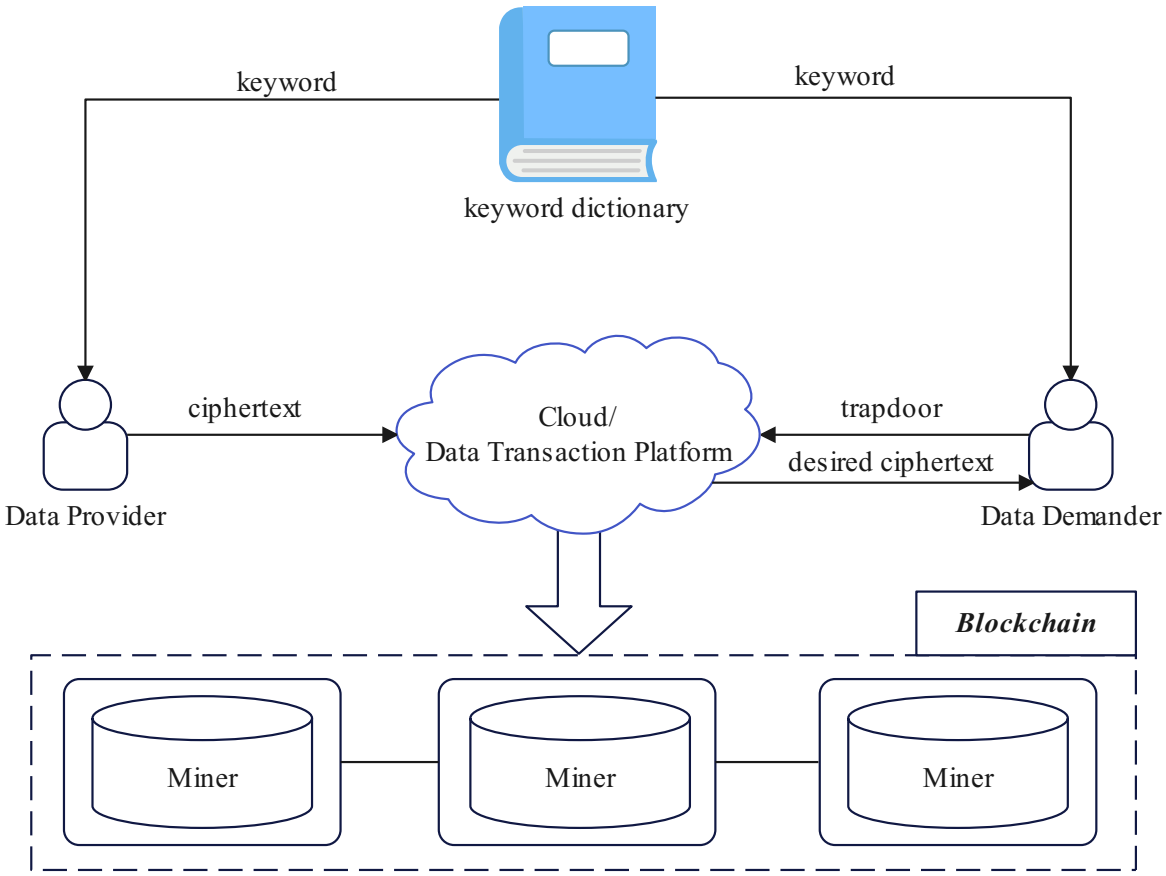

Figure 1. Typical searchable encryption (SE) schemes.

A typical data transaction platform consists of three parties: data providers, transaction platforms and data demanders [1]. Specifically, the provider transmits the data set to a trusted data transaction platform, and the data demander selects the data product of interest and places an order online. After receiving the transaction application and payment from the demand side, the data transaction platform transmits the target data to the demand side under the authorization of the provider, and pays the corresponding amount to the data provider (after deducting the platform commission). When data is transmitted and stored in clear text on cloud platforms, sensitive information is vulnerable to being compromised by attackers, unauthorized users, and semi-trusted cloud providers . In contrast, when data is uploaded in encrypted form, it is difficult for authorized users to find the specific encrypted information they need among the numerous ciphertexts stored on the platform [2]. Encrypted data faces the problem of difficult retrieval, which limits the availability and flexibility of data to some extent. For users, to retrieve encrypted data, the basic method is to download all the relevant ciphertext stored on the cloud platform, then decrypt the entire encrypted data set, and continue to operate until the target data is found. However, this approach comes with a lot of communication and computational overhead, which is extremely impractical.

To solve this problem, Boneh *et al.* [3] introduce public key encryption with keyword search (PEKS), a practical mechanism designed to facilitate searching encrypted data on cloud storage platforms, which can be applied to the data demander to retrieve the data from the provider in ciphertext. In a typical searchable encryption (SE) scheme, as shown in Figure 1, the data provider selects a keyword from a public dictionary, creates a search ciphertext, and empresses the selected keyword into that ciphertext [2]. At the same time, the data demander selects a keyword from the dictionary, generates a search trap linking to that keyword, and sends it to the transaction platform. The platform then uses the search trap to find the relevant ciphertext. The platform proposed in this paper also uploads the entire transaction process to the blockchain for storage.

This paper examines recent advancements in key cryptographic technologies related to data transactions and proposes a secure data transaction protocol scheme utilizing searchable encryption for various transaction scenarios. This is a unified system that integrates searchable encryption, blockchain technology, homomorphic encryption technology and hash computing technology. The main challenges of the system include:

1) Data providers are reluctant to disclose privacy concerns about real data to transaction platforms. Data providers often hold large amounts of sensitive information, which may involve important areas such as personal privacy, trade secrets, or national security. As a result, data providers are deeply concerned about providing data to transaction platforms, fearing that if the real data is exposed, it could have serious consequences. Transaction platforms play a key role in data transactions, but some dishonest transaction platforms may engage in bad behavior and may use their management and storage authority in data transactions to secretly cache data sets provided by data providers without their knowledge and authorization. To address the first challenge, common solutions employ techniques such as homomorphic encryption and searchable encryption [2][4], or rely entirely on blockchain for data transactions [5]. Considering the integrity and confidentiality that the data transaction system needs to meet and the limitations of relying solely on the blockchain for transactions in the big data transaction scenario, we still use the centralized transaction platform. In traditional data transaction systems, a centralized platform usually relies on a trusted third party to oversee the keys of providers, consumers, and the platform itself [4][6][7]. Our solution enables data transaction participants to manage public and private key pairs in the form of local storage, and can customize key management strategies according to specific security requirements. If participants stop the dynamic manipulation of data, the model allows them to remove keys from local storage. The original data is always transmitted and stored in a confidential state, the transaction platform has no access to the original data, and the data demander can only

retrieve the data through keyword search. For example, consider a data provider who wants to share medical records. The provider encrypts the records using searchable encryption, and the data demander can search for specific keywords (e.g., "diabetes") without decrypting the entire dataset.

2) The rejection problem of the data demander after obtaining the data. Dishonest claimants may refuse to acknowledge and refuse to deal after obtaining the raw data. The rejection behavior of data demanders has a serious negative impact on the data transaction ecosystem. From the perspective of the whole market, this behavior destroys the trust mechanism of data transactions, increases the cost and risk of data transactions, and hinders the healthy development of data markets. To address the second challenge and ensure the non-repudiation of the data transaction system, we propose the use of digital signature technology to verify that the data has been obtained by the demand side. In addition, blockchain technology will be used to record the entire data transaction process. If the demand side is still dishonest after receiving the data, the transaction platform can take measures such as stopping transactions and freezing funds, thus ensuring and improving the availability of the platform. For instance, when a data demander requests and receives data, the transaction is recorded on the blockchain, providing a tamper-proof record of the transaction.

3) Privacy of data transaction protocol at logical layer. The privacy analysis of data transaction systems usually rely on the application of laminar flow data capture, mainly checking whether the data privacy is guaranteed in the actual data transaction process. However, in order to conduct a more comprehensive privacy analysis, it is necessary to consider whether there are privacy disclosure vulnerabilities at the logical level of protocol design, so as to avoid the risk of privacy disclosure at the details that cannot be discovered by capturing packets. In order to solve the third challenge, we conduct formal security analysis of the proposed protocol using the Tamarin tool and perform logical layer simulation verification for key privacy, data security, protocol availability, and so on.

The main contributions of this work are summarized as follows.

1. We propose a blockchain-based data transaction protocol based on searchable encryption. The protocol relies on Searchable Public-Key Ciphertexts with Hidden Structures(SPCHS) and blockchain technology to ensure that the platform cannot steal data, the supplier cannot tamper with and forge data, and the demander cannot deny the transaction facts.

2. We perform an extensive informal analysis of our designed protocol and show that our proposed SDTP protocol can resist known attack vectors.

3. We formally analyze our proposed SDTP protocol in the automatic verification tool Tamarin, and prove the security of SDTP in terms of protocol executability, data confidentiality, key confidentiality and privacy confidentiality.

The rest structure of this paper is composed of: Section 2 reviews related work, while Section 3 discusses the key concepts that underpin the design of the proposed protocol. In Section 4, a thorough overview of the protocol is presented, breaking down each phase in detail. Section 5 offers a security analysis of the protocol. Finally, Section 6 summarizes the conclusions drawn from this study.

2. Related work

Public-key-based searchable encryption: Searchable encryption (SE) schemes [8] enable secure searching of outsourced data through encryption. Typically, these schemes compromise some security guarantees and permit limited information leakage to enhance search efficiency. Boneh *et al.* introduced the first public key encryption scheme that supports keyword search (PEKS), which has since been refined in terms of sublinear search capabilities [9], efficiency [10], and security [11]. In contrast, the Searchable Public key Ciphertext with Hidden Structure (SPCHS) [12] enables rapid keyword searches while maintaining the semantic integrity

of encrypted keywords. By concealing relationships among the data, SPCHS generates ciphertexts that are searchable, thereby minimizing the statistical information revealed to the search algorithm about the group members. The search complexity in SPCHS relies on the actual range of ciphertexts that contain the queried keywords, rather than the variety of all ciphertexts. Consequently, we ultimately choose the SPCHS system for our searchable encryption scheme.

Blockchain-supported data transaction: Lu *et al.* [13] and Dixit *et al.* [14] created a blockchain-based cloud storage protocol tailored for industrial Internet of Things (IoT) sensors, employing group signature schemes and integrating smart contracts with proxy re-encryption to enable secure data sharing in industrial settings. Meanwhile, Ma *et al.* [15] proposed a dynamic data-sharing scheme for smart factories that utilizes blockchain for authentication, storing ciphertext indexes and public keys to prevent tampering; their tracking algorithm effectively identifies and blacklists malicious users. Zhao *et al.* [16] presented a blockchain-based fair data transaction protocol for the big data market, utilizing ring signatures to enhance the privacy of data providers while promoting fairness in decentralized applications (DApps) and employing similarity learning to ensure the availability of transaction data. Wang *et al.* [17] introduced a data transaction scheme based on the Bitcoin framework, where data is first encrypted with a symmetric key and then further secured with RSA encryption. Only the user who completes the payment can access the private key required for decryption. Lastly, Dai *et al.* [18] developed a blockchain-based data transaction ecosystem (SDTE) that allows data buyers to access only the analysis results generated by Intel SGX. In this system, data from various sellers is processed in separate SGX environments, and the most-voted result is considered final, rewarding the seller who provided that result.

Recent advances in blockchain-based searchable encryption: recent works have explored the integration of blockchain and searchable encryption for various applications. For instance, Jiang *et al.* [19] proposed a bloom filter-based multi-keyword search protocol for blockchain-based intelligent transportation systems, significantly improving search efficiency and privacy preservation. However, their work primarily focuses on single-keyword searches and does not address the issue of data transaction integrity. Similarly, Hu *et al.* [20] introduced a decentralized privacy-preserving search scheme using Ethereum smart contracts, ensuring the correctness and fairness of search results. While their approach provides a robust solution for decentralized search, it does not consider the complexities of multi-party data transactions, particularly the issue of demander denial. Chen *et al.* [21] proposed a blockchain-based searchable encryption scheme for electronic health records sharing, ensuring data privacy and integrity in healthcare applications. However, their work is limited to the healthcare domain and does not address the broader challenges of data trading platforms. Additionally, Chen *et al.* [5] presented a secure and efficient blockchain-based data trading approach for the Internet of Vehicles, focusing on the security and efficiency of data transactions in vehicular networks. While their work provides valuable insights into secure data trading, it does not incorporate searchable encryption techniques.

While several prior works have explored the integration of blockchain and searchable encryption, our proposed protocol, SDTP, introduces several novel techniques to address the unique challenges of data transaction platforms:

1. Blockchain-based transaction integrity: our protocol leverages blockchain's decentralized and tamper-proof characteristics to record the entire data transaction process. This ensures that the transaction platform cannot tamper with the data, and the data demander cannot deny receiving the data after the transaction is completed.

2. Formal security verification: we introduce a formal verification tool to ensure that the proposed protocol meets the ideal security standards expected for secure data transactions. This is a significant advancement over existing approaches, which often rely on informal security analysis.

3. Dynamic data updates: our protocol supports dynamic updates to the encrypted data, allowing data providers to add or remove data without compromising the security or efficiency of the system. This is particularly important in data transaction scenarios where data is frequently updated.

3. Preliminaries

3.1. Searchable encryption

With the swift progress of cloud services, a growing number of users have decided to store their data in cloud platforms [22]. Although this brings convenience, it also makes data leakage incidents occur frequently. In order to protect data privacy, encryption technology has attracted much attention. However, as data volume and distribution continue to grow, traditional encryption methods struggle to address the demands of large-scale data management. This challenge has led to the emergence of searchable encryption technology [23].

Searchable encryption is an innovative concept that allows users to retrieve data from an untrusted third party without decryption. Let the data owner be D_O, the data user be D_U, and the cloud platform server be C_S. In the searchable encryption scheme, D_O uploads the encrypted data $E(D)$ to C_S. As a cloud service consumer, D_U can retrieve the uploaded data after being authorized by D_O. D_U submits a search token T (which can be regarded as a keyword query) to C_S, and C_S uses T to retrieve the encrypted data stored internally and returns the matching results.

Among them, asymmetric searchable encryption mainly relies on asymmetric encryption algorithm, namely PEKS. The fundamental operations consist of initialization, public key encryption, trapdoor generation using the private key, and search steps. In asymmetric searchable encryption, the public key P_K is employed to encrypt the plaintext data M and produce the corresponding ciphertext [24]. Meanwhile, the private key S_K is utilized for decrypting the ciphertext and generating keyword trapdoors. In contrast to symmetric searchable encryption, the asymmetric searchable encryption algorithm can create a searchable label L based on the public key and keyword K, allowing the encrypted data to be stored on a cloud platform server. That is $E(D,L) = Enc(P_K, M, K) \rightarrow C_S$, a user can generate a trap gate $T_{K,S_K} = Trap(S_K, K)$ based on the keyword and the searchable private key to retrieve the encrypted data. PEKS not only safeguards data privacy but also allows the demander to locate the provider's information. The provider generates searchable ciphertext related to critical data using the demander's public key and then uploads it to the platform. The demander, in turn, utilizes his key to create a keyword trapdoor, which they upload to search for the corresponding ciphertext.

Searchable encryption technology has a wide range of application scenarios, which can not only protect data privacy, but also achieve efficient data search and management. In cloud computing, it enables users to securely store and manage sensitive data while ensuring privacy protection. For data sharing, it enhances security and privacy measures, facilitating efficient data retrieval and sharing. Regarding location privacy, it allows users to access necessary services and information without compromising their personal location data.

3.2. Homomorphic encryption

Homomorphic encryption is a unique technique that allows for data processing without the need to access the actual data itself. Suppose the plaintext numbers a and b are added to obtain $c = a + b$, then encrypted into $Enc(a)$ and $Enc(b)$ by homomorphic encryption, and $Enc(c')$ is obtained by operating on the ciphertext. c' represents the result of performing homomorphic operations on encrypted data. When decrypted, c' is equal to c, that is, $Dec(Enc(a) + Enc(b)) = a + b$, which is the basic operation property of homomorphic encryption. It is like putting an "encryption cloak" on the data with special functions, so that the data can still be

used for meaningful calculation in the encrypted state.

General encryption technology needs to decrypt the data before calculation, which has disadvantages. If the decryption process goes wrong, the data will be exposed, and the decryption and re-encryption process may lead to the risk of information leakage. In contrast, homomorphic encryption provides a higher level of protection by performing computations on ciphertext without exposing the original data. Homomorphic Encryption is divided into Fully Homomorphic Encryption (FHE), Partially Homomorphic Encryption (PHE) and Somewhat Homomorphic Encryption (SHE). While FHE can perform any computation, PHE and SHE can only perform certain types of computations. Homomorphic encryption technology is widely used in secure cloud computing, privacy protection, data exchange and cryptography.

In the data transaction platform, data privacy protection and effective use are very important, and homomorphic encryption is just to meet this demand. The provider can upload the encrypted data to the platform, and the platform can perform various calculations and processing, such as data analysis and data mining, without decryption [16]. In a financial data transaction platform, banks can securely transmit encrypted customer transaction data to data analysis agencies. These agencies employ homomorphic encryption to perform statistical analyses on the ciphertext, such as calculating total transaction amounts and transaction frequency, without ever accessing the plaintext data. This method not only protects customer privacy but also promotes safe transactions and facilitates valuable insights from the data.

3.3. Blockchain

Blockchain is a distributed ledger technology that consists of a series of blocks of data, with each block containing transaction information from a particular time frame. Its key characteristics include decentralization, immutability, and transparency. Decentralization implies that no single entity governs the entire network; instead, it is sustained by numerous nodes. For instance, in the Bitcoin network, miner nodes globally engage in transaction validation and the creation of new blocks.

The immutability is due to the cryptographic techniques used in the blockchain. Let the current block be B_n, which contains $H(B_{n-1})$ of the previous block B_{n-1}. $H(B_{n-1})$ represents the hash value of the previous block. Once data is stored on the blockchain, altering it becomes a challenging task. Any modification to one block will cause the hash of subsequent blocks to change and thus be easily detected. Transparency means that the transaction information on the blockchain is visible to all nodes. Although the specific content of the transaction may be encrypted, the existence and history of the transaction are publicly available.

The work of blockchain is mainly divided into the steps of transaction generation, broadcast, verification and packaging into blocks. When a user initiates a transaction, such as sending a certain amount of bitcoins to another user in the Bitcoin network, this transaction information is broadcast to nodes throughout the network. Once the transaction information is received, the node verifies its legitimacy, checking factors like the sender's balance. Valid transactions are then placed into a pool, awaiting packaging. Miner nodes find a random number that satisfies certain conditions through competitive calculation, and this process is called mining. The miner who finds the random number collects transactions from the pool to form a new block, which is subsequently added to the blockchain. This new block is shared across the entire network, where other nodes check its validity. If the block is verified successfully, they incorporate it into their versions of the blockchain.

Blockchain technology is crucial in data transaction platforms. Its decentralized nature diminishes reliance on a single central authority, enhancing both the security and reliability of data transactions. Data can be stored on the blockchain, and multiple nodes work together to maintain the integrity of the data, avoiding the risk of data being tampered or deleted by a single authority. Additionally, the immutable nature of blockchain guarantees both the authenticity and traceability of data [25]. Every transaction record is securely stored on the

blockchain, making it resistant to tampering. This feature allows for the clear tracking of data sources and transaction processes, thereby increasing the overall credibility of the data.

4. Proposed protocol

Based on the cryptographic characteristics of public key searchable encryption, the data transaction protocol based on searchable encryption provides retrieval support for data demanders, ensures a supply-centered transaction system in the process of data transaction, and ensures that the data in the whole transaction is transmitted and stored in secret state. Each entity is described as follows:

(1) Data provider

The data provider owns the raw data and is responsible for encrypting it. Public key searchable encryption is used to prepare the data in a secret form for uploading to the transaction platform. Data providers must guarantee the security of information throughout both the encryption and transmission processes. They should choose reliable encryption algorithms and key management methods to prevent data from being illegally stolen or tampered with.

(2) Data demander

As a data demander, it has the right to purchase data and can submit search requests to the transaction platform according to its own needs. When the data demanders obtain the encrypted data that fulfills their requirements, they can decrypt and utilize it with the data provider's permission. The demander must adhere to applicable regulations regarding data usage and is prohibited from using the data for illegal or unauthorized purposes.

(3) Transaction platform

The transaction platform is responsible for receiving encrypted data uploaded by data providers and storing encrypted data. It is essential for them to guarantee both the security and availability of data. To prevent data loss or corruption, they should implement dependable storage technologies and effective backup strategies. At the same time, the transaction platform provides retrieval services for data demanders and carries out security supervision in the whole process. The transaction platform uses blockchain to record and track the entire transaction process.

Table 1. Notations.

Symbol	Description
Pro	Data provider
Dem	Data demander
Ser	Platform server
Key	Raw data keywords
A	Search trapdoor
B	Search ciphertext
M	Search plaintext
$Data$	Raw data
$Data\_Hash$	Hash value of raw data
$Data\_Hash2$	Hash value of fresh raw data
$Data\_Hash3$	Hash value of raw data before sent to the platform
$Data\_Hash4$	Hash value of the data obtained by the data demander

The detailed process of the data transaction protocol based on searchable encryption (Figure 2) is as follows.

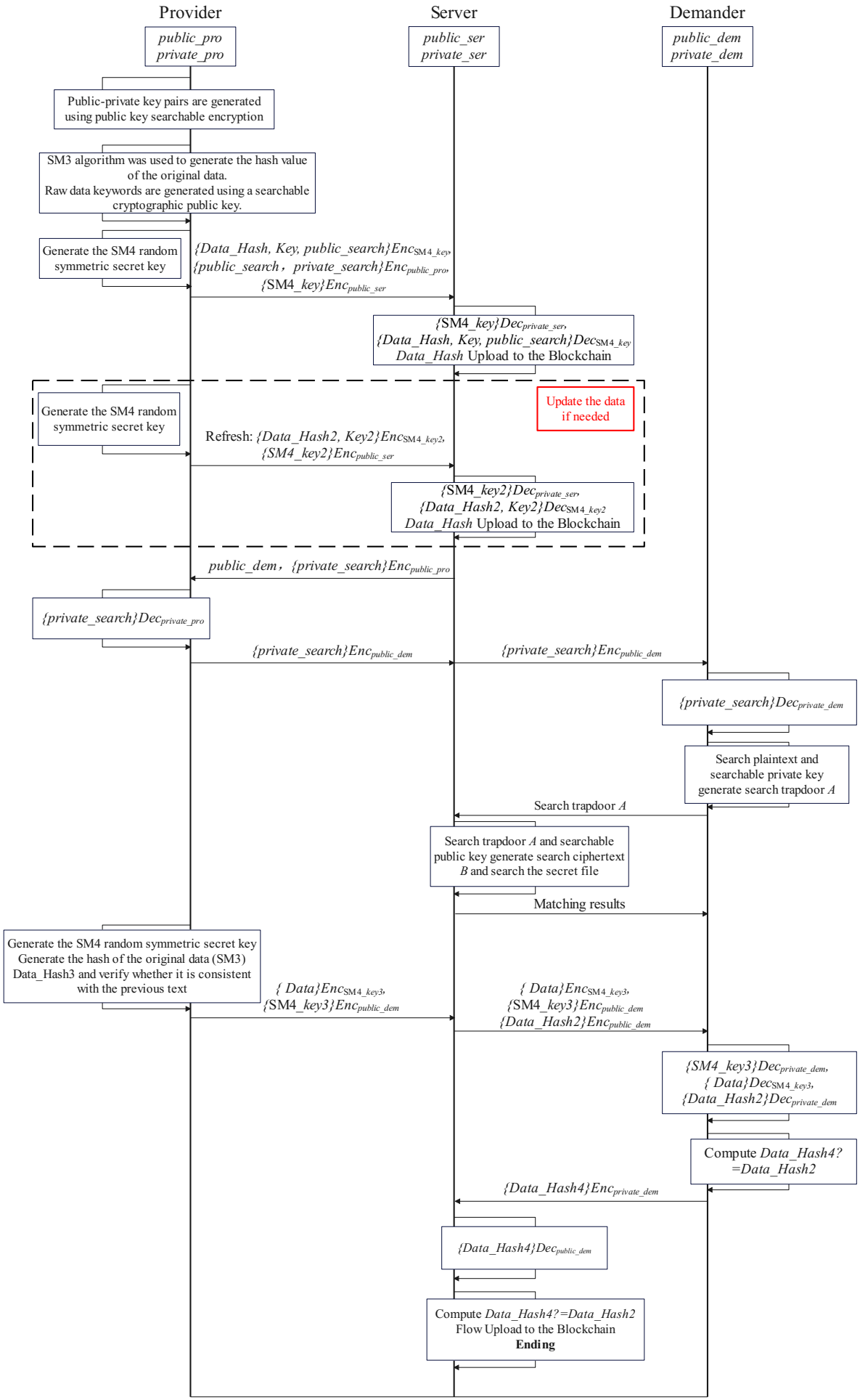

Figure 2. Flow chart of the data transaction protocol.

4.1. Data initialization

Step 1: The data provider and the data demander first complete user registration on the platform, and obtain the account and password of the login transaction platform, that is, the identity authentication certificate. Upon user login, the platform authenticates the individual's identity using a password.

Step 2: The public and private key pairs generated by the data provider, data demander and transaction platform using SM2 algorithm are $\{public\_pro, private\_pro\}$, $\{public\_dem, private\_dem\}$, $\{public\_ser, private\_ser\}$, respectively. The data provider and demander send their public keys to the transaction platform.

Step 3: The data provider runs searchable public key cryptography locally (e.g., the SPCHS system) to create a searchable encryption public and private key pair $\{public\_search, private\_search\}$, uses $public\_search$ to encrypt $Data$ keyword to generate the original data keyword Key, and uses the hash algorithm – SM3 algorithm to generate the original data hash value $Data\_Hash$. The original $Data$ is encrypted by using $public\_pro$ of the provider. The encrypted public-private key pair which can be searched and encrypted and the hash value $Data\_Hash$ of the data file are appended to generate the encrypted data file. The $private\_search$ is encrypted by the public key $public\_pro$ of the data provider. Since only the data provider possesses the private key, only he can decrypt the shipping data encrypted with the public key, making it impossible for the platform and the data demander to access the ciphertext.

Algorithm 1 Data Initialization

Require: Raw data ($Data$), keywords (Key), security parameters
Ensure: Encrypted data, searchable keys, and hash values

Generate public-private key pairs:
  $(public\_pro, private\_pro)$ for data provider ▷ Generate provider's key pair
  $(public\_dem, private\_dem)$ for data demander ▷ Generate demander's key pair
  $(public\_ser, private\_ser)$ for transaction platform ▷ Generate platform's key pair
Data provider generates searchable encryption keys:
  $(public\_search, private\_search) \leftarrow$ SPCHS_KeyGen() ▷ Generate searchable keys
Encrypt raw data keywords:
  $Key \leftarrow$ Encrypt$(public_search, keywords)$ ▷ Encrypt keywords using public key
Compute hash value of raw data:
  $Data\_Hash \leftarrow$ SM3$(Data)$ ▷ Compute hash using SM3 algorithm
Encrypt raw data using data provider's public key:
  $Enc\_Data \leftarrow$ Encrypt$(public_pro, Data)$ ▷ Encrypt raw data
Encrypt $private\_search$ using data provider's public key:
  $Enc\_private\_search \leftarrow$ Encrypt$(public_pro, private_search)$ ▷ Encrypt private search key
Store encrypted data, keys, and hash values locally. ▷ Local storage for secure access

4.2. Data upload

Step 1: The data provider sends a key pair $\{public\_search, private\_search\}_{Enc_{public\_pro}}$, encrypted with the platform's public key, to the platform. The platform then stores this key pair, which is encrypted using the $public\_pro$.

Step 2: The data provider will use $public\_search$, Key and $Data\_Hash$ obtained by the SM3 algorithm to upload to the platform through the SM4 algorithm. The SM4 key is encrypted through the public key $public\_ser$ of the transaction platform. The platform can decrypt the searchable encryption public key $public\_search$, the secret data (keyword) Key and the original data hash value $Data\_Hash$ by SM4 algorithm, but the platform can not further get the plaintext. This process ensures that the data is not stolen by the malicious platform. The platform side stores and backs up the secret data, and $Data\_Hash$ of is stored on the chain certificate. When the data provider updates the information, he computes the hash

value and keywords for the new data, encrypts it using the SM4 algorithm, and then uploads the encrypted data to the platform. The platform stores the updated secret data along with its hash value.

To verify the authenticity and reliability of the data submitted by providers, the platform can conduct random spot checks during regular maintenance. This process entails choosing a segment of the confidential data and then requesting the original data from the provider. If the raw data is not standard, the platform can refuse to distribute this data set.

Algorithm 2 Data upload

Require: Encrypted data (*Enc_Data*), searchable keys (*public_search*, *Enc_private_search*), hash value (*Data_Hash*)

Ensure: Encrypted data stored on the transaction platform

Data provider sends encrypted key pair to the platform:
  Send($\{public\_search, Enc\_private\_search\}, public\_ser$) ▷ Send keys to platform
Platform stores the encrypted key pair. ▷ Store keys securely
Data provider encrypts *Key* and *Data_Hash* using SM4:
  $Enc\_Key \leftarrow$ SM4_Encrypt($SM4\_key, Key$) ▷ Encrypt keywords
  $Enc\_Data\_Hash \leftarrow$ SM4_Encrypt($SM4\_key, Data\_Hash$) ▷ Encrypt hash value
Data provider sends encrypted *Key* and *Data_Hash* to the platform:
  Send($\{Enc\_Key, Enc\_Data\_Hash\}, public\_ser$) ▷ Send encrypted data to platform
Platform decrypts *Key* and *Data_Hash* using its private key:
  $Key \leftarrow$ SM4_Decrypt($private\_ser, Enc\_Key$) ▷ Decrypt keywords
  $Data\_Hash \leftarrow$ SM4_Decrypt($private\_ser, Enc\_Data\_Hash$) ▷ Decrypt hash value
Platform stores encrypted *Key* and *Data_Hash*, and uploads *Data_Hash* to the blockchain. ▷ Store and record data
Platform verifies data authenticity by requesting raw data from the provider if necessary. ▷ Optional verification

4.3. Data search

Step 1: The data demander sends the transaction application to the platform, and the platform forwards the transaction application to the data provider, and at the same time sends *public_dem* of the data demander to the provider.

Step 2: After the data provider agrees to the transaction, the platform establishes this transaction, and records and protects the transaction process through the blockchain technology. The data provider uses *public_dem* of the data demander to encrypt the searchable encryption private key *private_search*, and transmits it to the data demander through the platform.

Step 3: The data demander utilizes *private_dem* to decrypt and retrieve the searchable encrypted private key *private_search*. The search trapdoor A is generated by the search plaintext M and the searchable encrypted private key *private_search*, and the search trapdoor A is sent to the platform.

Step 4: The platform uses *public_search* and the search plaintext M to generate the search ciphertext B, and then combines the received search trapdoor A to retrieve the encrypted data, and returns the search results to the data demander.

Algorithm 3 Data search

Require: Data demander's search request, keywords (*Key*), searchable keys (*public_search*, *private_search*)
Ensure: Matching encrypted data

Data demander sends a search request to the platform:
  Send(*Search_Request*, *public_ser*) ▷ Send search request
Platform forwards the request to the data provider. ▷ Forward request to provider
Data provider agrees to the transaction and encrypts *private_search* using data demander's public key:
  $Enc\_private\_search \leftarrow \text{Encrypt}(public\_dem, private\_search)$ ▷ Encrypt private search key
Platform sends encrypted *private_search* to the data demander:
  Send(*Enc_private_search*, *public_dem*) ▷ Send encrypted key to demander
Data demander decrypts *private_search* using its private key:
  $private\_search \leftarrow \text{Decrypt}(private\_dem, Enc\_private\_search)$ ▷ Decrypt private search key
Data demander generates a search trapdoor A using *private_search* and search plaintext M:
  $A \leftarrow \text{Trapdoor}(private\_search, M)$ ▷ Generate search trapdoor
Data demander sends search trapdoor A to the platform:
  Send(A, *public_ser*) ▷ Send trapdoor to platform
Platform generates search ciphertext B using *public_search* and search plaintext M:
  $B \leftarrow \text{Search\_Ciphertext}(public\_search, M)$ ▷ Generate search ciphertext
Platform retrieves matching encrypted data using A and B:
  $Matching\_Data \leftarrow \text{Retrieve}(A, B)$ ▷ Retrieve matching data
Platform returns search results to the data demander. ▷ Return results to demander

4.4. Data transaction

Step 1: After receiving feedback from the platform, the data demander may apply to exchange all (or part of) the data. Firstly, the data provider generates SM4 random symmetric encryption key, encrypts *Data* with the symmetric encryption key, encrypts the symmetric key with *public_dem* of the data demander, and then sends it to the data demander.

Step 2: The Data demander decrypts the symmetric key with *private_dem*, then decrypts the symmetric key to get the original data $Data'$, calculates the hash value of the original data and encrypts it with the demander's private key *private_dem* before sending it to the transaction platform. The transaction platform uses *public_dem* of the demander to verify the identity of the demander, and determines whether the hash value matches the hash value on the blockchain. If the hash value is verified, the platform updates the asset balances in both the data provider's and demander's accounts, and subsequently records the transaction on the blockchain.

4.5. Technical integration details

The proposed SDTP protocol integrates three key cryptographic technologies: searchable encryption, homomorphic encryption, and blockchain tools. Below, we explain how these technologies are integrated into the protocol to ensure security, privacy, and efficiency.

4.5.1. Searchable encryption

Searchable encryption (SE) is a cryptographic technique that allows users to search over encrypted data without decrypting it. In SDTP, SE is used to enable data demanders to retrieve encrypted data using keywords. The integration of SE in SDTP is as follows:

Keyword encryption: The data provider encrypts keywords using a searchable encryption public key ($public_search$). This allows the data demander to generate a search trapdoor (A) using their private key ($private_search$) and search for encrypted data without revealing the keywords or the data content.

Trapdoor generation: The data demander generates a trapdoor (A) by combining the search plaintext (M) with their private key ($private_search$). The trapdoor is then sent to the transaction platform, which uses it to retrieve the matching encrypted data.

Algorithm 4 Data transaction

Require: Encrypted data (Enc_Data), data demander's request, transaction details
Ensure: Successful data transaction recorded on the blockchain
Data demander requests data exchange from the platform:
    Send($Exchange\_Request, public\_ser$) ▷ Send exchange request
Data provider generates a random symmetric key ($SM4\_key$) and encrypts raw data using $SM4\_key$:
    $Enc\_Data \leftarrow$ SM4_Encrypt($SM4\_key, Data$) ▷ Encrypt raw data
Data provider encrypts $SM4\_key$ using data demander's public key:
    $Enc\_SM4\_key \leftarrow$ Encrypt($public\_dem, SM4\_key$) ▷ Encrypt symmetric key
Data provider sends encrypted data and $SM4\_key$ to the data demander:
    Send($\{Enc\_Data, Enc\_SM4\_key\}, public\_dem$) ▷ Send encrypted data and key
Data demander decrypts $SM4\_key$ using its private key:
    $SM4\_key \leftarrow$ Decrypt($private\_dem, Enc\_SM4\_key$) ▷ Decrypt symmetric key
Data demander decrypts raw data using $SM4\_key$:
    $Data \leftarrow$ SM4_Decrypt($SM4\_key, Enc\_Data$) ▷ Decrypt raw data
Data demander computes hash value of raw data and encrypts it using $private\_dem$:
    $Data\_Hash \leftarrow$ SM3($Data$) ▷ Compute hash value
    $Enc\_Data\_Hash \leftarrow$ Encrypt($private\_dem, Data\_Hash$) ▷ Encrypt hash value
Data demander sends encrypted hash value to the platform:
    Send($Enc\_Data\_Hash, public\_ser$) ▷ Send encrypted hash to platform
Platform verifies the hash value against the blockchain record:
    if Verify($Data\_Hash, Blockchain\_Record$) **then** proceed. ▷ Verify hash value
Platform updates account balances and records the transaction on the blockchain. ▷ Record transaction

Search efficiency: By leveraging searchable encryption, SDTP ensures that data demanders can efficiently search for encrypted data without compromising data privacy. This is particularly important in scenarios where the data provider does not want to reveal the raw data to the transaction platform.

4.5.2. Homomorphic encryption

Homomorphic encryption (HE) allows computations to be performed on encrypted data without decrypting it. In SDTP, HE is used to enable secure data processing on the transaction platform. The integration of HE in SDTP is as follows:

Data processing: The transaction platform can perform computations (e.g., data aggregation or analysis) on encrypted data without accessing the raw data. This ensures that the data remains confidential even during processing.

Secure transactions: HE is used to encrypt sensitive transaction details (e.g., payment information) before they are stored on the blockchain. This ensures that the transaction details are protected from unauthorized access.

Privacy preservation: By using HE, SDTP ensures that the data provider's privacy is preserved throughout the data transaction process. The transaction platform cannot access the raw data, even when performing computations.

4.5.3. Blockchain tools

Blockchain technology is used in SDTP to ensure the integrity, transparency, and immutability of data transactions. The integration of blockchain tools in SDTP is as follows:

Transaction recording: All data transactions are recorded on the blockchain, ensuring that the transaction history is tamper-proof and transparent. This prevents issues such as data tampering or demander denial.

Decentralization: The use of blockchain eliminates the need for a centralized authority, reducing the risk of single points of failure and enhancing the security of the data transaction platform.

Smart contracts: Smart contracts are used to automate the data transaction process. For example, when a data demander requests data, the smart contract verifies the transaction details and ensures that the data provider is compensated fairly.

Data integrity: The blockchain ensures that the data stored on the transaction platform is authentic and has not been altered. This is achieved by storing the hash value of the data ($Data_Hash$) on the blockchain and verifying it during data retrieval.

By integrating searchable encryption, homomorphic encryption, and blockchain tools, SDTP provides a secure, efficient, and privacy-preserving solution for data transactions. These technologies work together to ensure that the data provider retains control over their data, the data demander can efficiently search for and retrieve encrypted data, and the transaction platform ensures the integrity and transparency of the transaction process.

5. Security analysis

This section offers a comprehensive security evaluation of the proposed protocol. We employ Tamarin, an automated verification tool, for formal safety assessment. In addition, we explore prevalent security vulnerabilities and threats related to authentication and access management, as well as strategies on how to address these challenges in decentralized environments such as data transaction platforms.

5.1. Formal security verification: simulation used by tamarin tool

We leverage the Tamarin tool, a widely recognized automated verification software, to evaluate the resilience of the proposed protocol against different attack vectors in the authentication framework. Simulations using this tool show that the proposed protocol effectively protects against threats like impersonation, passive secret disclosure, and man-in-the-middle attacks, providing robust security.

The Tamarin prover is a powerful tool for symbolically modeling and analyzing security protocols. It requires a detailed model of the protocol, outlining the operational specifications for different roles, the characteristics of potential adversaries, and the properties that the protocol must satisfy. Tamarin can automatically generate proofs to demonstrate compliance with these specified properties, even when multiple instances of protocol roles interact with simultaneous adversarial actions.

Tamarin provides extensive capabilities for modeling and analyzing security protocols. It employs a rich language based on multiset rewriting rules to specify both protocols and adversaries, establishing labeled transition systems where states include adversary knowledge, network messages, newly generated values, and symbolic representations of protocol states. The attacker interacts with the protocol by altering existing messages and generating new ones. Additionally, Tamarin accommodates equational specifications for certain cryptographic operators. Safety properties are represented as trace properties, which are then evaluated against the transition system's trace or the observed equivalence of two transition systems.

In the proposed protocol, data provider, data demander and transaction platform communicate with each other and are modeled as the roles Supplier, Purchaser and Server. We begin by defining the cryptographic primitives utilized in the protocol, representing messages as terms constructed from functions that fulfill property descriptions. Next, we establish multiset rewriting rules to model the protocol's behavior. Finally, we articulate the properties to be proven (lemmas) to outline the necessary security attributes of the protocol. These declarations are essential to the protocol design; without them, the Tamarin tool would lack the information needed for verification.

The following security properties (See Appendix for the code.) are defined for the proposed protocol:

(1) Protocol executability

```
lemma execute:
exists-trace
"   Ex A B x #j #i. Suc1(A,x)@j & Suc2(B,x)@i"
```

Figure 3. Formal analysis program of protocol executability.

The protocol executability lemma ensures that all parties can successfully complete their respective steps in the protocol. The variables and actions/facts used in this lemma are as follows:

- **Suc1(A, x)**: This predicate indicates that entity A has successfully completed a protocol step related to data x. For example, $Suc1(\text{Sup}, \text{Data})$ means that the supplier Sup has successfully uploaded the data Data. This fact is generated in the **Sup_ac_Verif_to_Pur** rule, where the supplier completes the data upload process.
- **Suc2(B, x)**: This predicate indicates that entity B has successfully completed a protocol step related to data x. For example, $Suc2(\text{Pur}, \text{Data})$ means that the purchaser Pur has successfully retrieved the data Data. This fact is generated in the **CC9** rule, where the purchaser completes the data retrieval process.
- **#i, #j**: These are time points in the protocol execution trace. They represent the order in which events occur.

The results of formal analysis (Figure 4) show that the executable path can be found during the execution of the protocol interaction, and the proposed protocol is proved to be executable, which indicates that the design of the protocol matches the current technical environment and capabilities. It can run smoothly on the existing software and hardware platforms, whether it is the encryption operation of the data provider, the retrieval request of the data demander, or the storage and management service of the transaction platform, it can be realized by the existing technical means. This means that each part of the protocol is clear and easy to operate. The data provider can successfully complete the data encryption and upload in accordance with the prescribed steps, and there will be no errors or omisations due to the complex operation. The data demander can accurately submit the search request and decrypt and use the data correctly after obtaining authorization. The transaction platform can also effectively manage data storage, provide retrieval services and carry out security supervision, and the whole process of data transaction is smooth and unimpeded.

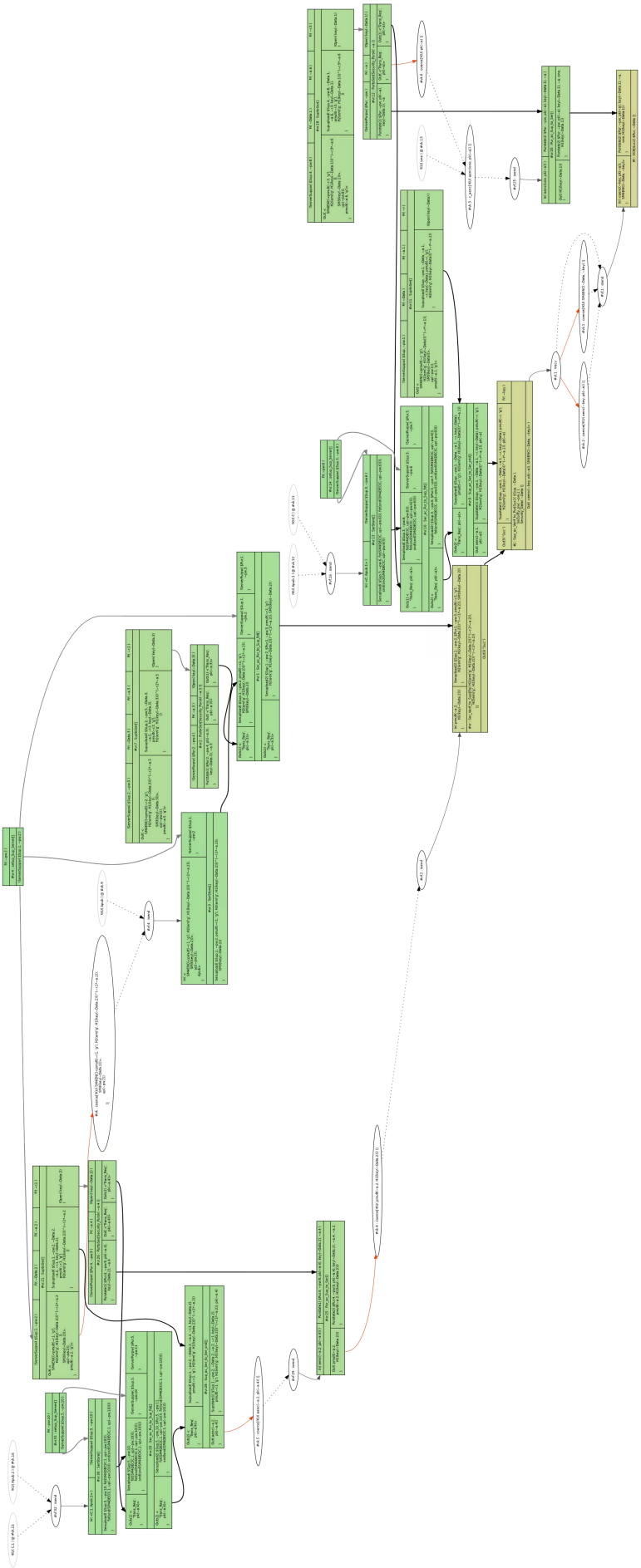

Figure 4. Formal analysis results of protocol executability.

(2) Data Security

```
lemma Security_Data:
"   All  x #i. Security_Data(x)@i  ⟹ not (Ex #j. K(x)@j)"
```

Figure 5. Formal analysis program of data security.

The data security lemma ensures that the data remains confidential throughout the transaction process. The variables and actions/facts used in this lemma are as follows:

- **Security_Data(x)**: This predicate indicates that data x is secure and has not been compromised. For example, $Security\_Data(\mathrm{Data})$ means that the data Data remains confidential and has not been leaked. This fact is generated in the **Sup_ac_Verif_to_Pur** rule, where the data is encrypted and securely transmitted.
- **K(x)**: This represents the knowledge of an adversary about data x. If $K(x)$ is true, it means that the adversary has obtained data x. This fact is generated in the **reveal** rule, where the adversary attempts to reveal sensitive information.

The formal analysis results (as shown in Figure 6) show that there is no leakage event in the whole process of data interaction, which proves that the proposed protocol satisfies data security, indicating that the design of the data transaction protocol is scientific, reasonable, rigorous and effective. The analysis results show that the encryption mechanism of the protocol is strong enough to deal with potential security threats, whether from external hackers or internal misbehavior. At the same time, it also shows that the protocol takes security risks into account in all aspects of data transaction, and takes corresponding preventive measures.

(3) Provider's private key privacy

The provider's private key privacy lemma ensures that the provider's private key remains secure throughout the transaction process. The variables and actions/facts used in this lemma are as follows:

- **Security_pw(x)**: This predicate indicates that the private key x of the provider is secure and has not been compromised. For example, $Security\_pw(\mathrm{pw})$ means that the provider's private key pw remains confidential. This fact is generated in the **setup_Sup_Server** rule, where the provider's private key is securely generated and stored.
- **K(x)**: This represents the knowledge of an adversary about the private key x. If $K(x)$ is true, it means that the adversary has obtained the private key x. This fact is generated in the **reveal** rule, where the adversary attempts to reveal the private key.

The formal analysis results (Figure 8) indicate that the provider's private key remains secure throughout the transaction process. This confirms that the proposed protocol upholds privacy and safeguards the supplier's security during transactions. It shows that the protocol designer has carefully addressed the critical role and sensitivity of the supplier's private key in data transactions, implementing multi-layered and comprehensive security measures to safeguard it.

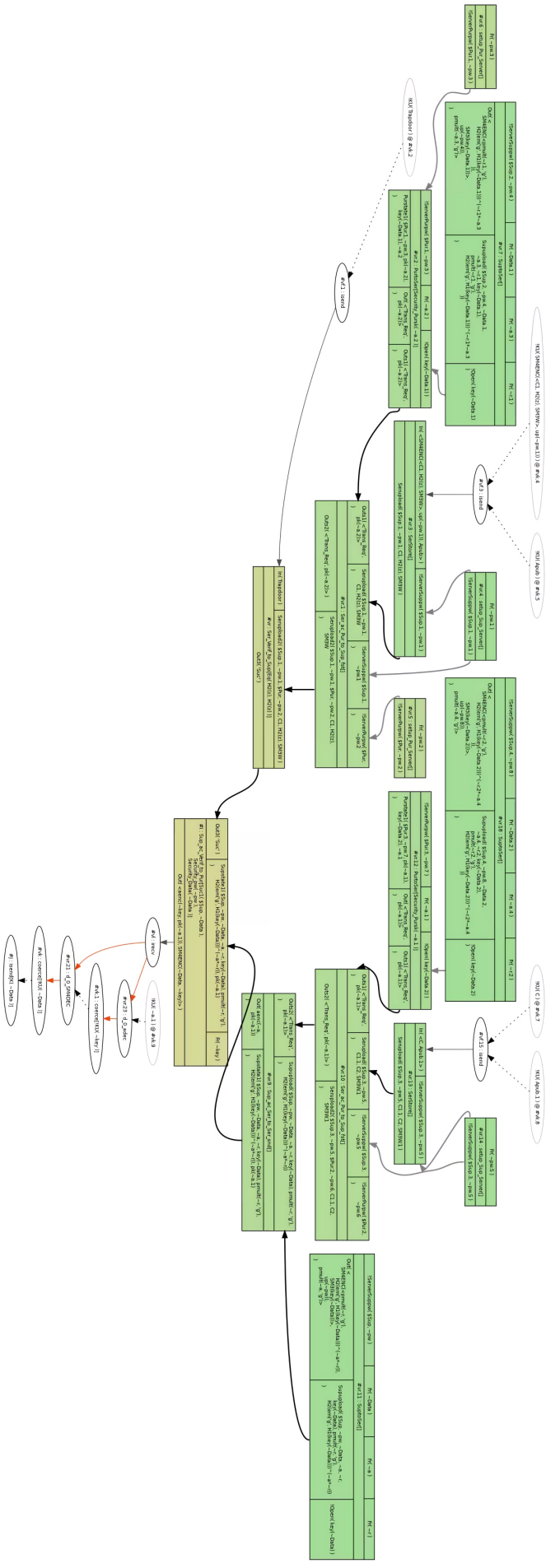

Figure 6. Formal analysis results of data security.

```
lemma Security_pw:
"   All  x #i. Security_pw(x)@i  ⟹ not (Ex #j. K(x)@j)"
```

Figure 7. Formal analysis program of provider's private key privacy.

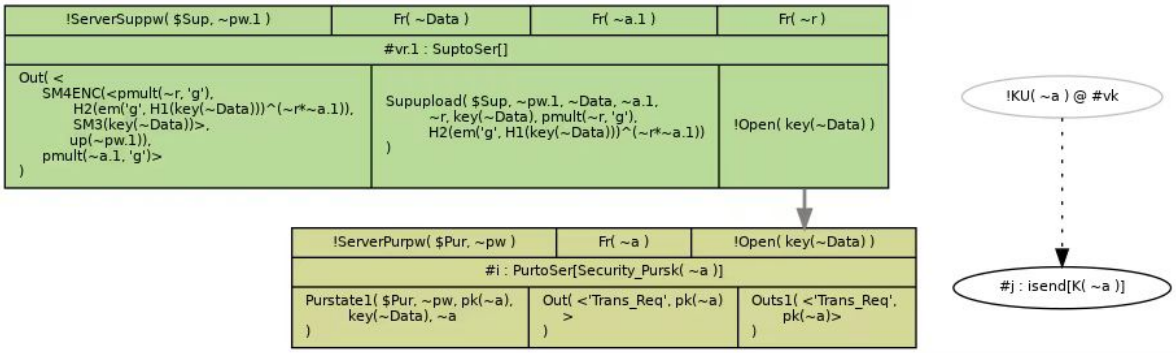

Figure 8. Formal analysis results of provider's private key privacy.

(4) Demander's private key privacy

```
lemma Security_Pursk:
"   All  x #i.  Security_Pursk(x)@i ⟹ not (Ex #j. K(x)@j)"
```

Figure 9. Formal analysis program of demander's private key privacy.

The demander's private key privacy lemma ensures that the demander's private key remains secure throughout the transaction process. The variables and actions/facts used in this lemma are as follows:

- **Security_Pursk(x)**: This predicate indicates that the private key x of the demander is secure and has not been compromised. For example, $Security\_Pursk(\mathrm{sk})$ means that the demander's private key sk remains confidential. This fact is generated in the **setup_Pur_Server** rule, where the demander's private key is securely generated and stored.
- **K(x)**: This represents the knowledge of an adversary about the private key x. If $K(x)$ is true, it means that the adversary has obtained the private key x. This fact is generated in the **reveal** rule, where the adversary attempts to reveal the private key.

The formal analysis results (Figure 10) indicate that the private key of the demander remains secure throughout the entire transaction process. This demonstrates that the proposed protocol effectively safeguards privacy and protects the demander's security during transactions. Furthermore, it highlights that the protocol's designer has thoroughly considered the critical importance and sensitivity of the demander's private key in data transactions. Multi-level and all-sided security measures are adopted to protect the private key of the demander.

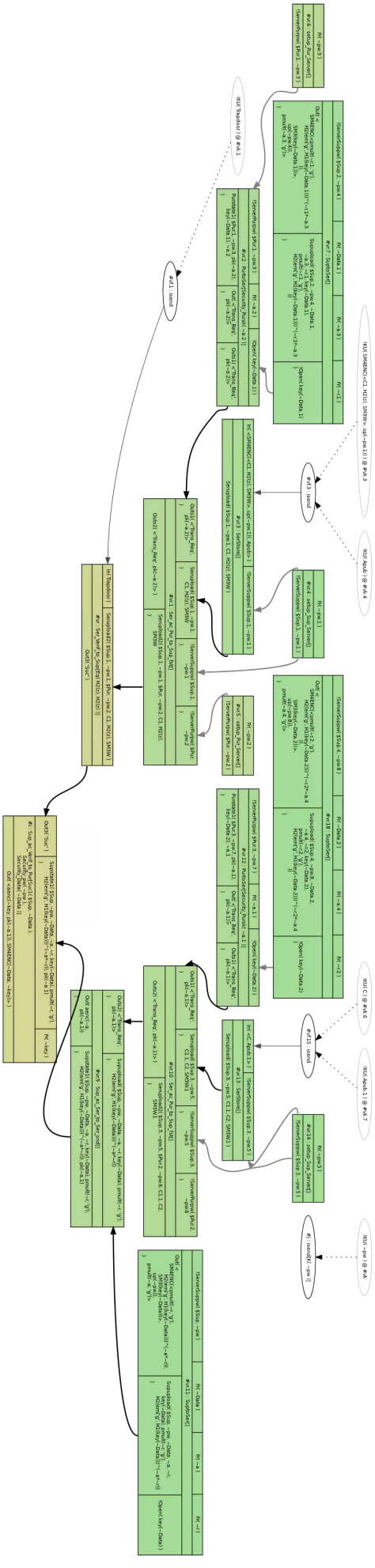

Figure 10. Formal analysis results of demander's private key privacy.

In summary, after rigorous formal analysis, the protocol shows many outstanding features. With its executability, data security and private key privacy, it provides a reliable, efficient and secure solution for the field of data transaction. The protocol demonstrates high compatibility with the current technical landscape, facilitating smooth operations for data providers during encryption and upload, for data demanders in retrieval and usage, and for the transaction platform in storage management and retrieval services. This compatibility establishes a solid foundation for efficient data transactions. Throughout the entire protocol, data security is comprehensively ensured. From the generation of data to its final use, robust encryption, secure transmission channels, and stringent storage management and access controls work together to create a resilient security framework, preventing unauthorized access, tampering, or destruction of data during transactions. In terms of private key privacy, both the provider's private key and the demander's private key are strictly protected in the whole data transaction process without leakage risk. This makes the ownership and use rights of data accurately controlled, and data providers and demanders can interact with each other in a secure environment, which greatly enhances the trust of all parties in data transaction.

5.2. Informal security analysis

We will illustrate the security of the proposed protocol against different attack vectors through clear and intuitive reasoning. It's important to note that we use an informal, heuristic approach for this security analysis. This approach seeks to showcase the protocol's robustness against additional attacks not addressed in the previous section.

Credential theft and padding: In conventional centralized systems, attackers frequently steal credentials to obtain unauthorized access. Because a user's identifier and credentials are kept in a single authentication repository, an attacker may exploit access to the authentication server to steal user credentials. While in our proposed protocol, decentralized identity and credential management is achieved by using the concept of key localization. Each party creates and manages its own secret key without relying on the market to maintain it. Users have full control over their own keys, which gives them greater autonomy. Although in the proposed protocol, credentials still have the possibility of being stolen or leaked, it is less attractive to attackers because there is no centralized repository, and sometimes it is not even worth the attacker's great effort to steal.

Non-repudiation: When a party uses its private key to sign a statement, it cannot later deny the action, thus guaranteeing non-repudiation. This process relies on digital signatures and verification, which are founded on asymmetric cryptography. In order to prevent this attack, before the data demander confirms the receipt of the goods and the provider obtains the transaction income, the data demander is required to digitally sign the information confirming the receipt of the goods with the private key, and the platform is verified to confirm the identity, because only the data demander knows its private key, which ensures non-repudiation.

Replay attack: An attacker might intercept and replay messages exchanged between a supplier and a demander, potentially gaining access to sensitive information such as payment details or confidential data. To mitigate this risk, the data provider should verify the freshness of the public/private key pair and other key information before initiating a transaction with the demander. Moreover, it is crucial to update the key pair before every transaction and implement strategies to mitigate the risk of inference attacks.

Privilege escalation and abuse: To mitigate the risks posed by malicious users, adopting a least privilege access model is essential. This approach ensures that individuals are assigned only the permissions necessary for their specific tasks, confined to defined timeframes and conditions. Additionally, usage-based access control continuously evaluates the relevance of access permissions for various resources. Implementing targeted authorization policies that are bound by specific conditions and obligations can greatly reduce the potential entry points for attackers.

Denial of service attack: In a traditional centralized system, a DoS attack can cripple the entire platform by overwhelming the central server with excessive requests, rendering it unable to process legitimate transactions. However, our protocol leverages the decentralized nature of blockchain technology to mitigate this risk. Since the transaction platform is supported by a distributed network of nodes, even if one or several nodes are targeted by a DoS attack, the remaining nodes can continue to operate and process transactions. This ensures the availability and reliability of the platform. Additionally, the use of blockchain consensus mechanisms (e.g., Proof of Work or Proof of Stake) ensures that malicious actors cannot easily disrupt the network by flooding it with invalid transactions. Each transaction must be validated by multiple nodes, making it computationally infeasible for an attacker to overwhelm the system. Furthermore, the protocol incorporates rate-limiting mechanisms at the application layer to prevent excessive requests from a single user, further reducing the risk of DoS attacks.

Data tampering attack: Data integrity is a critical concern in data transaction platforms, as any unauthorized modification of data can lead to severe consequences, such as financial losses or breaches of trust. Our protocol addresses this issue by leveraging the immutability of blockchain technology. Once data is uploaded to the blockchain, it cannot be altered without changing the entire chain, which is computationally infeasible due to the cryptographic hash linking between blocks. Specifically, the protocol uses the SM3 hash algorithm to generate a unique hash value for each piece of data. This hash value is stored on the blockchain, and any attempt to tamper with the data will result in a mismatch between the stored hash and the recalculated hash, immediately exposing the tampering attempt. Moreover, the protocol employs digital signatures to ensure the authenticity of data transactions. Each transaction is signed by the data provider using their private key, and the signature is verified by the platform using the corresponding public key. This ensures that only authorized parties can initiate transactions, and any unauthorized modifications to the data will be detected and rejected.

6. Performance evaluation

To validate the efficiency and scalability of the proposed SDTP protocol, we conducted extensive experiments comparing its performance with a traditional data transaction scheme. This section presents the experimental setup, performance metrics, results, and analysis.

6.1. Experimental setup

The experiments were performed on a workstation running Python 3.9, utilizing libraries such as NumPy and Matplotlib for data generation and visualization. We generated a dataset consisting of 1000 data points, each associated with 1 to 10 keywords randomly selected from a pool of 100 keywords. The comparison scheme was implemented to simulate a traditional data transaction protocol without the optimizations introduced in SDTP. Both schemes were evaluated based on four key metrics: query time, storage cost, communication cost, and computation cost.

6.2. Performance metrics and results

The performance of SDTP was evaluated against the traditional scheme, and the results are summarized as follows:

1. Query time: SDTP exhibited a slightly higher query time (0.0006 s) compared to the traditional scheme (0.0005 s) (Figure 11). This minor difference is attributed to the additional security measures implemented in SDTP, such as blockchain-based transaction recording and searchable encryption.

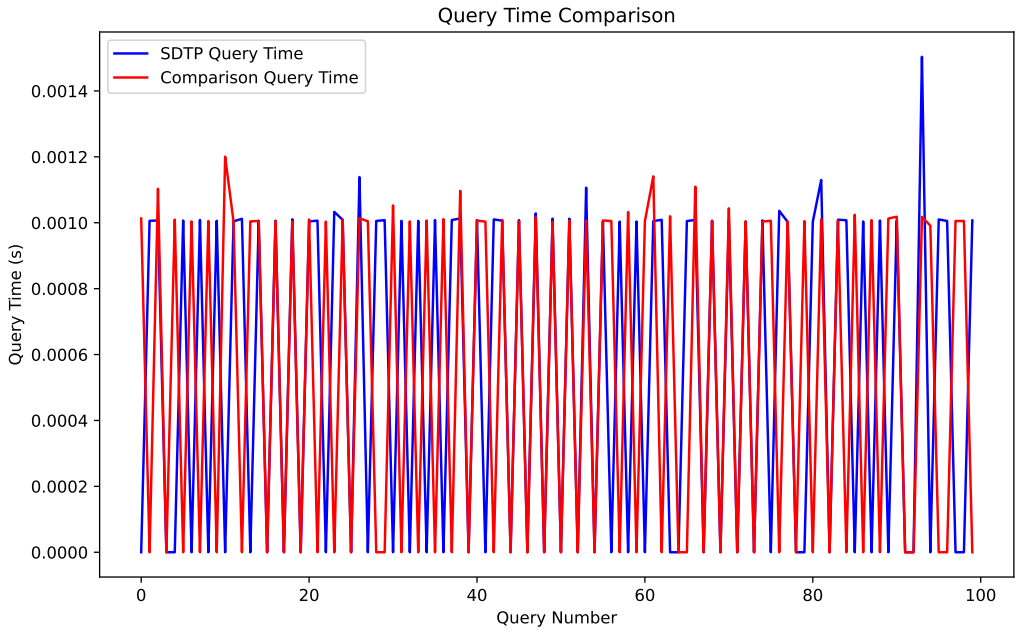

Figure 11. Query time comparison.

2. Storage cost: SDTP significantly reduced storage overhead, achieving an average cost of 1000.00 units compared to 1200.00 units for the traditional scheme (Figure 12). This 16.7% reduction is due to SDTP's efficient encryption and storage strategies.

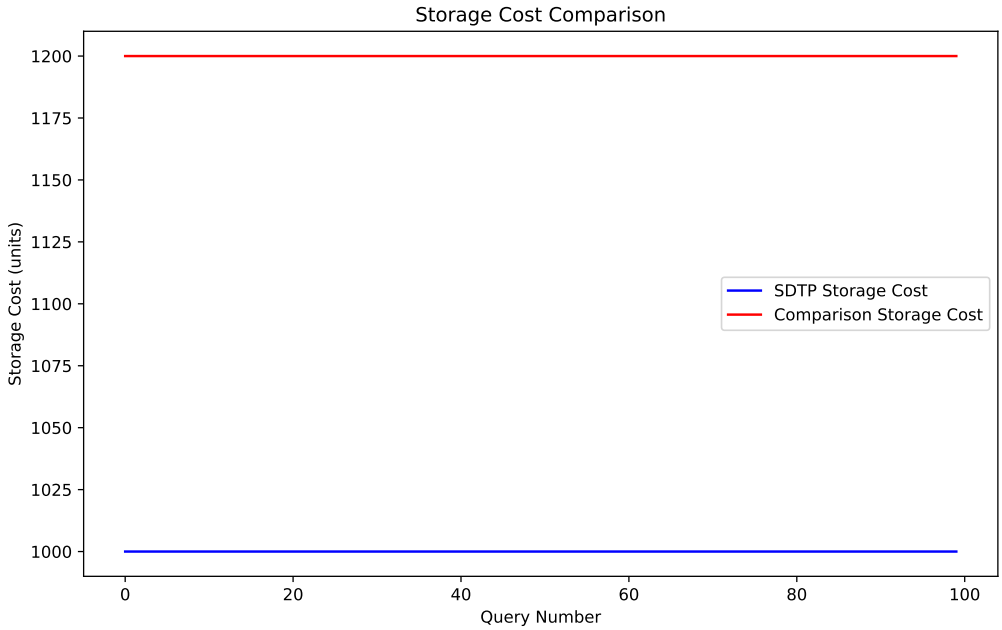

Figure 12. Storage cost comparison.

3. Communication cost: SDTP demonstrated a 32.6% reduction in communication cost, with an average of 0.29 units compared to 0.43 units for the traditional scheme (Figure 13). This improvement is achieved by minimizing unnecessary data transmissions and optimizing the communication protocol.

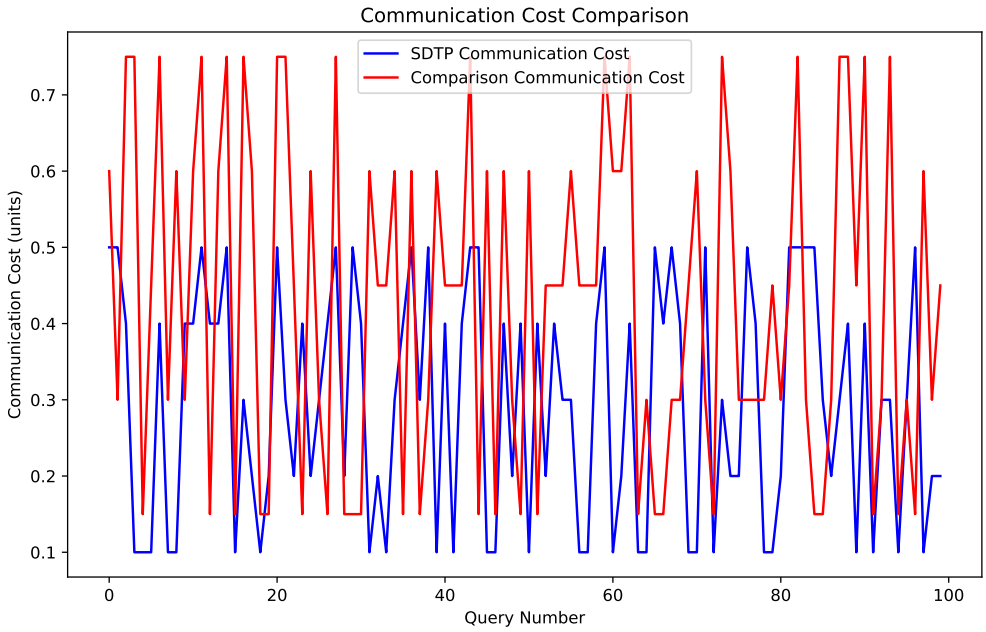

Figure 13. Communication cost comparison.

4. Computation cost: SDTP outperformed the traditional scheme in computation efficiency, reducing the average cost to 1.4833 units compared to 2.1982 units (Figure 14). This 32.5% reduction is a result of SDTP's streamlined computation process and elimination of redundant calculations.

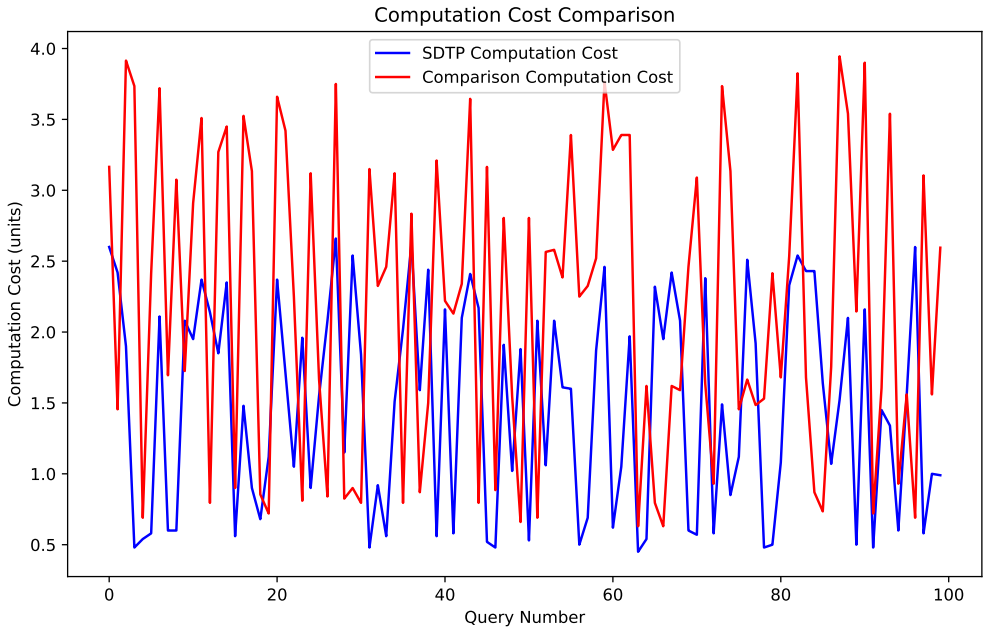

Figure 14. Computation cost comparison.

6.3. Analysis and discussion

The experimental results highlight the superior performance of SDTP in storage, communication, and computation costs. While the query time of SDTP is slightly higher, this trade-off is

justified by the enhanced security and transparency provided by blockchain technology.

The slight increase in query time is a reasonable trade-off for the additional security features, such as tamper-proof transaction recording and privacy-preserving search capabilities. These features are essential for building trust in data transaction platforms, especially in public cloud environments.

7. Conclusion

With the continuous growth of data volume, more and more valuable data are eager to be widely used, and the demand for data transaction is also constantly expanding. Although the research on cryptographic techniques that can be used to protect the security of data transaction has accumulated a large number of rich achievements in various directions, the comprehensive application research on the whole process of data transaction is less. Integrating protection technologies for data storage, trading, sharing, computing, and analysis presents significant challenges. Achieving synergy among diverse research outcomes to address security issues in data transactions is complex. A mere mechanical combination of solutions can lead to various problems, including security vulnerabilities, inefficiencies, and compatibility issues, such as redundant encryption and frequent communication demands. In this paper, we propose a blockchain-supported data transaction protocol based on searchable encryption. The protocol and system rely on searchable public key encryption and blockchain technology, taking into account privacy and security, ensuring that the platform cannot steal data, the provider cannot tamper with data, and the demander cannot deny the transaction fact. Finally, we conduct a comprehensive formal and informal security analysis of the proposed protocol. The findings indicate that our protocol demonstrates several outstanding features, offering a reliable, efficient, and secure solution for data transactions. Its strengths lie in executability, data security, and the protection of private key privacy.

Acknowledgments

This work is supported by The National Key Research and Development Program of China No. 2023YFB2705000 and National Engineering Laboratory for Big Data Distribution and Exchange Technologies. The authors would like to thank the reviewers for their valuable comments that helped to improve this manuscript.

Conflicts of Interests

The authors declared that they have no conflicts of interests.

Authors contribution

Conceptualization, L.S.Q., T.Q.F.; Investigation, H.X.Y., F.H.P., W.Y.J., C.Y.D., X.H.Y.; Writing - original draft, L.S.Q., F.H.P., H.X.Y., X.H.L.; Writing - review & editing, H.X.Y., C.Y.D., X.H.Y., W.Y.J; Project administration, H.X.Y., F.H.P., C.Y.D., T.Q.F.; Supervision, L.S.Q., X.H.L; All authors have read and agreed to the published version of the manuscript.

References

[1] Si Y, Huang Y, Wang T, Su J. End-to-End Data Commodity Delivery Based on Metadata. In *Proceedings of the 4th International Conference on Computer Science and Application Engineering*, Sanya, China, October 20–22, 2020, pp. 1–5.

[2] Bao Y, Qiu W, Cheng X. Secure and Lightweight Fine-Grained Searchable Data Sharing for IoT-Oriented and Cloud-Assisted Smart Healthcare System. *IEEE Internet Things J.* 2021, 9(4):2513–2526.

[3] Boneh D, Di Crescenzo G, Ostrovsky R, Persiano G. Public Key Encryption with Keyword Search. In *Advances in Cryptology - EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques*, Interlaken, Switzerland, May 02–06, 2004, pp. 506–522.
[4] Shen J, Zeng P, Choo KKR. Multicopy and Multiserver Provable Data Possession for Cloud-Based IoT. *IEEE Internet of Things J.* 2021, 9(14):12300–12310.
[5] Chen C, Wu J, Lin H, Chen W, Zheng Z. A Secure and Efficient Blockchain-Based Data Trading Approach for Internet of Vehicles. *IEEE Trans. Veh. Technol.* 2019, 68(9):9110–9121.
[6] Dhakad N, Kar J. EPPDP: An Efficient Privacy-Preserving Data Possession With Provable Security in Cloud Storage. *IEEE Syst. J.* 2022, 16(4):6658–6668.
[7] Yang Y, Chen Y, Chen F, Chen J. An Efficient Identity-Based Provable Data Possession Protocol With Compressed Cloud Storage. *IEEE Trans. Inf. Forensics Secur.* 2022, 17:1359–1371.
[8] Song DX, Wagner D, Perrig A. Practical techniques for searches on encrypted data. In *Proceeding 2000 IEEE symposium on security and privacy. S&P 2000*, California, USA, May 14–17, 2000, pp. 44–55.
[9] Xu P, Gao X, Wang W, Susilo W, Wu Q, *et al.* Dynamic Searchable Public-Key Ciphertexts with Fast Performance and Practical Security. *Cryptology ePrint Archive* 2017, p. 741.
[10] Boyen X, Waters B. Anonymous Hierarchical Identity-Based Encryption (Without Random Oracles). In *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference*, California, USA, August 20–24, 2006, pp. 290–307.
[11] Rhee HS, Park JH, Susilo W, Lee DH. Trapdoor security in a searchable public-key encryption scheme with a designated tester. *J. Syst. Softw.* 2010, 83(5):763–771.
[12] Xu P, Wu Q, Wang W, Susilo W, Domingo-Ferrer J, *et al.* Generating Searchable Public-Key Ciphertexts With Hidden Structures for Fast Keyword Search. *IEEE Trans. Inf. Forensics Secur.* 2015, 10(9):1993–2006.
[13] Lu J, Shen J, Vijayakumar P, Gupta BB. Blockchain-Based Secure Data Storage Protocol for Sensors in the Industrial Internet of Things. *IEEE Trans. Ind. Inform.* 2021, 18(8):5422–5431.
[14] Dixit A, Zarpelao BB, Smith-Creasey M, Rajarajan M. A privacy-aware authentication and usage-controlled access protocol for IIoT decentralized data marketplace. *Comput. Secur.* 2024, 146:104050.
[15] Ma R, Zhang L, Wu Q, Mu Y, Rezaeibagha F. BE-TRDSS: Blockchain-Enabled Secure and Efficient Traceable-Revocable Data-Sharing Scheme in Industrial Internet of Things. *IEEE Trans. Ind. Inform.* 2023, 19(11):10821–10830.
[16] Zhao Y, Yu Y, Li Y, Han G, Du X. Machine learning based privacy-preserving fair data trading in big data market. *Inf. Sci.* 2019, 478:449–460.
[17] Wang D, Gao J, Yu H, Li X. A Novel Digital Rights Management in P2P Networks Based on Bitcoin System. In *Frontiers in Cyber Security: First International Conference, FCS 2018*, Chengdu, China, November 05–07, 2018, pp. 227–240.
[18] Dai W, Dai C, Choo KKR, Cui C, Zou D, *et al.* SDTE: A Secure Blockchain-Based Data Trading Ecosystem. *IEEE Trans. Inf. Forensics Secur.* 2019, 15:725–737.
[19] Jiang S, Cao J, Wu H, Chen K, Liu X. Privacy-preserving and efficient data sharing for blockchain-based intelligent transportation systems. *Inf. Sci.* 2023, 635:72–85.
[20] Hu S, Cai C, Wang Q, Wang C, Luo X, *et al.* Searching an Encrypted Cloud Meets Blockchain: A Decentralized, Reliable and Fair Realization. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*, HI, USA, April 16–19, 2018, pp. 792–800.
[21] Chen L, Lee WK, Chang CC, Choo KKR, Zhang N. Blockchain based searchable

encryption for electronic health record sharing. *Future Gener. Comput. Syst.* 2019, 95:420–429.

[22] Xu J, Ying C, Tan S, Sun Z, Wang P, *et al.* An Attribute-Based Searchable Encryption Scheme Supporting Trapdoor Updating. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech)*, Athens, Greece, August 12–15, 2018, pp. 7–14.

[23] Han F, Qin J, Hu J. Secure searches in the cloud: A survey. *Future Gener. Comput. Syst.* 2016, 62:66–75.

[24] Jia H, Aldeen MS, Zhao C, Jing S, Chen Z. Flexible privacy-preserving machine learning: When searchable encryption meets homomorphic encryption. *Int. J. Intell. Syst.* 2022, 37(11):9173–9191.

[25] Abbasi A, Prieto J, Shahraki A, Corchado JM. Industrial data monetization: A blockchain-based industrial IoT data trading system. *Internet Things* 2023, 24:100959.

Appendix

```
theory PSE begin


builtins: bilinear-pairing,diffie-hellman,asymmetric-encryption
functions: key/1, SM3/1,SM4ENC/2,SM4DEC/2,up/1,H1/1,H2/1
equations:SM4DEC(SM4ENC(M,K),K)=M

rule setup_Sup_Server:
[ Fr(~pw)]
--[  ]->
[ !ServerSuppw($Sup,~pw) ]

rule setup_Pur_Server:
[ Fr(~pw)]
--[  ]->
[ !ServerPurpw($Pur,~pw) ]
/*rule reveal:
[ !SharedKeyWithServer(X, k) ]
--[ Reveal(X,k),Init_once('1') ]->
[ Out(k), Reveal(X,k)]
*/

/* 1 */
rule SuptoSer:
let
h=pmult(~a,'g')
SM4key=up(pw)
W=key(~Data)
C1=pmult(~r,'g')
t=em(H1(W),pmult(~r,h))
C2=H2(t)
C=SM4ENC(<C1,<C2,SM3(W)>>,SM4key)
in
[ !ServerSuppw($Sup,pw),Fr(~Data),Fr(~a),Fr(~r)]
--[  ]->
[ Out(<C,h>) ,Supupload($Sup,pw,~Data,~a,~r,W,C1,C2) , !Open(W)]

/* 2 */
rule SerStore:
let
SM4key=up(pw)
M=SM4DEC(C,SM4key)
C1=fst(M)
C2=fst(snd(M))
SM3W=snd(snd(M))
in
[ In(<C,Apub>),!ServerSuppw($Sup,pw)]
--[  ]->
```

```
[ Serupload($Sup,pw,C1,C2,SM3W)]

/* 3 */
rule PurtoSer:
let
h=pk(~a)
in
[ !ServerPurpw($Pur,pw),Fr(~a),!Open(W)]
--[  Security_Pursk(~a)]->
[ Purstate1($Pur,pw,h,W,~a),Out(<'Trans_Req',h>),Outs1(<'Trans_Req',h>)]

/* 4 */
rule Ser_ac_Pur_to_Sup_fst:
[ Outs1(<'Trans_Req',Apubs>),Serupload($Sup,pw,C1,C2,SM3W),!ServerSuppw($
--[  ]->
[ Outs2(<'Trans_Req',Apubs>),Serupload2($Sup,pw,$Pur,pws,C1,C2,SM3W)]

/* 5 */
rule Sup_ac_Ser_to_Ser_snd:
let
C=aenc(a,Apubs)
in
[ Outs2(<'Trans_Req',Apubs>),Supupload($Sup,pw,Data,a,r,W,C1,C2)]
--[  ]->
[ Out(C),Supstate1($Sup,pw,Data,a,r,W,C1,C2,Apubs)]

/* 6 */
rule Pur_ac_Sup_to_Ser:
let
a=adec(C,sk)
Trapdoor=pmult(a,H1(W))
in
[ In(C),Purstate1($Pur,pw,h,W,sk)]
--[  ]->
[ Out(Trapdoor),Purstate2($Pur,pw,h,W,sk,a,Trapdoor)]

/* 7 */
rule Ser_Verif_to_Sup:
let
A=H2(em(Trapdoor,C1))
in
[ In(Trapdoor),Serupload2($Sup,pw,$Pur,pws,C1,C2,SM3W)]
--[ Eq(A,C2)]->
[ Out3('Suc')]

/* 8 */
rule Sup_ac_Verif_to_Pur:
let
Csk=aenc(~key,Apubs)
C=SM4ENC(Data,~key)
in
```

```
[ Out3('Suc'),Supstate1($Sup,pw,Data,a,r,W,C1,C2,Apubs),Fr(~key)]
--[ Suc1($Sup,Data),Security_pw(pw),Security_Data(Data)]->
[ Out(<Csk,C>)]

/* 9 */
rule CC9:
let
key=adec(Csk,sk)
Data=SM4DEC(C,key)
in
[ In(<Csk,C>),Purstate2($Pur,pw,h,W,sk,a,Trapdoor)]
--[ Suc2($Pur,Data)]->
[ ]


restriction Verif:
"
All x y #i. Eq(x,y) @ i ==> x = y
"

lemma execute:
exists-trace
"
Ex A B x #j #i. Suc1(A,x)@j & Suc2(B,x)@i
"

lemma Security_Data:
"
All  x #i. Security_Data(x)@i  ==> not (Ex #j. K(x)@j)
"

lemma Security_pw:
"
All  x #i. Security_pw(x)@i  ==> not (Ex #j. K(x)@j)
"

lemma Security_Pursk:
"
All  x #i.  Security_Pursk(x)@i ==> not (Ex #j. K(x)@j)
"

end
```