National College of Ireland

Asymmetric Encryption and Blockchain based Multi-keyword Encrypted Search

MSc Research Project

Akash Kulkarni
Student ID: x21138419

School of Computing
National College of Ireland

Supervisor: Shivani Jaiswal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	AKASH KULKARNI		
Student ID:	X21138419		
Programme:	MSc Cloud Computing	**Year:**	2022-2023
Module:	Research Project		
Supervisor:	Shivani Jaiswal		
Submission Due Date:	15th December 2022		
Project Title:	Asymmetric Encryption and Blockchain based Multi-keyword encrypted Search		
Word Count:	7338		
Page Count:	18		

Signature: Akash Kulkarni

Date: 15th December 2022

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Asymmetric Encryption and Blockchain based Multi-keyword Encrypted Search

Akash Kulkarni
x21138419

Abstract
Blockchain technology has recently gained significant attention as a potential solution for securing data in cloud computing environments. This is due to its ability to provide a decentralized and immutable ledger of transactions, which can be used to store and manage sensitive information in a secure and transparent manner. In this paper, we propose the use of blockchain-based searchable encryption for improving the security of cloud computing systems. Searchable encryption is a cryptographic technique that allows for searching encrypted data without decrypting it. This allows for more secure storage and retrieval of sensitive information, as the data remains encrypted at all times, even when being searched. By combining searchable encryption with blockchain technology, we can provide a secure and efficient method for storing and searching sensitive data in the cloud. Our proposed system provides several benefits for cloud security. First, it allows for secure storage and retrieval of sensitive data in the cloud, as the data remains encrypted at all times. Second, it allows for efficient searching of encrypted data, making it practical for use in a variety of applications. Third, it provides a decentralized and immutable ledger of transactions, which can be used to provide transparency and accountability for the storage and management of sensitive data. Overall, the proposed system offers a promising approach for improving the security of cloud computing systems using blockchain-based searchable encryption.

***Keywords**: Cloud storage, security, blockchain, smart contract, asymmetric encryption*

1. Introduction

Cloud storage has become one of the most essential parts of technology world. Hardly very few technology giants are capable of storing data in servers which are in their control. Storing data in cloud helps organizations to reduce the costs on storage, organizing, scaling and maintenance. So, security aspect of cloud storage becomes extremely important as remote servers are untrusted parties. To maintain privacy levels encrypting data and then storing in cloud is a solution. The issue pertaining to this is data utilization, as retrieving data and then decrypting it on user end is a challenging as well as inefficient task. Using searchable encryption technology keyword search can be performed on the encrypted data without compromising any kind of information to the cloud. Searchable encryption was revealed by {practical paper} which actually allowed to search over the encrypted data in database. Particularly there are two types of Searchable encryption techniques:

- *Symmetric Searchable Encryption*

Symmetric searchable encryption is a cryptographic method that allows for searching encrypted data without decrypting it. This is accomplished by using a trapdoor function, which is a type of mathematical function that is easy to compute in one direction but difficult to compute in the reverse direction. The trapdoor function is used to encrypt the data, and a searchable index is generated that allows for efficient searching of the encrypted data. When a search query is made, the query is encrypted using the same trapdoor function and matched against the searchable index. Any matching records can then be retrieved without revealing the underlying plaintext. The key advantage of symmetric searchable encryption is that it allows for secure storage and retrieval of sensitive information, as the data remains encrypted at all times. This provides a higher level of security than traditional methods of encrypting data, as the data can be searched without the need to decrypt it first. This makes it possible to search encrypted data without exposing it to potential attackers. Additionally, the use of a trapdoor function allows for efficient searching of the encrypted data, making it practical for use in a variety of applications.

- *Asymmetric Searchable Encryption*

Asymmetric searchable encryption, also known as public-key searchable encryption, is a cryptographic method that allows for searching encrypted data without decrypting it. This is accomplished by using a public-key encryption scheme, in which a pair of keys - a public key and a private key - are used to encrypt and decrypt data. The public key is used to encrypt the data, and a searchable index is generated that allows for efficient searching of the encrypted data. When a search query is made, the query is encrypted using the public key and matched against the searchable index. Any matching records can then be retrieved without revealing the underlying plaintext. The key advantage of asymmetric searchable encryption is that it allows for secure storage and retrieval of sensitive information, as the data remains encrypted at all times. This provides a higher level of security than traditional methods of encrypting data, as the data can be searched without the need to decrypt it first. This makes it possible to search encrypted data without exposing it to potential attackers. Additionally, the use of a public-key encryption scheme allows for efficient searching of the encrypted data, making it practical for use in a variety of applications.

General Searchable encryption algorithms have two steps. One where Data owner (DO) encrypts the data, creates its index along with encryption of keywords and stores that information in cloud database. Second step is where a Data user (DU) generates a trapdoor which consists of its credentials and keywords and submits it to the cloud. Upon creating and searching keyword index cloud responds with particular dataset to the DU. Most of the SSE and PEKS schemes available are based on trusting centralized server to hold the value of encrypted file index and keywords. Most of the cloud service providers follow honest-but-curious model, which means that a cloud server might try to read what kind of data user is trying to access by looking into information in trapdoor such as keywords and credentials. To avoid such issues, data such as trapdoor query and user credentials needs to be away from cloud server but available on request. Blockchain plays a very important part in this, if the indexed keywords and trapdoor queries are stored in blockchain and upon request delivered to cloud server, server won't be able to sniff and read the data. This solves the hones-but-curious part of the problem. Blockchain works on different consensus algorithms such as proof-of-work (bitcoin), proof-of-stake (Ethereum/polygon), hashgraph consensus(hedera) and so on. Every mechanism has its own benefit over others. For searchable encryption throughput is very vital, hence hashgraph consensus shall be used. Blockchain also enables fair payment mechanism to the data owners because of its features of smart contracts, which enables to create virtual payment systems using tokens.

Honest-but-curious servers are not the only problem with centralized systems. Its very important to ensure the credibility of the trapdoor request from data user, whether the user who has requested the data has the actual authorization or the trapdoor information is not getting leaked after the search. So here layers of verification are required such as

By gathering all the sufficient information, this paper proposes a scheme which using combination of SSE and PEKS along with blockchain to achieve secure searchable encryption mechanism having unique properties such as multi-keyword search, forward and backward privacy and fair payment. The scheme proposed is **A**symmetric **E**ncryption and **B**lockchain-based **M**ulti-keyword **E**ncrypted **S**earch (**AEBMES**).

The scheme is built along the following research question: How to improve security and enhance multi-keyword search on encrypted data in cloud by using blockchain and asymmetric key encryption mechanism ?

AEBMES scheme is summarized as follows:

- Scheme uses combination of symmetric encryption and public-key encryption in a way where symmetric key is used to encrypt the data while public-key encryption is used to encrypt the symmetric key, so that sharing the key will not be a issue.
- Blockchain will be used to store data related to encrypted file indexes and keywords along with trapdoor information from data owner
- Smart contracts will be used to store above information, along with there will be a native token for this scheme which will be used to pay as reward for data owner for providing correct information required to data user.
- Scheme includes layers of verification process to ensure there are no data leaks, not allowing access to unauthorized user even if trapdoor is leaked, verifying if cloud provider has delivered the correct data required using document notarization.
- Multiple updates on files will ensure forward and backward privacy of the data. Scheme will include multi-keyword search rather than single word to save computing time and use of resources.

The structure of document is as follows: Section 2 consists of prior research work on similar topic which is followed by section 3 called as preliminaries which will explain of some terms and concepts used in building the scheme. Section 4 will include Architecture and its description along with introduction to various algorithms used in this scheme with the security model. Section 5 will have result and implementation phase along with comparative study regarding complexity, gas-cost and search time, then followed by conclusion.

2. Literature Survey

AEBMES scheme focuses on two things, searchable encryption and blockchain integration in searchable encryption. Thus, the related work completed on above topics is discussed below, which includes their focus and some shortcomings which AEBMES will try to overcome.

Song et al, first came up search based mechanism on encrypted data in cloud based environment. (Dawn Xiaoding Song, Wagner and Perrig, 2000). The main focus of that research was to have a optimal fast keyword search over encrypted data. The best case scenario is for the applications to use schemes that give the opponent no information at all. Most of the time, these schemes can be made with oblivious RAM, homomorphic encryption, or multiple computation. However, they usually have high bandwidth costs and a lot of extra work to do. So, a practical SE scheme gives away more or less some information to the opponent as a trade-off to make things work better. Unfortunately, the leakage of information has been used in different ways to compromise the security of SE schemes.(Cash *et al.*, no date). A deadly adaptive attack was proposed by Zhang which can recover the term from search trapdoors by adding as few as 10 additional files(Z. Li *et al.*, 2021). With the advent of the adaptive leakage-exploiting attack, forward privacy has become a focal point of security research.

The forward-privacy-preserving techniques rely on symmetric-key cryptography for the most part. Common among them is a lack of flexibility in crucial distribution and management. Public-key encryption with keyword search was proposed as the initial searchable public-key encryption technique by Boneh et al., which sidesteps the need for communication between the data owner and the data consumer. However, SPE typically has two major drawbacks: (1) it requires a huge number of public-key operations with the ever-increasing dataset size, and (2) it is vulnerable to the inside keyword guessing attack (IKGA) if the smaller space of keywords than that of keys(Boneh *et al.*, 2004). In addition, the storage server is assumed to be an honest but inquisitive entity in most extant searchable encryption methods, which always follows the specified protocol and never tries to depart from it. However, in practice, the storage server might only offer partially complete or incorrect results for a given benefit, or it might even throw execution problems. To ensure that the storage server has properly completed the search process, Zheng et al. devised a verified attribute-based keyword search technique(Zheng, Xu and Ateniese, 2014). Several techniques that are similar to those proposed by Jarecki et al. presume that the server is honest but curious, meaning that it will not diverge from the present protocol, but may act maliciously, i.e., return erroneous results, in order to save money or in the event of technical difficulties. This necessitates keeping an eye on the server to detect any suspicious activity and responding appropriately(Jarecki *et al.*, 2013).

This paper talks about different parts of searchable encryption, such as searching for multiple keywords, changing the values of keys, and using smart contracts to access and store data about trapdoors. The plan is called BPKEMS. Bilinear Pairing, the Blockchain, and the Decisional Diffie-Hellman algorithm are the main parts of this scheme (DDH). Bilinear Pairing has three features: it's linear, doesn't get worse over time, and is super easy for computing purposes. DDH will be used to make pairs of keys for each user. The consensus mechanism, the gas system, which is basically a fee for blockchain transactions, and smart contracts, which are pieces of code used to store data, transfer fees from user to user, etc., are all important parts of blockchain(Chen *et al.*, 2021). Later Li et al, suggested a public-key encryption system based on the blockchain that can be used for keyword searches, allowing for secure, distributed, and easily searchable encryption. The authors have set out to address a widespread problem faced by verifiers today: how to overcome mistrust about cloud-based information. The TrueBit protocol used in this technique is an attempt to circumvent this problem. The key-aggregate approach, which is discussed in this paper, further minimizes the cost of storing key values. The technique allows the data owner to change its private key and revoke access to the data at any time. The proposed method guarantees both many owners and equitable compensation for all parties involved. User only pays for the confirmed data they need and any excess is returned(H. Li *et al.*, 2021). Xu et al, proposes a scheme which provides a blockchain-based verifiable keyword search strategy that may be used to execute quick multi-keyword searches, guarantee fair verifications, and update files in real-time. Uses a blockchain and smart contracts to verify the outcomes, guaranteeing accuracy and fairness, while keeping verification costs down via a bitmap and hash function.(Xu *et al.*, 2022). Chen et al, came up with searchable encryption for vehicular network. This paper presents a new type of public-key encryption that can be searched and has both forward and backward privacy that is based on the blockchain

(BSPEFB). In this method, keywords are looked up in the cloud via smart contracts, which verifies and immutably stores the results. The technique incorporates forward and backward secrecy to further safeguard users' personal information. It is a lightweight system that makes use of only the most fundamental BSPEFB, hence reducing the amount of computation-intensive processes and increasing the search efficiency. The goal of this study is to develop a secure and efficient method of searching encrypted data stored in the cloud, which may then be used in a cloud-assisted vehicular network. Pseudorandom permutation functions, bilinear pairing, and the Diffie-Hellman key exchange are all utilized in the generation of key pairs in this system(Chen *et al.*, 2020). In next study, Yang et al. propose a blockchain-based certificateless searchable public key authenticated encryption (CL-SPKAE). Encrypted indexes are used to store files and guarantee that the cloud server is unaware of the search terms used or the results returned. This protects against servers that are well-intentioned but overly nosy, and it also enables the detection of criminal activity. Managing certificates is simplified with the certificate-less system. The integrity of encrypted indexes may be tracked and any tampering with them can be detected thanks to this scheme's anti-tampering and authenticity check methods. A third-party user cannot guess keywords to employ in an attack on the stored data since the indexes are constructed using Data users' public key and the data owners' private key. Blockchain's built-in anti-tampering protections guarantee that users always get authentic results from their queries. Data users and data owners can conduct lawful financial transactions with the help of smart contracts. In terms of safety, the approach is impenetrable to both keyword guessing attacks and random oracle attacks(Yang *et al.*, 2020).

This study by Chakraborty et al, offers a searchable encryption method based on the blockchain and the bloom filter, which aids in data privacy and keeps tabs on potentially dangerous cloud server activity. Data leaking is prevented by creating a trapdoor for insert and search queries to go through. The method can withstand assaults like search pattern leaking, access pattern leakage, and volume pattern leakage. The effectiveness of the method is tested experimentally on Ethereum test networks(Chakraborty *et al.*, 2022). Wu et al, present the first study of the public verification problem for encrypted numerical search on dynamic data. It develop a system called Succinct Order-Revealing Encryption (SORE) that slices an order condition into many slices, each of which can be used as a keyword search, in order to facilitate numerical search. We also create public verification techniques for these slices, based on multiset hash and the RSA accumulator. To conduct public verification impartially and ensure the data's currency, we use the blockchain as the trusted third party. To ensure insertion privacy, we additionally implement the trapdoor combination to accomplish forward security(Wu *et al.*, 2022).

3. Research Methodology

In this section, aspects of this research will be explained which includes system model, preliminary information related to blockchain used, smart contracts, tokenisation, cryptography used to include encryption algorithms which will be followed by design and security goals of AEBMES scheme.

3.1. System Model

This scheme has following entities:

- Data Owner (DO): DO will encrypt the document with a key which is then encrypted using asymmetric encryption and then along with keyword ciphertext, information is stored in blockchain. Then encrypted file is send to CB which stores it in Database.
- Blockchain Platform (BP): BP is a platform where smart contracts are deployed. DO, DU and CB generally connect with BP to store or retrieve the information. Token contracts and AEBMES scheme contracts are deployed in BP.
- Cloud Database (CD): A database hosted on cloud platform. Only encrypted files are stored in Database. This component won't be connected to blockchain and hence won't have any data related to search query and also won't have access to any keys to decrypt the data stored.
- Cloud Backend (CB): This is where entire logic will take place including key encryption, communication with blockchain. CB takes
- Data User (DU): DU will generate a trapdoor and call smart contract to get the file information and will pass it on to CB.
- Cloud Key Manager (CKM) : It is responsible for creating keypair for different entities such as Owner, user, Cloud Backend. CKM also stores keys in encrypted format provided by the system.

Abbreviation	**Explanation**
PK, SK	Public key (PK), Private key(SK)
K_w	Keyword array
File	File Data
K''_w	Encrypted keyword index
E_k	Symmetric key for AES encryption
fileHash	Sha256 hash of file data
fileCipherText	Cipher text of a file after encryption wih E_k
E'_k	Encrypted key using PK_{DO} and SK_{DU}
S_w	Search keywoed array
Ei	Encrypted Index information
T_h	Trapdoor hash

Table 1 Abbreviation Explanation

3.2. Algorithms

- *Key_Generation => (PK, SK):*

This algorithm generates keypairs for different entities in the system. It outputs public key (*PK*) and private-key (*SK*) for users

- *Encrypt_File(K_w, PK_{du}, File) => (fileHash, K''_w, E'_k):*

This algorithm takes K_w which is array of keywords associated with *File,* PK of Data user (PK_{du}) as input. It first generates E_k (Symmetric) and encrypts the file using E_k which generates *fileCiphertext*, then encrypts E_k using (SK_{DO}, PK_{DU}) and generates E'_k, maps keyword hashes with cipherText and sends the information to blockchain and then uploads generated file ciphertext in cloud database(CD).

- *Generate_trapdoor(S_w, PK_{DO}) =>(Ei, T_h):*

This algorithm is performed by DU, takes S_w as input for array of keywords to be search and PK of Data-owner(DO). This will create a hash of S_w and call smart contract function to get related encrypted index array (*Ei)* and unique trapdoor hash (T_h)

- *Search(T_h, PK_{du}, Ei) =>(fileCipherText):*

This operation is performed by CB where DU passes its PK_{du}, *Ei* and T_h got from previous algorithm. CB will first verify whether the DU has generated the given trapdoor or not and then it will search for the recommended cipherText, if available it will forward it to DU.

- *Verify_Trapdoor(T_h, PK_{DU}) => Boolean:*

This algorithm is performed by CB to verify incoming search request from the DU. It takes T_h aand PK_{DU} as input and calls smart contract function to check in trapdoor for the user exists or not. If Yes, then CB searches for fileCipherText in CD and returns it to DU. If No, the process is terminanted.

- *Decrypt_File(PK_{DO}, fileCipherText, E'_k) => $FILE^d$*

This is executed by DU after receiving file from CB, it will first decrypt E'_k using (SK_{DU}, PK_{DO}), which will generate E_k using which *fileCipherText* will be decrypted by AES and will give $FILE^d$.

- *Verify_Result(fileHash, T_h)=>Boolean:*

This is to verify the result given by CB, first DU will create SHA256 of *FILE,* then with the use of T_h will verify the result in smart contract.

- *Update_document(K_w, old_cipherText, newFile):*

This will be execute by DO and will be same as *Encrypt_file algo,* just this will replace the old_ciphertext with new one, and update the smart contracts along with it.

3.3. Blockchain

A blockchain is a distributed database that maintains a continuously growing list of records, called blocks, secured from tampering and revision. Each block contains a timestamp and a link to the previous block, and is typically managed by a peer-to-peer network that follows a specific protocol for validating new blocks. Blockchains are often used to store and transmit data in a secure and transparent way. Because the data in a blockchain is distributed across a network of computers, it is considered to be highly resistant to tampering and revision. This makes blockchains well-suited for applications that require a high degree of trust and security, such as the storage and transfer of digital currencies.

3.3.1. Hedera Consensus

Hedera Consensus is a distributed ledger technology (DLT) platform that uses a unique consensus mechanism called "Hedera Hashgraph" to achieve high performance, security, and fairness. The platform is designed to support a wide range of applications, including payments, supply chain, digital identity, and more. Hedera Hashgraph is a directed acyclic graph (DAG) data structure that allows for fast and efficient consensus among the nodes in the network. Unlike other DLT platforms that use proof-of-work or proof-of-stake mechanisms, Hedera Hashgraph uses a consensus algorithm called "gossip about gossip" that allows nodes to quickly and securely reach consensus without the need for mining or staking. Hedera Consensus also uses a unique governance model that combines elements of both centralized and decentralized systems(Gross and Thibeau, no date).

Hashgraph is a distributed ledger technology that is based on a different underlying technology called "hashgraph" which is claimed to be faster and more efficient than traditional blockchain technology. This allows hashgraph to process transactions more quickly and with lower fees than other distributed ledger systems. In contrast, PoS is a specific algorithm that is used by some blockchain networks to achieve distributed consensus and validate transactions. In a PoS system, the nodes that validate transactions are chosen based on the amount of stake they hold in the network, rather than the amount of computing power they contribute, as is the case with proof of work (PoW) systems.

3.3.2. Smart Contract and Tokenization

A smart contract is a computer program that automatically executes the terms of a contract when certain conditions are met. It is called a "smart" contract because it can self-execute and self-enforce the terms of the agreement without the need for third-party intervention. Smart contracts have the following properties:

- Automation: The terms of the contract are automatically executed when certain conditions are met, without the need for manual intervention.
- Self-execution: The contract executes on its own, without the need for a third party to enforce the terms of the agreement.
- Self-enforcing: The contract enforces itself, without the need for a third party to arbitrate disputes or enforce the terms of the agreement.
- Trustless: The contract does not rely on the trustworthiness of any particular party, as it is executed automatically by the underlying blockchain technology.
- Immutable: Once a smart contract is deployed on a blockchain, it cannot be modified or deleted. This ensures that the terms of the contract remain unchanged and cannot be altered by any party.
- Transparent: The terms of the contract and the execution of its clauses are visible on the blockchain, allowing for transparency and accountability.
- Secure: The use of cryptographic techniques and the decentralized nature of blockchain technology make smart contracts highly secure and resistant to tampering.

3.4. Cryptography and Encryption Algorithms

3.4.1. AES

AES (Advanced Encryption Standard) is a symmetric-key encryption algorithm that is widely used to secure data. It was first published in 1998 by the National Institute of Standards and Technology (NIST) and has since been adopted by governments and organizations around the world as a standard for secure data encryption. AES uses a fixed block size of 128 bits and supports key sizes of 128, 192, and 256 bits(Lu, Zhang and Cao, 2022). It is considered to be a very secure and effective algorithm, and is used in a variety of applications, including encryption of data at rest and in transit. The steps for encrypting and decrypting data using AES are as follows:

- The sender and receiver agree on a shared secret key. The key can be of any length, but is typically 128, 192, or 256 bits long.
- The sender uses the shared key to encrypt the plaintext message using the AES algorithm. The encrypted message, known as the ciphertext, is sent to the receiver.
- The receiver uses the same shared key to decrypt the ciphertext and recover the original plaintext message.

3.4.2. RSA

RSA is a public-key cryptography algorithm that is widely used for secure data transmission. It is named after its creators, Ron Rivest, Adi Shamir, and Leonard Adleman, who published it in 1977. In RSA, a user generates a public key and a private key. The public key is typically shared with others, and is used to encrypt messages. Only the user who has the corresponding private key can decrypt the messages. RSA is based on the mathematical fact that it is computationally infeasible to factorize a large composite number into its prime factors. This makes RSA difficult to break, even with powerful computers(Atmaja *et al.*, 2020).

To encrypt a message using RSA, the sender first obtains the recipient's public key. The sender then uses this public key to encrypt the message. The encrypted message, known as the ciphertext, can only be decrypted using the recipient's private key. To decrypt the message, the recipient uses their private key to recover the original plaintext message. Because the private key is not shared, only the recipient is able to decrypt the message(Patgiri and Singh, 2022). In RSA, the security of the algorithm is based on the computational difficulty of factoring large composite numbers. Given a composite number, it is relatively easy to find its prime factors if you know them, but it is computationally infeasible to find the prime factors of a large composite number without knowing them. This makes RSA difficult to break, even with powerful computers.

3.4.3. SHA256 Hashing

SHA-256 is a cryptographic hash function that is often used in searchable encryption schemes. In searchable encryption, a searchable index is generated from the encrypted data, and this index is used to efficiently search the encrypted data without decrypting it. The use of a cryptographic hash function, such as SHA-256, can help to ensure the security and integrity of the searchable index.(Jatikusumo and Nurhaida, 2020) When a searchable index is generated, the plaintext data is first encrypted using a searchable encryption scheme. The encrypted data is then hashed using SHA-256, and the resulting hash values are used to construct the searchable index. When a search query is made, the query is encrypted using the same searchable encryption scheme and hashed using SHA-256. The resulting hash value is then matched against the searchable index, and any matching records can be retrieved from the encrypted data without revealing the underlying plaintext. The use of SHA-256 in searchable encryption provides several benefits. First, it ensures the integrity of the searchable index, as any changes to the encrypted data will result in a different hash value. This prevents an attacker from modifying the encrypted data and potentially obtaining sensitive information. Second, it provides additional security, as the hash values in the searchable index are difficult to reverse and do not reveal any information about the underlying plaintext. This makes it more difficult for an attacker to obtain sensitive information even if they are able to access the searchable index. Overall, the use of SHA-256 in searchable encryption helps to improve the security and effectiveness of the encryption scheme.

3.5. Security Model

3.5.1. Forward Privacy:

Forward privacy, also known as "forward secrecy," is a property of certain encryption systems that ensures that the confidentiality of past communications is not compromised if the system's secret keys are later revealed. This is achieved by generating a unique session key for each individual communication, rather than using the same key for multiple communications. In the context of searchable encryption, forward privacy refers to the ability to search encrypted data without compromising the confidentiality of past searches or the data that was searched. This is important in situations where the security of the encryption system may be at risk, such as when the system's secret keys are compromised. With forward privacy, the confidentiality of past searches and the data that was searched remains protected, even if the system's keys are later revealed. Forward privacy is an important property of secure encryption systems, as it helps to ensure the confidentiality of past communications and protect against potential threats to the system's security.

3.5.2. Backward Privacy:

Backward privacy, also known as retroactive privacy, is a property of certain cryptographic protocols that allows for the deletion of previously indexed data. This is useful in the context of searchable encryption, where a user may want to remove certain sensitive information from an encrypted database without requiring the decryption of the entire database. Backward privacy ensures that the deletion of this information does not leave any trace and cannot be detected by an attacker. This is in contrast to forward privacy, which focuses on protecting the confidentiality of data as it is being indexed and added to the database.

3.5.3. Decentralization:

In the context of searchable encryption, decentralization refers to the use of distributed networks, such as blockchain technology, to store and manage encrypted data. This allows individuals and organizations to store and share sensitive information securely, without relying on a central authority to manage the encryption keys. By distributing the keys across the network, searchable encryption can provide greater security and privacy for users, while still allowing for the search and retrieval of encrypted data. This can be especially useful in applications where large amounts of sensitive data need to be shared among multiple parties, such as in healthcare or finance.

4. Design & Implementation Specification

4.1. Architecture

The architecture followed by AEBMES scheme is represented below in figure 1. Table 2 below explains technology used to execute AEBMES.

Component	**Technology used**
Cloud Backend System	Golang, AWS EC2
Cloud Database	MongoDB Atlas (cloud)
Key Manager	Hashicorp Valut Cloud
Blockchain	Hedera Hashgraph

Table 2 Technological Component

4.2. Primary Algorithms

In ths section, the key algorithms used in AEBMES scheme are explained in detail, while also mentioning the structure and use of smart contract for different steps.

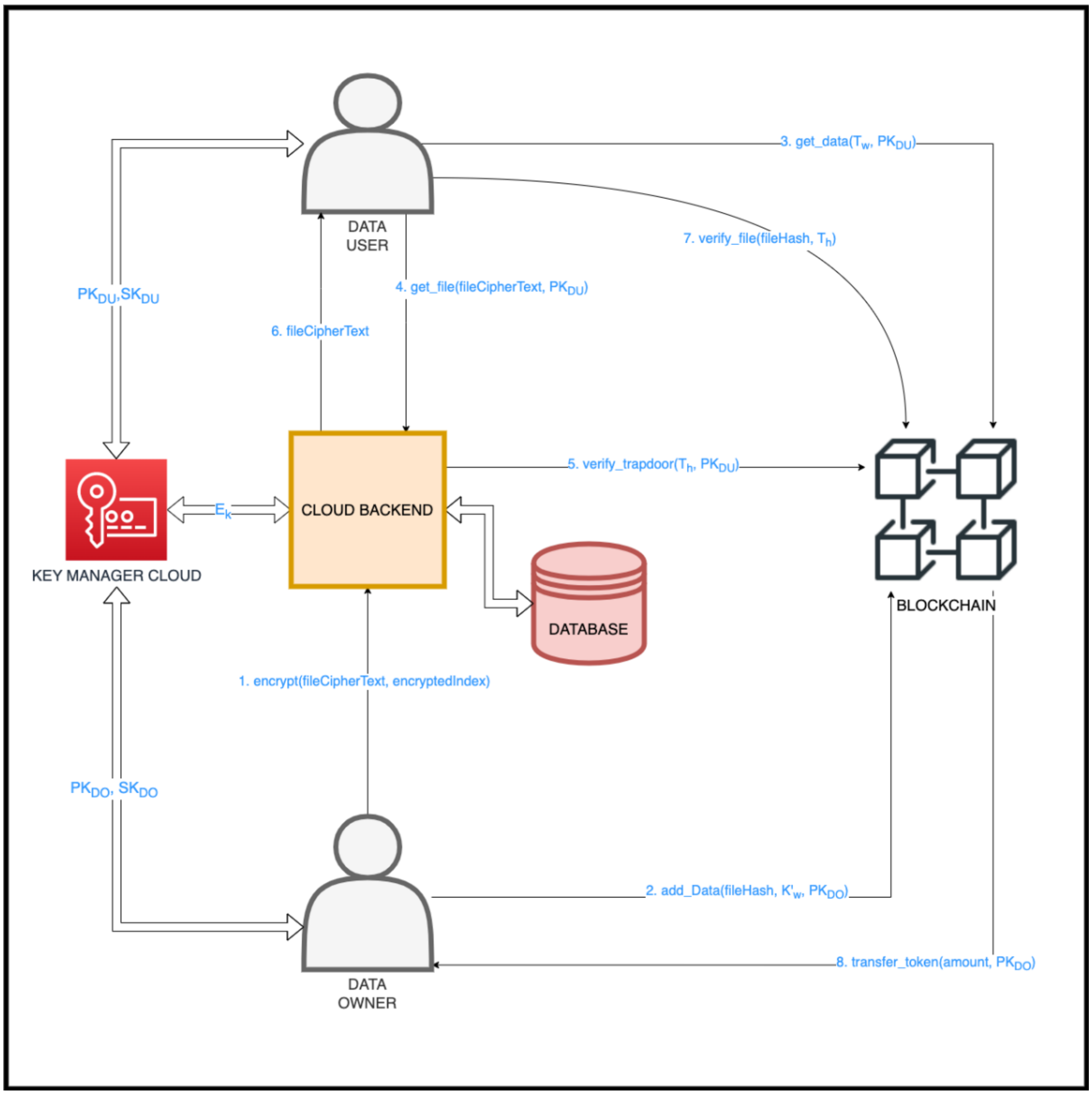
DATA USER
3. get_data(T_w, PK_{DU})
7. verify_file(fileHash, T_h)
PK_{DU},SK_{DU}
4. get_file(fileCipherText, PK_{DU})
6. fileCipherText
5. verify_trapdoor(T_h, PK_{DU})
E_k
CLOUD BACKEND
KEY MANAGER CLOUD
DATABASE
BLOCKCHAIN
1. encrypt(fileCipherText, encryptedIndex)
PK_{DO}, SK_{DO}
2. add_Data(fileHash, K'_w, PK_{DO})
8. transfer_token(amount, PK_{DO})
DATA OWNER

Figure 1 AEBMES Architecture

4.2.1. Key Generation:

- **RSA Key Generation:**

RSA keys are generated using a mathematical algorithm that involves the use of two large prime numbers. The process typically involves the following steps:

i. Choose two large prime numbers, p and q. These numbers should be chosen such that they are not easily factorizable.
ii. Compute n = p * q. n is known as the modulus and is used as part of the RSA key pair. Choose an integer e that is relatively prime to (p-1)*(q-1). e is known as the public exponent and is used as part of the public key.
iii. Compute d, the private exponent, such that (d * e) % ((p-1)*(q-1)) = 1. d is used as part of the private key.

iv. The RSA key pair is now generated, and consists of the modulus n, the public exponent e, and the private exponent d. The public key is the combination of n and e, and the private key is the combination of n and d.

These steps are performed using mathematical algorithms that are designed to be computationally infeasible to reverse. As a result, it is extremely difficult to determine the private key given only the public key, which is one of the key security properties of the RSA algorithm.

4.2.2. AES Key Generation

AES keys are generated using a mathematical algorithm that involves the use of a pseudorandom number generator. The exact process for generating an AES key varies depending on the specific implementation, but it typically involves the following steps:

i. Choose the desired key size, which can be 128, 192, or 256 bits.
ii. Use a pseudorandom number generator to generate a sequence of random bits of the chosen key size.
iii. Use a cryptographic hash function to compute a hash of the generated key. This is done to ensure the key is of high quality and has a low probability of being predictable.
iv. The resulting hash value is used as the AES key.

These steps are performed using mathematical algorithms that are designed to be computationally infeasible to reverse. As a result, it is extremely difficult to determine the original key given only the hashed value, which is one of the key security properties of the AES algorithm.

4.2.3. **Algorithm 1**: *Encrypt_File (Executed by DO)*

```
INPUT: $(K_w, PK_{DU}, File)$
OUTPUT: $(fileCiphertext, K''_w, E'_k)$

  1. Generate $E_k$
  2. Load file File => convert into []bytes
  3. Generate keyword hashes => SHA256($K_w$) => $K''_w$
  4. Encrypt file:
         aes($E_k$,FILE) => fileCipherText
  5. Encrypt $E_k$:
         rsa($SK_{DO}$, $PK_{DU}$, $E_k$)=> $E'_k$
  6. Generate File hash: sha256(File)=> fileHash


  7. SMART CONTRACT : add_data($K''_w$, fileHash)
         // Create file Map
     i.      data=map[fileCipherText] ={wordsMap(w)=>string, fileHash,cost }
         // set file hash
     ii.     data.fileHash = fileHash
     iii.    for i = 0 to n      > n = $K''_w$ length
                 data.wordsMap($K''_w$[i]) = fileHash
         end for
         // set cost
```

```
        data.cost = (readPrice)
    .
    END
```

4.2.4. **Algorithm 2**: *Generate_Trapdoor (DU)*

```
INPUT: ($S_w$, $PK_{DO}$)
OUTPUT: ($Ei$, $T_h$):
  1. Generate keyword search array => $sha256(S_w) = T_w$
  2. Smart_contract: search_trapdoor($T_w$, $Pk_{DO}$, endTimestamp)
     i.    check = endTimeStamp > block.timestamp => boolean
     ii.   if(!check) return err;
     iii.  data = map[$PK_{DO}$]
     iv.   Ei = [].    > empty array
     v.    for i=0 to n  > n=$T_w$.length
                   Ei.push(data.wordsMap[$T_w[i]$])
               End for
     vi.   Generate Trapdoor hash: $T_h$=keccak(block.timestamp+$PK_{DU}$+endTimeStamp)
     vii.  Create trapdoor map: T[$T_h$]={trapdoor_hash: $T_h$, data_user: $PK_{DU}$, fileHash:
           $T_w[0]$}
     viii. Lock readPrice : token.lockAmount($PK_{DU}$, readPrice)

END
```

4.2.5. **Algorithm 3:** *Search=>(fileCipherText) DU->CB*

```
INPUT: ($T_h$, $PK_{du}$, Ei)
OUTPUT: (Ei, $T_h$):
  1. check = Verify_trapdoor($T_h$, $PK_{du}$)
  2. if(!check) return err;
  3. search file in cloud database = search_db(Ei.value) => fileCipherText
END
```

4.2.6. **Algorithm 4:** *Verify_Trapdoor(T_h, PK_{DU}) => Boolean*

```
INPUT: $T_h$, $PK_{DU}$
OUTPUT: Boolean
  1. Smart_contract: check_trapdoor($T_h$, $Pk_{DU}$) => Boolean
     // check trapdoor map
     i.    data=trapdoor_map[$T_h$]          > {$T_h$, data_owner}
     ii.   if(data.data_owner != $PK_{DU}$)   >. Boolean
               return false;
     iii.  return true;
```

4.2.7. **Algorithm 5:** *Decrypt_File => DU*

```
INPUT: ($PK_{DO}$, fileCipherText, $E'_k$, $T_h$)
OUTPUT: $FILE^d$
  1. Decrypt keyCipherText: rsa_decryption($PK_{DO}$, $SK_{DU}$, $E'_k$) => $E_k$.     > keyCipherText
  2. Decrypt fileCipherText: aes_decrypt(fileCipherText, $E_k$)    => $FILE^d$    >output File
```

4.2.8. **Algorithm 6:** Verify_Result:

```
INPUT: ($FILE^d$, $T_h$)
```

```
OUTPUT: Boolean
  1. Create hash of output file: $fileHash^d = sha256(File^d)$
  2. smart_contract: verify_result($T_h$, $fileHash^d$) => Boolean
     i.    data=trapdoor_map[$T_h$]          > {$T_h$, data_owner, fileHash}
     ii.   bool check = data.fileHash == $fileHash^d$
     iii.  if(check)
                token.releaseAmount($PK_{DO}$).   >It will release locked token amount to
           DO
          End if
     iv.   else
                token.realaseAmount($PK_{DU}$).  > revert back amount to DU
     v.    return check
```

---

5. Evaluation

5.1. Security analysis

5.1.1. Forward Privacy:

In AEBMES, forward privacy depends on ciphertext and encrypted index updation once the file is updated. In smart contract, keywords are mapped with fileHash, once file is updated the mapping of the keywords changes, hence the old trapdoor wont give away the information of changed filehash, as the trapdoor hash will also change, hence it proves the forward privacy of AEBMES scheme.

5.1.2. Backward Privacy:

In (Bost, Minaud and Ohrimenko, 2017), a formal definition of backward privacy was presented along with three distinct kinds of leakage. In a nutshell, backward privacy conceals the existence of indexes that were created in the past but afterwards removed from use by making sure that they are not exposed in search queries.
PK_{du} with a public key constitutes a isuue that it is shared among various users. The attacker cannot obtain the useful information of indexes without the secret key SK_{DU}, even if the search results are publicly stored in a transparent blockchain. This accomplishes the desired level of anonymity when looking backwards.

5.1.3. Soundness:

In the context of blockchain-based searchable encryption, "soundness" refers to the property of a cryptographic scheme that ensures that encrypted data cannot be decrypted by anyone who does not have the correct decryption key. This property is important in searchable encryption because it ensures that only authorized users can access and read the encrypted data, even if they are able to search for keywords within the encrypted text. Soundness is a fundamental property of many cryptographic schemes, and it is essential for ensuring the security and privacy of encrypted data in blockchain-based systems. In AEBMES, User private keys are not leaked to any system or users, key-manager hashicorp ensures secure storage and distribution of the keys

5.1.4. Confidentiality:

In this scheme, the file is encrypted using a symmetric key E_k, after that the key is encrypted using RSA where SK_{DO} and PK_{DU} are used, for decryption CB cannot do it without SK_{DU}, which ensures the confidentiality of the E_k.

5.2. Comparative study

Schemes used for comparative analysis of AEBMES denoted as (1) are CL-SPKAE as (2)(Yang *et al.*, 2020) , BSPEFB(3) (Chen *et al.*, 2020) and BSMFS as (3) (Chakraborty *et al.*, 2022)

5.2.1. Feature based comparison

Table below represent feature available for various schemes

Scheme	Blockchain	Symmetric	Asymmetric	Multi - keyword search	Cloud Verification	FP	BP
CL-SPKAE	Ethereum	N	Y	Y	N	Y	N
BSPEFB	Ethereum	N	Y	Y	N	Y	Y
BSMFS	Ethereum	Y	Y	Y	N	N	N
AEBMES	Hedera	Y	Y	Y	Y	Y	Y

Table 3 Feature Comparison

5.2.2. Time Based comparision

- Encryption time

All the schemes mentioned have different names for encryption algorithm, so to generalize here is the comparison of time taken for each scheme to encrypt and store files in cloud and then store metadata in smart contract. In chart 1, it is clearly visible that AEBMES scheme is faster as compared to others. The reason is, AEBMES scheme uses less number of hashing and encryption computation and also hedera hashgraph gives faster response as compared to EVM based blockchains used by other schemes.

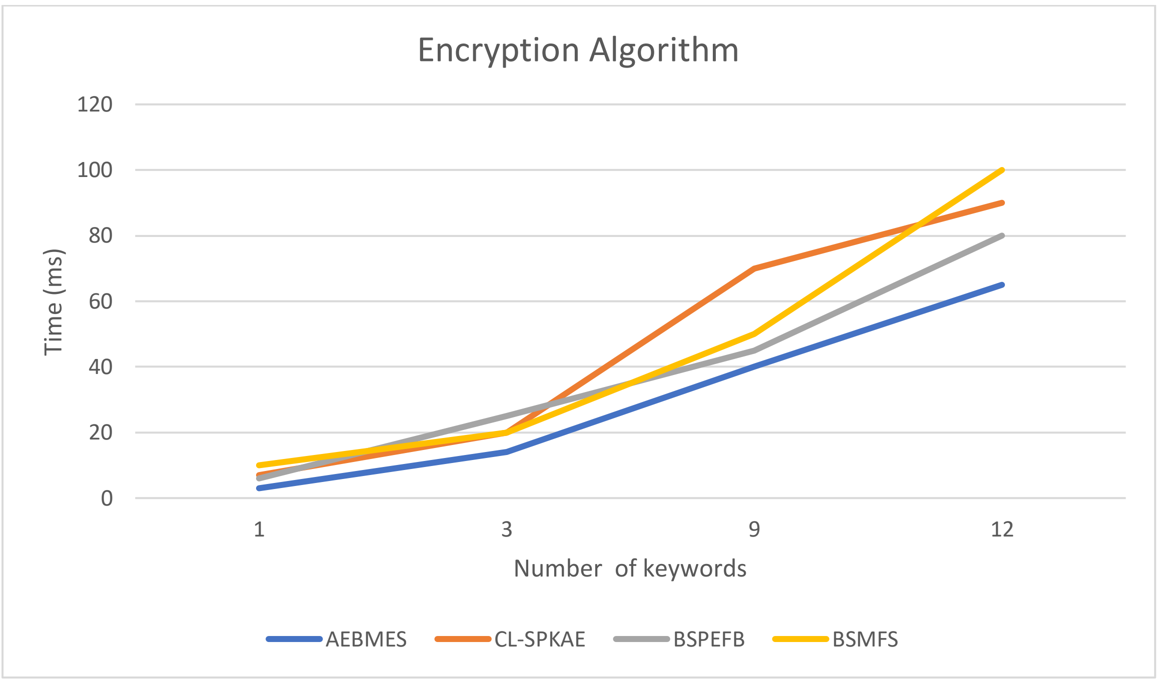

Chart 1 Computational chart for Encryption algorithm

- Search time

All the schemes mentioned have different names for searching algorithm, so to generalize here is the comparison of time taken for each scheme to create trapdoor and search stored files in cloud and then store decrypt the cipher text. In chart 2, it is clearly visible that AEBMES scheme is faster as compared to others. The reason is, AEBMES scheme uses less number of hashing and decryption computation and also hedera hashgraph gives faster response as compared to EVM based blockchains used by other schemes.

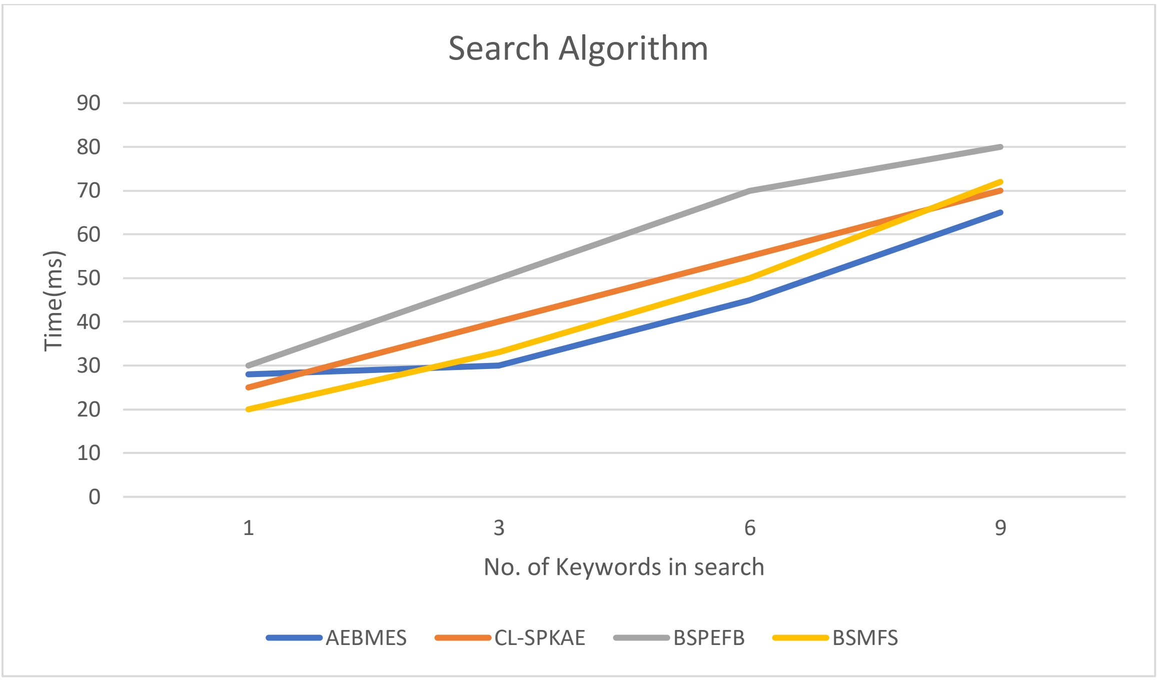

Chart 2 Search and Decryption Time Taken

6. Conclusion

There are a lot of traditional verified SSE methods that can only search single keyword files, and they don't do range searches at all. In AEBMES scheme, using symmetric and asymmetric key encryption. By utilizing an innovative SSE method, our suggested solution, Slicer, is able to achieve a verifiable and safe range search. Additionally, AEBMES assures the fairness of the system through public verification. In order to produce updated search results, AEBMES make use of a technology known as blockchain. As a result of the deployment of the forward security function, this scheme system protects users' privacy despite the presence of data updates. Formal analysis demonstrates that AEBMES is secure, fast, cost-efficient and extensive experiments demonstrate that it is also efficient. The scheme shows scope for future work to be done mostly on fuzzy keyword search and also on implementation of better smart contract storage structure.

References

Atmaja, I.M.A.D.S. *et al.* (2020) 'Document Encryption Through Asymmetric RSA Cryptography', in *2020 International Conference on Applied Science and Technology (iCAST)*. *2020 International Conference on Applied Science and Technology (iCAST)*, pp. 46–49. Available at: https://doi.org/10.1109/iCAST51016.2020.9557723.

Boneh, D. *et al.* (2004) 'Public Key Encryption with Keyword Search', in C. Cachin and J.L. Camenisch (eds) *Advances in Cryptology - EUROCRYPT 2004*. Berlin, Heidelberg: Springer Berlin Heidelberg (Lecture Notes in Computer Science), pp. 506–522. Available at: https://doi.org/10.1007/978-3-540-24676-3_30.

Bost, R., Minaud, B. and Ohrimenko, O. (2017) 'Forward and Backward Private Searchable Encryption from Constrained Cryptographic Primitives', in *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*. *CCS '17: 2017 ACM SIGSAC Conference on Computer and Communications Security*, Dallas Texas USA: ACM, pp. 1465–1482. Available at: https://doi.org/10.1145/3133956.3133980.

Cash, D. *et al.* (no date) 'Leakage-Abuse Attacks Against Searchable Encryption'. Available at: https://eprint.iacr.org/undefined/undefined (Accessed: 13 December 2022).

Chakraborty, P.S. *et al.* (2022) 'BSMFS: Blockchain assisted Secure Multi-keyword Fuzzy Search over Encrypted Data', in *2022 IEEE International Conference on Blockchain (Blockchain)*. *2022 IEEE International Conference on Blockchain (Blockchain)*, Espoo, Finland: IEEE, pp. 216–221. Available at: https://doi.org/10.1109/Blockchain55522.2022.00037.

Chen, B. *et al.* (2020) 'A Blockchain-Based Searchable Public-Key Encryption With Forward and Backward Privacy for Cloud-Assisted Vehicular Social Networks', *IEEE Transactions on Vehicular Technology*, 69(6), pp. 5813–5825. Available at: https://doi.org/10.1109/TVT.2019.2959383.

Chen, Z. *et al.* (2021) 'Blockchain-Enabled Public Key Encryption with Multi-Keyword Search in Cloud Computing', *Security and Communication Networks*. Edited by Q. Li, 2021, pp. 1–11. Available at: https://doi.org/10.1155/2021/6619689.

Dawn Xiaoding Song, Wagner, D. and Perrig, A. (2000) 'Practical techniques for searches on encrypted data', in *Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000*. *2000 IEEE Symposium on Security and Privacy*, Berkeley, CA, USA: IEEE Comput. Soc, pp. 44–55. Available at: https://doi.org/10.1109/SECPRI.2000.848445.

Gross, B. and Thibeau, D. (no date) 'Co-Founder & Chief Scientist Hedera Hashgraph'.

Jarecki, S. *et al.* (2013) 'Outsourced symmetric private information retrieval', in *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security - CCS '13*. *the 2013 ACM SIGSAC conference*, Berlin, Germany: ACM Press, pp. 875–888. Available at: https://doi.org/10.1145/2508859.2516730.

Jatikusumo, D. and Nurhaida, I. (2020) 'Data Securing of Patients in Cloud Computing Using A Combination of SHA256 and MD5', in *Proceedings of the 2020 2nd Asia Pacific Information Technology Conference*. *APIT 2020: 2020 2nd Asia Pacific Information Technology*

Conference, Bali Island Indonesia: ACM, pp. 16–22. Available at: https://doi.org/10.1145/3379310.3381043.

Li, H. *et al.* (2021) 'Blockchain-based searchable encryption with efficient result verification and fair payment', *Journal of Information Security and Applications*, 58, p. 102791. Available at: https://doi.org/10.1016/j.jisa.2021.102791.

Li, Z. *et al.* (2021) 'Forward and backward secure keyword search with flexible keyword shielding', *Information Sciences: an International Journal*, 576(C), pp. 507–521. Available at: https://doi.org/10.1016/j.ins.2021.06.048.

Lu, Y., Zhang, W. and Cao, L. (2022) 'Data Security Encryption Method Based on Improved AES Algorithm', in *2022 Global Reliability and Prognostics and Health Management (PHM-Yantai)*. *2022 Global Reliability and Prognostics and Health Management (PHM-Yantai)*, pp. 1–6. Available at: https://doi.org/10.1109/PHM-Yantai55411.2022.9942058.

Patgiri, R. and Singh, L.D. (2022) 'An Analysis on the Variants of the RSA Cryptography', in *2022 International Conference on Information Networking (ICOIN)*. *2022 International Conference on Information Networking (ICOIN)*, pp. 40–45. Available at: https://doi.org/10.1109/ICOIN53446.2022.9687262.

Wu, H. *et al.* (2022) 'Slicer: Verifiable, Secure and Fair Search over Encrypted Numerical Data Using Blockchain', in *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*. *2022 IEEE 42nd International Conference on Distributed Computing Systems (ICDCS)*, Bologna, Italy: IEEE, pp. 1201–1211. Available at: https://doi.org/10.1109/ICDCS54860.2022.00118.

Xu, W. *et al.* (2022) 'Towards efficient verifiable multi-keyword search over encrypted data based on blockchain', *PeerJ Computer Science*, 8, p. e930. Available at: https://doi.org/10.7717/peerj-cs.930.

Yang, X. *et al.* (2020) 'Multi-Keyword Certificateless Searchable Public Key Authenticated Encryption Scheme Based on Blockchain', *IEEE Access*, 8, pp. 158765–158777. Available at: https://doi.org/10.1109/ACCESS.2020.3020841.

Zheng, Q., Xu, S. and Ateniese, G. (2014) 'VABKS: Verifiable attribute-based keyword search over outsourced encrypted data', in *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*. *IEEE INFOCOM 2014 - IEEE Conference on Computer Communications*, pp. 522–530. Available at: https://doi.org/10.1109/INFOCOM.2014.6847976.