Blockchain-Driven Conjunctive Keyword Searchable Technique Over Encrypted Data for Cloud-Based Cyber Physical Social Systems

Surendra Kumar, Mridula Dwivedi, Mohit Kumar, Huaming Wu, *Senior Member, IEEE*, and Sukhpal Singh Gill

***Abstract*—Cyber-physical-social (CPS) systems require searchable encryption (SE) to safeguard sensitive data before storing it in the cloud. Existing dynamic searchable symmetric encryption (DSSE) methods have problems with index creation, document updates that lose data, and search speed. These issues must be addressed to develop an efficient CPS system. Therefore, we have developed a new approach that integrates a DSSE protocol with a blockchain-based data management system, thereby establishing a secure and efficient method for managing encrypted data. We make sure that past data remains private and that we can quickly search through data by building a forward index with a special pseudorandom function (PRF) and checking it with symmetric encryption. Keeping the encrypted index on a private distributed ledger and the secret data on a public ledger reduces storage and speeds up transaction processing. To enhance data privacy and access verification, an additional authentication system must prevent unauthorized access to the private blockchain. Authorization systems verify access permissions and execute the outcomes. Performance evaluation shows that the proposed solution improves the integrity of encrypted data and the speed of queries in the Chicago Crime and Enron Email datasets. The proposed method used only 0.68 MB client and 121.4 MB servers, builds in 121.4 s, updates in 156.4 for client and 167.2 ms for server, and outperforms all in speed, storage, and efficiency. The results are also checked for correctness and originality at the same time to reduce the complexity and computational cost by 60% and 70%, respectively, for modern cyber-physical social applications. Finally, the proposed method improves existing methods based on an extensive evaluation of Chicago Crime and Enron Email datasets, and can benefit modern CPS systems.**

***Index Terms*—Blockchain, cloud computing, conjunctive search, cyber-physical-social (CPS) system, symmetric encryption.**

Received 9 January 2025; revised 30 April 2025 and 4 July 2025; accepted 5 November 2025. This work was supported by the National Natural Science Foundation of China under Grant 62071327. *(Corresponding author: Huaming Wu.)*

Surendra Kumar is with the Department of Computer Engineering and Applications, GLA University, Mathura, India (e-mail: kumar.surendra1989@gmail.com).

Mridula Dwivedi is with the Department of Computer Science, Babasaheb Bhimrao Ambedkar University, Lucknow, India (e-mail: md.cs.bbau@gmail.com).

Mohit Kumar is with the Department of Information Technology, Dr. B.R. Ambedkar National Institute of Technology, Jalandhar, India (e-mail: kumarmohit@nitj.ac.in).

Huaming Wu is with the Center for Applied Mathematics, Tianjin University, Tianjin, China (e-mail: whming@tju.edu.cn).

Sukhpal Singh Gill is with the School of Electronic Engineering and Computer Science, Queen Mary University of London, London, U.K. (e-mail: s.s.gill@qmul.ac.uk).

Digital Object Identifier 10.1109/TCSS.2025.3630492

I. Introduction

CLOUD computing has been deeply ingrained in our society, evident from the extensive range of commercial applications in several areas, including cyber-physical-social (CPS) systems [1] and critical infrastructure sectors [2]. Cloud servers (CSs) still face vulnerabilities during data upload and analysis, despite efforts to ensure integrity [3]. Data encryption significantly impacts modern CPS applications [4]. Several methods have been proposed to encrypt data for searching, such as searchable symmetric encryption (SSE) [5]. Using advanced encryption techniques in SSE ensures data security and allows activities on secret data without requiring plaintext knowledge. SSE approaches involve retrieving constant data, which might be costly if customers change saved data [6]. Data centers in the cloud network manage and secure transmitted data [7]. It cannot prevent system failures because of cloud storage. The user's document is clear and accurate. The cloud and user may lack trust in each other due to their unique identities [8]. To enhance dynamic data updating various techniques are discussed [9]. Public cloud storage of encrypted communication documents is the most effective way to improve capacity and scalability [10]. To develop storage systems efficiently, consumers can lease CSs [11]. Improving data storage efficiency and reliability may provide good outcomes [12]. Responsibilities for data accidents include monitoring data loss and Internet of Things (IoT) device failures, that can increase operational costs [13]. IoT requires a dependable distributed system to prevent single points of failure and assure data reliability [14]. Zero-trust enforcement with blockchain, especially in edge contexts [15]. Blockchain technology can improve compliance, features, and cost efficiency in IoT applications within modern CPS contexts [16]. The proposed system analysis with data modeling is shown in Fig. 1. In Table I, searchable encryption (SE) methods are evaluated for enabling single or conjunctive queries, inverted privacy to avoid reverse keyword inference, and result in integrity verification. It improves constant time efficiency by comparing update and

TABLE I
COMPARATIVE ANALYSIS OF PROPOSED TECHNIQUE WITH EXISTING WORKS

Method	**SR-DSSE [20]**	**ESVSE [21]**	**FAST [22]**	**FSAAT [23]**	**DUAL [24]**	**ODiSC [25]**	**VBTree [26]**	**This Work**
Search Process	Single	Single	Single	Single	Single	Conjunctive	Conjunctive	Conjunctive
Forward Privacy	✓	✓	✓	✓	✓	✓	✓	✓
Inverted Privacy	✓	✗	✗	✗	✗	✓	✗	✓
Verification	✓	✓	✗	✓	✓	✗	✗	✓
Update Cost	$O(L)+1)$	$O(1)$	$O(1)$	$O(L)$	$O(1)$	$O(\xi_\kappa)$	$O(L)$	$O(1)$
Verification Cost	✗	✓	✗	$O(\Psi)$	✗	✗	✗	$O(\Psi)$
Time Analysis	✓	✗	✗	✗	✗	✗	✗	✓

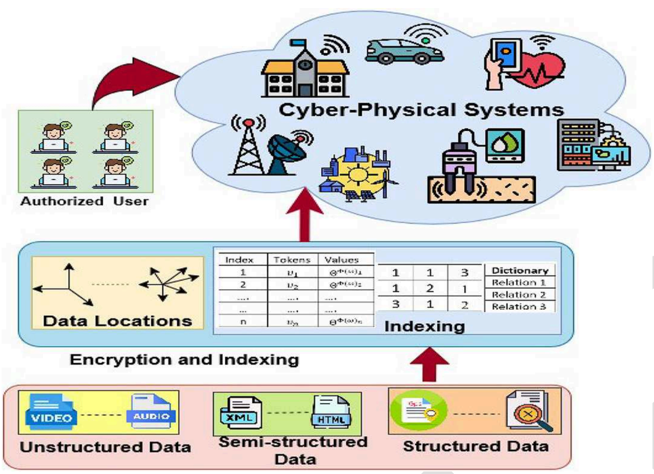


Fig. 1. Data modeling and framework analysis.

verification costs and using time analysis to evaluate real-world performance and computational feasibility.

A. Motivation

CPS systems enable smart grids, autonomous vehicles, and healthcare monitoring networks by seamlessly merging physical processes, computational power, and human interactions [17]. Cloud storage manages huge amounts of sensitive data from sensors and user inputs in these systems [18]. Traditional data management methods faced issues in meeting CPS systems' particular needs for security, privacy, real-time responsiveness, scaling to enormous datasets, and trust [19]. The proposed method is secure, efficient, and scalable cloud-based encrypted data management technology addresses these CPS-specific needs. It addresses CPS system needs such as security and privacy, real-time efficiency, scalability, and trustworthy verification.

B. Our Contributions

This study introduces a dynamic searchable symmetric encryption (DSSE) method that integrates forward security with conjunctive queries. To enable quick noninteractive query and update operations, our method requires the creation of an inverted index in addition to a forward index. The main contributions of this work are shown as follows.

1) Designed a forward index and inverted index using pseudorandom functions (PRFs) to establish document-keyword links. It reduces communication costs and enables highly scalable conjunctive and multidimensional keyword searches with minimum latency.
2) The proposed system uses symmetric cryptography entirely for verification labels and index development, efficient search result validation, accuracy, and completeness for single and conjunctive keyword queries with low computing overhead.
3) Two datasets Chicago crime and Enron email, are used for validation with the PyCrypto toolkit. The results show high update rates, enabling fast and more scalable changes to encrypted data.
4) Our caching and optimized index structures reduce redundant computation, achieving 0.68 MB client, 121.4 MB server load, and faster build/update times with improved conjunctive search efficiency for modern cyber-physical social applications.

The rest of the article is structured as follows. Section II discusses a related study that works on various applications in relation to document search for security, and privacy concerns that are combined with cloud-enabled blockchain technology. Section III discusses various SE in cloud systems' background and current situation. Blockchain security design technique is presented in Section VI. System design security parameters are evaluated in Section V. Security analysis is in Section VI. Section VII presents system analysis and results. Finally, Section VIII concludes the article and highlights the future directions.

II. RELATED WORK

Cloud systems processing outsourced data are not reliable in practice. Typical models are honest-curious entities or semihonest-curious entities [27]. The threat model attacker seeks to retrieve sensitive information from encrypted documents, rather than modifying or deleting them [28]. Search results may be manipulated or verified by an adversary. Verifiable computing approaches enable SE [29]. Validated attribute-based keyword search retrieves CSs accurately. Public-key SE systems use regular language retrieval to maintain data integrity [30]. Verified multikeyword public-key SE allows dynamic data owners (DOs) to allow search access to approved DOs [31]. Verifiable forward secure SSE is used to ensure search result trustworthiness and security [32]. Multiset hash functions update data efficiently and verifiably. Public verifiability was achieved by symmetric SE for single and Boolean keyword searches in diverse settings [33].

The verifiable SE framework provides accurate and trustworthy analytical results through a distinct evaluation [34]. Malicious CSs need anti-keyword guessing attacks (KGA)–verifiable SE framework (VSEF). Anti-KGA VSEF prevents internal adversaries on authentic CSs. It requires significant processing resources and does not combine keys for authentication [35]. For confidentiality and efficiency, classic SE methods use document token keys. Data users (DUs) hold more keys as document retrieval rises. The standard DU burden of storing the encryption key prevents key transfer and maintenance. The constant-size key created using key aggregation allows DUs to decrypt different files with one key [36]. Pairing data is transmitted using Key-aggregate searchable encryption (KASE) and the model's initial set time. Searchable encryption is achieved by integrating verification permissions from several document sets using aggregate keys [37]. Blockchain-based KASE, which requires supplementary input for secure data sharing, is unsuitable for IoT environments due to pairing costs [38]. Attribute-based encryption (ABE) shares and restricts data using search encryption. The ABE system reduces mobile device resource usage and computing strain by outsourcing decryption to third-party servers [39]. Hybrid cloud computing utilizes Ciphertext-policy ABE (CP-ABE) for data encryption, decryption, and validation, ensuring outsourced data accuracy [40]. The cloud offers flexible resources, documents, and efficient computing for smart systems [41]. Distributed system zero trust architecture implementation using learning. Machine learning and blockchain boost security and efficiency, supporting our design goals [42], [43]. They integrate with blockchain for access control and verification, providing security to the dynamic SE system [44]. Enabled hierarchical ABE with user revocation, secret key delegation, and ciphertext updating, decreasing mobile processing costs through online and offline outsourcing [45]. Blockchain can protect intelligent edge clusters, which are connected to distributed CPS systems. It shows that blockchain can improve security without sacrificing efficiency [46]. Blockchain-based encryption, employing smart contracts (SCs), enhances data sharing confidence by maintaining indexes and doing keyword searches, minimizing CS attacks [47]. Blockchain analytics improves search and encryption and users analyze search results fairness in cloud-based search [48]. Using blockchain-based anti-key leakage key aggregation SE, DUs may verify data integrity and nontampering in IoT SearchBC and verifier issues arise in blockchain solutions requiring search result verification [49].

Secure SE in cloud instances must address cryptographic and operational vulnerabilities. Cryptography inconsistency against transmission service (CI-TS) demands end-to-end verifiability since an adversarial transmission layer could induce an encrypted query and result in consistency. Cryptography uniformity for CS (CU-CS) enforces protocol constraints during query and update operations to standardize CSs. CU-RCS ensures data owners and receivers obtain consistent and correct answers from encrypted inputs. The unidentifiable trapdoor attack on sender server (UT-SS) explains that adversaries might inject or exploit indistinguishable search trapdoors, making malicious interference difficult to detect. Trapdoor uniformity for CS (TU-CS) makes all search trapdoors statistically uniform and unlinkable, preventing correlation attacks, while TU-RS makes receiver trapdoors similar to prevent keyword pattern inference and analysis leakage [50]. These ideas aim to improve SE pipeline privacy, integrity, and consistency in adversarial or semitrusted clouds.

III. Proposed Methodology

A. Creation

Here, we provide a comprehensive explanation of the construction process for the proposed algorithms. The method comprises six divided algorithms.

1) ParamGen(1^λ, n) $\rightarrow \Pi$: The system generates system parameters by executing this method, with an authentication parameter (λ).

a) Execute the function $\gamma(1^\lambda)$ to create a set $\mathbb{G}$ with a primitive order p that is greater than 2^λ.
b) Define n as the upper limit for the total document data that every DO is allowed to upload.
c) Select two unidirectional cryptographic hash routines. The functions phi and phi' are defined as follows: phi maps strings of binary digits to elements in the finite field $mathbbZ_p$, whereas phi' maps elements in the group $mathbbG$ to strings of binary digits of length m.
d) Execute the variables at $\text{AccumSetup}(\lambda) = (\nu, \upsilon)$ in order to generate the accumulation function.
e) Select a generator at random and assign the variable γ the value of the set $\mathbb{G}$ and n randomly produced items. Allocate values $\gamma_1, \ldots, \gamma_n$ from the set $\mathbb{G}$.
f) Output the external attributes and value of

$$\pi = (\gamma, \{\gamma, \ldots, \gamma_n\}, \phi, \phi', (\nu, \upsilon)). \tag{1}$$

2) Setup(Δ) $\rightarrow \Xi_\Delta$:

a) Select a random value $\alpha \leftarrow \mathbb{Z}_p$ and set DO's private key $\sigma = \alpha$ and the public secret key $\pi = \gamma^\alpha$.
b) Consider δ_i represent each searched document, where i is an element of the collection $\{1, \ldots, n\}$. The entire set of phrases that match δ_i is denoted as ω_i. Choose a random $\tau_i \leftarrow \mathbb{Z}_p$. Then, compute $\lambda_i = \text{Accumulate}(\omega_i, \nu, \upsilon)$ as an accumulator for the keyword set ω_i. A matrix indicates the presence of keywords in documents

$$\mathbf{M}_{n\times m} = \begin{cases} 1, & \text{if } \tau_i \in \delta_i, \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

Calculate every keywords

$$\chi_{ij} = \tau_i + \alpha \cdot \phi(\omega_{ij}), \forall \omega_{ij} \in \omega_i. \tag{3}$$

We set $\chi_i = \{\chi_{ij}\}$. Find υ_{ik} as $\{\gamma^{\tau_i^k}\}$ for all $k \in \{1, \ldots, n\}$, and set $\upsilon_i = \{\upsilon_{ik}\}$. With $\psi_{i1} = \gamma^{\tau_i}$ and $\psi_{i2} = \lambda_i \oplus \phi'(\gamma^{\tau_i + \alpha})$, let $\Psi_i = (\psi_{i1}, \psi_{i2})$. The key is generated based on a security parameter

$$\kappa \leftarrow \mathcal{K}(1^\lambda). \tag{4}$$

c) Apply a symmetric encryption approach $\mathbb{E} = \{\text{Setup}, \text{Enc}, \text{Dec}\}$ for executing encryption on the file itself, denoted as $\Delta_i^* = \text{Enc}(\Delta_i)$. The protect index containing this file is denoted as $\Xi_i = (\chi_i, \upsilon_i, \Psi_i)$.

d) Output $\Xi_\Delta = \{\Delta_i^*, \Xi_i\}_{i=1}^n$ and upload Ξ_Δ to the Cloud Service Provider (CSP).
Index Token for δ_i, θ_j An index token is created for each document and keyword pair

$$\pi_{i,j} = \phi_\kappa(\theta_j \| \kappa_{\delta_i}). \tag{5}$$

The encrypted index is formed as a union across all documents

$$\mathcal{I} = \bigcup_{i=1}^{n} \{\pi_{i,j}, \text{Enc}_\kappa(\kappa_{\delta_i})\}. \tag{6}$$

The query token set collects trapdoors for queried keywords

$$\Gamma = \{\gamma_{\theta_1}, \ldots, \gamma_{\theta_q}\}. \tag{7}$$

3) Share(σ, σ) $\rightarrow$ θ:

a) Given the secret key σ of DO and the complete set of authenticated documents σ which is an instance of the set $\{1, \ldots, n\}$, we define $\eta_1 = \gamma^{\alpha_1}, \ldots, \eta_n = \gamma^{\alpha_n}$. The approach computes the aggregate authorization key $\theta = \prod_{k\in\sigma} \eta_k$.

4) TokenGen(θ, ω) $\rightarrow$ υ_u:

a) The method is executed by the *DU* to produce a private key for the search term ω. Calculate the value of the token υ_u using the formula $\upsilon_u = \theta^{\phi(\omega)}$.

5) Test(Ξ_Δ, σ, υ_u) $\rightarrow$ (γ_S, Tag): while taking an encrypted token υ_u generated by the DU, the CSP executes this method to determine which documents are a match for υ_u. For every Δ_i, verify if the product of $\prod_{k\in\sigma} \gamma^{\chi_{ij}} = \prod_{k\in\sigma} \upsilon_{ik} \cdot \upsilon_u$. If the statement is true, include the document index i in the result set γ_S. Next, calculate the value of $\text{GenWitness}(\omega_i, \nu, \upsilon, \omega)$ and produce a matching proof $\text{Tag}_i = (\pi_{i1}, \pi_{i2}, \text{wit})$, where π_{i1} is equal to ψ_{i1} and π_{i2} is equal to ψ_{i2}. Ultimately, the algorithm produces the result $(\gamma_S = \{i\}, \text{Tag} = \{\text{Tag}_i\})$ and transmits it to the verification contract.

6) Verify(ω, γ_S, Tag) $\rightarrow$ ACC: The intelligent contract executes this method to verify the accuracy of the resulting dataset, ensuring that all the documents are included. Calculate the value of λ_i' for each i in the set σ using

$$\lambda_i' = \phi'(\pi \cdot \pi_{i1}) \oplus \pi_{i2}. \tag{8}$$

While certain λ_i' cannot be retrieved, terminate the process and output $\perp$. Verify the presence of the search term ω by executing the function $\text{AccVerify}(\nu, \lambda, \omega, \text{wit})$, that returns the value acc_i. The method generates the set of verification results, denoted as ACC.

7) Revoke: This can be performed at the data owner phase and data multiuser phase.

a) *Data owner phase*: By encrypting c with the shared communication key σ_{share}, the revocation command revoke $= \text{Enc}_{\sigma_{\text{share}}}(c)$ is generated once the DO has chosen the aggregated key serial number c to be revoked. In order to transmit data, the DO will receive the exchanged secret private key σ_{share} if the remote authentication is successful. The DO transmits the revocation request to the authorized execution using an encrypted communication channel.

b) *Data multiuser phase*: The trustworthy execution obtains c by decrypting the signal revoke using the public communication key σ_{share}, that is received upon obtaining the revocation request. Then it deletes the DO memory that has the matching key after looking it up. Then, it sends the encrypted result to the DO. The last step in revoking a key is for the DO to add (DU, c) to the removal list.

B. Security Proof

The validity of our proposed approach is contingent upon the accurate functioning of both the Testing and Verification functions. Upon obtaining a query token provided by the DU, the CSP performs a search method on all records to ascertain the documents that correspond to the token. The procedure is outlined as follows:

$$\begin{aligned}\sum_{k\in\lambda} \gamma_k^{\chi_{ij}} &= \sum_{k\in\lambda} \gamma_k^{\tau_i + \alpha\cdot\phi(\omega_{ij})} = \sum_{k\in\lambda} \gamma_k^{\tau_i} \cdot \sum_{k\in\lambda} \gamma_k^{\alpha\cdot\phi(\omega_{ij})} \\ &= \sum_{k\in\lambda} \gamma_k^{\tau_i} \cdot \sum_{k\in\lambda} \eta_{\phi(\omega)}^{k} = \sum_{k\in\lambda} \gamma_k^{\tau_i} \cdot \theta_{\phi(\omega)} \\ &= \sum_{k\in\sigma} \upsilon_{ik} \cdot \upsilon_u. \end{aligned} \tag{9}$$

Therefore, the user possessing the combined key can effectively conduct keyword searches. To verify and get the accumulation rate λ_i, the following steps are taken:

$$\begin{aligned}\lambda_i' &= \phi'(\pi \cdot \psi_{i1}) \oplus \psi_{i2} \\ &= \phi'(\gamma^\alpha \cdot \gamma^{\tau_i}) \oplus (\lambda_i \oplus \phi'(\gamma^{\alpha+\tau_i})) = \lambda_i. \end{aligned} \tag{10}$$

Subsequently, the AccVerify verify process was executed to confirm the existence of the search phrase ω. Ultimately, verification procedures can be completed with a successful search. The computational cost of the search intersection is calculated

$$\mathcal{T}_\cap = \sum_{j=1}^{q} \rho_j \cdot \log \rho_j. \tag{11}$$

C. Security Analysis

The security components of the proposed system are examined on the basis of integrity, privacy, and fair payment methods.

1) Proof of Integrity: To guarantee integrity, our system implements an authentication process as in SCs on the crypto blockchain system. The robustness of crypto's security, coupled with the accuracy of our algorithm, ensures the preservation of integrity. The technique is openly verifiable, enabling any miner within the cryptosystem to authenticate results from searches. To manipulate the current state and semantics of the contract, a malicious party would require more than 50% of the network's computing capacity, a highly unlikely scenario

$$\varsigma(\delta_i) = \sum_{\omega_j\in\Omega} \text{tf}(\omega_j, \delta_i) \cdot \log\left(\frac{N}{\text{df}(\omega_j)}\right) \tag{12}$$

where $\varsigma(\delta_i)$ computes the relevance score of document δ_i, Ω is the set of plaintext search keywords, tf is the term frequency of keyword ω_j in document δ_i, df is the document frequency of ω_j, N is the total number of documents in the corpus.

The proposed privacy-preserving technique satisfies integrity if the chance of the verification procedure authorizing while a searchable set fails to contain the proper keywords is extremely small. The verification phase rebuilds the accumulator $\mathbb{A}_i'$ by applying the function $\mathbb{H}'$ to the product of $\iota\daleth$ and ι_{i1}, and then performs a bitwise XOR operation with ι_{i2}. If the restoration of $\mathbb{A}_i'$ is not possible, the process will stop. Alternatively, it validates the search results by the following equation:

$$\mathrm{AccVerify}(\lambda, \nu, \mathbb{A}_i, \omega, \mathrm{wit}) \rightarrow \mathrm{acc}_i. \tag{13}$$

If the Service Provider produces an inconsistent set of results, the verification process will fail, hence guaranteeing the integrity of the security scheme.

2) Proof of Privacy: Privacy in our scheme refers to the condition that only those with proper authorization can retrieve information, while preventing unauthorized entities from gaining access to it.

Theorem 1: An attacker cannot extract search keywords from the query search token.

Proof: Token building is represented by $\mathbb{T}_\mu = \alpha\kappa._\rho^{\mathbb{H}(\omega)}$. To extract ω, an adversary $\mathbb{A}$ would need $\alpha\kappa_\rho$, which is generated using the user's private key α. Since α is not available to $\mathbb{A}$, and only public parameters along with the authorization set $\mathbb{S}$ are accessible, the likelihood of $\mathbb{A}$ obtaining α is negligible. Therefore, the opponent is unable to ascertain the search keywords.

Theorem 2: Stored ciphertext cannot be deciphered by an attacker.

Proof: The adversary can access not just the public parameters, but also the ciphertext $\mathbb{D}^*$ and the auxiliary sets $\mathbb{X}_i = (\mathbb{C}_i, \mathbb{U}_i, \mathbb{V}_i)$. The adversary may attempt the following.

1) Recover τ from υ_i and φ_{i1}; however, due to the discrete logarithm problem, this attack is infeasible.
2) Extract accumulator $\mathbb{A}$ from $\varphi_{i2} = \mathbb{A} \oplus \mathbb{H}'(\gamma^{\tau_i+\alpha})$. The secure hash function $\mathbb{H}'$ makes this extraction impractical. Each blockchain entry is a hashed combination of trapdoor and identifier:

$$\eta_i = \mathcal{H}(\gamma_\theta \| \mathrm{Enc}_\kappa(\kappa_{\delta_i})). \tag{14}$$

However, the opponent is unable to obtain any valuable data through the search query token or the encrypted data, thereby guaranteeing the confidentiality of the system.

3) Proof of Equitable Payment: Blockchain ensures fair payment and Searchable encryption systems historically used a trusted server to search and retrieve results. If multiusers get correct results, they must pay honestly, that's difficult. Users may try to avoid paying for erroneous or partial results from subscription-based systems. For fair transactions and blockchain proof of contract, we use SCs. Fair transactions require depositing the search fee before searching. The verification contract verifies search results for accuracy and completeness. After verification, the server will be paid. If the user does not match the requirements, the deposit is returned and the server is not paid, ensuring a fair transaction.

Algorithm 1: Build Index.

Require: κ: master key, γ: keyword map, ϕ: encrypted index, θ: PRF key, λ: Fg key, φ: pointer count, ω: file ID list

```
for each ω in γ do
    for i = 1 to φ do
        ρ ← λ · ω · i
        ρ' ← λ · ω · (i + 1)
        γ_i ← Encrypt(κ, i)
        Π_i ← θ · ω · ρ
        Π̄_i ← θ · ω · ρ'
        φ.append((ρ, Π_i, Π̄_i, γ_i))
    end for
    Π̄_ω ← θ · ω · ρ' · random(1, 100)
    φ.append((ρ', Π_i, Π̄_ω))
    Update γ[ω] ← (ρ, u, s)
end for
Upload φ to private blockchain return φ, γ
```

IV. System Design

A. Algorithm Design

1) Index and Search Initialization: The inputs to the scheme are id and w. The update count and search count resets to 0 after setting the search status to N, indicating the search word has not been searched shown in Algorithm 1. For every search keyword.

a) Create the search query token, denoted as t_{w1}, and key token, denoted as t_{w2}.
b) Generate a template, denoted as pt_{ri}, for each index using the search query token.
c) During the initialization process of the initial pointer, since there has been no preceding pointer, it is assigned a value of void. Next, the XOR (exclusive OR) technique is used to calculate the existing encrypted pointer value (Pi) and its matching pointer (Vi). EDB is used for storing and modifying these.
d) The state of the search query keyword is maintained in a data structure called a Map.
e) At last, the EDB is transferred to the secure blockchain through a smart contract.

The DO makes local modifications to Map.

2) Search Operation Flow: Algorithm 2 shows the procedure in which the private blockchain processes the access control data received from the DU. DU sends a query token to the private blockchain. The user ν_j is validated as authorized after analysis. The private blockchain activates a smart contract (SC) to search. Starting with a blank list $\mathbb{L}_R$, data are collected on query keyword status. The DU creates a unique token identification for the search word and analyzes $\mathbb{ST}$. If $\mathbb{ST} = \mathbb{Y}$, the query keyword remains unchanged after the search. Alternatively, it implies that the present index for the specified keyword has not been searched yet, therefore it is necessary to calculate the most recent index pointer $\rho_{\tau i}$. The search result is the intersection of document sets retrieved by trapdoors

$$\mathcal{R} = \bigcap_{j=1}^{q} \mathrm{DB}_{\mathcal{I}}(\gamma_{\theta_j}). \tag{15}$$

Algorithm 2: Search Algorithm.

procedure SEARCH(κ, Map, PRFfGg, ω, EDB)
  $\rho t_{\rho i}, \upsilon, s \leftarrow$ Map.get(ω, (None, None, None))
  if $\rho t_{\rho i}$ is None **then**
    return []
  end if
  $\tau_{1_\omega} \leftarrow$ PRFfGg $+' .' + \omega +' .' +$ str($\rho t_{\rho i}$)
  $\tau_{2_\omega} \leftarrow$ PRFfGg $+' .' + \omega +' .' +$ str($\rho t_{\rho i}$) $+' .' +' 2'$
  search_query $\leftarrow$ "ft1_ω: " + τ_{1_ω} + ", t2_ω: " + τ_{2_ω} + ", $\rho t_{\rho i}$: " + $\rho t_{\rho i}$
  print("Sending search query:", search_query)
  broadcast_query_to_blockchain(search_query)
  search_result $\leftarrow$ []
  function GET_ENCRYPTED_DATA FROM BLOCKCHAIN($\rho t_{\rho i}$)
    return [("Πi_value", "Vi_value", "$Cidi$_value")]
  end function
  function DECRYPT(data, key)
    return data ▷ Placeholder decryption logic
  end function
  for each encrypted_tuple **in** get_encrypted_data_from_blockchain($\rho t_{\rho i}$) **do**
    $\Pi i, Vi, Cidi \leftarrow$ encrypted_tuple
    decrypted_Cidi $\leftarrow$ decrypt(Cidi, κ)
    search_result.append(decrypted_Cidi)
  end forreturn search_result
end procedure

Algorithm 3: Update Process.

function ADDITION(κ, γ, ω, Map, δ_β, σ_τ)
  for ω, fid_list in δ_β **do**
    for ϕ in fid_list **do**
      $\rho \leftarrow$ randint$(1, 200)$
      $\gamma \leftarrow \gamma +' .' + \rho$
      $(\rho t, \upsilon, s) \leftarrow Map$.get$(\omega, (None, None, 0))$
      if $\rho t =$ None **then continue**
      end if $s \leftarrow s + 2$
      $len \leftarrow$ length(fid_list)
      if $\sigma_\tau =' Y'$ **then**
        $\tau_1 \leftarrow$ PRF$(\gamma, \omega +' .' + s +' .' + len)$
        $\tau_2 \leftarrow$ PRF$(\gamma, \omega +' .' + len +' .' + s)$
        $\Pi \leftarrow$ PRF$(\tau_1, \rho t)$
        Vi $\leftarrow$ PRF$(\tau_2, \rho t)$
      else $\tau_1 \leftarrow$ PRF$(\kappa, \omega +' .' + s +' .' + (len + 1))$
        $\tau_2 \leftarrow$ PRF$(\kappa, \omega +' .' + (len + 1) +' .' + s)$
        $\Pi \leftarrow$ PRF$(\tau_1, \rho t) +' ?'$
        Vi $\leftarrow$ PRF$(\tau_2, \rho t) +$ PRF(κ, ϕ)
      end if EDB_entry $\leftarrow (\Pi, Vi, \gamma, \rho t, \rho)$
      EDB_hash $\leftarrow \omega +' .' +$ PRF$(\gamma, \gamma +' .' + EDB_entry)$
      Map$[\omega] \leftarrow (\rho t, \upsilon, s)$
      upload_to_private_blockchain(EDB_hash)
      update_public_blockchain_documents(ϕ, EDB_hash)
    end for
  end for
end function

A trapdoor ensuring forward privacy is generated

$$\gamma'_\theta = \phi_\kappa(\theta \| \tau_\theta). \tag{16}$$

After obtaining authorization and control data, ν_j sends the search query token to the private blockchain. The calculations confirm that ν_j is an authorized user. The private ledger uses SCs to query the encrypted index for the search phrase ω until it reaches an empty value. Afterwards, the generated list $\mathbb{L}_R$ is created without elements. The encrypted messages retrieved and their related results generated randomly, represented as $\mathbb{L}_R = \{\mathbb{C}_{i1}, \ldots, \mathbb{C}_{in}, \alpha\omega\}$, are then sent back to the DU matching ν_j.

3) Update Process: Algorithm 3 updates the encrypted index for keyword ω using a dual-ledger system. If flag $\sigma_\tau =' Y'$, it uses state γ (updated with random $\rho \in \{1200\}$) for PRFs; otherwise, it uses key κ. For each file identifier ϕ in δ_β's fid_list for ω, it retrieves $(\rho t, \upsilon, s)$ from Map, increments counter s by 2, and computes len (length of fid_list). It generates τ_1, τ_2 (PRF outputs), Π, Vi (encrypted/verification values), and $EDB_entry = (\Pi, Vi, \gamma, \rho t, \rho)$. The entry is hashed into EDB_hash for private blockchain storage, while ϕ and EDB_hash update the public blockchain. The unused υ and set of natural numbers $\mathbb{N}$ are defined for potential extensions. The index is updated by replacing an index token

$$\mathcal{U}_\mathcal{I}(\delta, \theta) = \text{Replace}(\pi_{\delta,\theta}, \gamma'_\theta). \tag{17}$$

4) Deletion Process: Algorithm 4 manages the deletion of file identifiers for keyword ω in a dual-ledger system. For each ω and its file identifier list ι in dictionary γ, it processes each file identifier ϕ, invoking a smart contract with secret key ν to delete ϕ for ω. It updates the encrypted index database Ξ with ω, ϕ, and ν, and modifies the public blockchain for ϕ using ν. The key ν ensures secure deletion, while Ξ maintains the encrypted index integrity across the private blockchain.

Algorithm 4: Deletion Process.

function DELETE(ν, γ, Ξ)
  for each ω, ι in γ **do**
    for each ϕ in ι **do**
      CALL_DELETED_SMART_CONTRACT(ω, ϕ, ν)
      UPDATE_ENCRYPTED_INDEX_DATABASE(Ξ, ω, ϕ, ν)
      UPDATE_PUBLIC_BLOCKCHAIN_DOCUMENTS(ϕ, ν)
    end for
  end for
end function

B. Authorized Access Control

The authentication approach requires the DO to create an access management request to the private ledger by use of a smart contract. Fig. 2 shows that the authentication approach consists of both the query information and the authentication token. The requested information comprises the request ID (RID), session ID (SID), i.e., the DO's ID, and receiver entity ID (REID) i.e., data receiver's ID, represented as RIW fRID; SID; REIDg. Access is granted or denied based on user permissions

$$\mathcal{A}(\xi, \delta) = \begin{cases} 1, & \text{if } \rho(\xi) \in \text{Perm}(\delta) \\ 0, & \text{otherwise.} \end{cases} \tag{18}$$

The private blockchain then notifies all relevant DUs of the access control request. After getting the information on the access control. The DUs perform two checks. First, they check if the token has arrived and make sure it matches their own Token0. Next, DUs verify that their REID matches the REID in the request details. The users are authorized DUs if both tests succeed, as shown in Fig. 2.

1) Token Generation: In Algorithm 5, it requires U_i to acquire U_j's private key K_{uj} and its own random integer r. Next, U_i determines the Token and uses U_j's private key K_{uj} to make a query. As a transaction on the Ethereum secure blockchain,

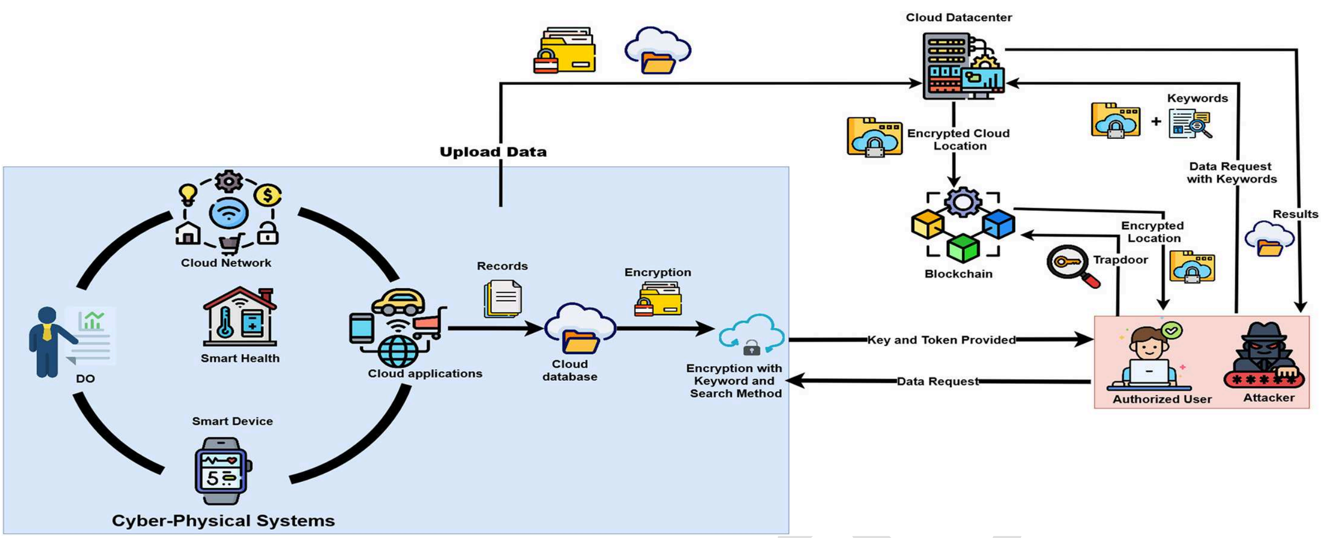


Fig. 2. Proposed architecture for SE scheme.

Algorithm 5: Access Control Token Generation.

for each r in random_numbers **do**
  Step R1: U_j computes $\theta = \phi(r)$ and $c = \mathrm{Enc}(\{\Psi, \sigma, \omega\})$.
  Step R2: Return θ and c.
end for
for each token and ciphertext pair (θ, c) **do**
  Step R3: U_i broadcasts c and θ as transactions on the Ethereum private blockchain.
end for

Algorithm 6: Retrieve Access Token.

Require: θ: token, c: encrypted request, κ: private key, ϕ: hash function, Ψ: request ID, ω: resource ID
$h \leftarrow \phi(\theta)$
if $h \neq \theta$ **then return** “Access Denied: Invalid Token”
end if
$k \leftarrow$ restorePrivateKey(κ)
$r \leftarrow$ decrypt(c, k)
if $\Psi \neq r$.ID **then return** “Access Denied: Invalid Request ID”
end if
if $\omega = r$.REID **then return** “Access Granted”
else
  return “Access Denied: Unauthorized Resource”
end if

DO transmits the encrypted text c along with the request search token after encrypting the request information RI.

2) Access Control Token Retrieval: Algorithm 6 involves the examination of the most recent transaction in the newly created block by U_j to retrieve the authentication request data c and Token. At first, the DU evaluates Token0 and verifies if the received search query token keywords it. If the analysis results in equality, U_j will recover the decryption key itself with K_{uj}, decode the encrypted data c, and compare the resulting REID with its own ID. If the query ID also is identical, the user is considered to be an authorized user. Alternatively, the process advances to the subsequent user for evaluation.

V. Security Analysis

Theorem 1. If ϕ_1, ϕ_2, and ϕ_3 are preserved PRF (ϕ_τ), υ_1 and υ_2 are random data, subsequently our system achieves λ-adaptively secure, where $\lambda = (\lambda_{\text{Stp}}, \lambda_{\text{Srch}}, \lambda_{\text{Updt}}, \lambda_{\text{Vrfy}})$ includes as follows features: $\lambda_{\text{Stp}}(\lambda) = \varnothing$, $\lambda_{\text{Srch}}(q) = \{\sigma(\omega), \text{Hist}(\omega)\}_{\omega \in q}$, $\lambda_{\text{Updt}}(\text{op}, \text{in}) = \lambda'(\text{op}, \{\kappa, |\Xi_\kappa|\}, \theta(\omega))$, $\lambda_{\text{Vrfy}}(R, \text{proof}) = \{\sigma(\omega), \theta(\omega)\}$.

Proof: Using the simulator σ to see things from the adversaries' point of view by just using the information that leaks. That can make $\lambda = (\lambda_{\text{Stp}}, \lambda_{\text{Srch}}, \lambda_{\text{Updt}}, \lambda_{\text{Vrfy}})$, that represents the following.

Scheme for Random Data: To set randomly generated data υ_1 and υ_2, the simulate σ maintains the hashing tables θ_{υ_1} and θ_{υ_2}. These tables store tuples (ι, ν, ζ), where ι represents the file identifier, ν represents the input, and ζ represents the output. If an input ν belongs to θ_{υ_1} (or θ_{υ_2}), and there exists a tuple (τ_1, τ_2, τ_3) where $\tau_2 = \nu$ is stored in θ_{υ_1} (or θ_{υ_2}), therefore the simulator will output τ_3. Alternatively, the system randomly chooses the value of ζ, and outputs it. Then, it includes the tuple values ($\varnothing$, ν, ζ) to either θ_{υ_1} or θ_{υ_2}.

Configuration Modelling: This phase follows a similar process as Algorithm 1, but with a difference. The total number of private keys, denoted as $\sigma\lambda$, has not been generated and $\Delta[\omega]$ is modified to preserve Ψ_ω, which becomes the token utilized to get the earlier search outcome for individual searches using keywords.

Simulate Updated Tokens: Since the deletion token may be generated similarly to the addition token, updating a document ID with the related keywords ξ_κ should primarily involve simulating the addition of token α, as stated in Algorithm 2 by result σ. Using addition token simulation begins by randomly initializing the values, tag, mask, and verification tags. These values are then stored in the relevant hash table, following the steps outlined in the access control algorithm. The simulator's value σ provides the simulated addition token. It is important to observe that σ generates tag using random values instead of the PRF ϕ_τ. Suppose an adversary β can discern the disparities between the actual and simulated values of tag. In that case, they possess the ability to differentiate the findings of ϕ_τ and random data. The variation between the simulated values obtained from the tag simulation and the analysis of the PRF. The restriction of ϕ_τ is denoted as $\upsilon_\phi(\lambda)_{\beta,\phi_1}$. Simulated value Δ' is used to differentiate and compared to the real value $\upsilon_\phi(\lambda)_{\gamma,\phi_2}$, while simulated result ς_t and $\varsigma_{t'}$, to the detected original value is $\upsilon_\phi(\lambda)_{\Delta,\phi_3}$.

A PRF, denoted as ϕ_τ, can utilize the result of $\phi_\tau(\kappa,\xi)$ to ascertain the membership of element ξ in set κ, as previously described. We utilize this functionality to ascertain if a search query keyword has been an addition to the search query set of a document. While an attack η may identify the variable ξ in the search κ by using κ and the outcomes of ϕ_τ, then they can create a reduction that can distinguish between the result of ϕ_τ and a random number. This function occurrence probability is $\theta_{\phi_\tau}(\lambda)_{\eta,\phi_\tau}$

$$\begin{aligned}&\Delta \Pr[\mathrm{Real}P_\alpha(\lambda)=1]-\Pr[\mathrm{Ideal}P_\alpha, S, L(\lambda)=1]\Delta\\&\le \sum_{i=1}^{n}\mathrm{AdvPRF}(\lambda)x_i,\phi_i+\Delta\mathrm{Adv}\phi_\tau(\lambda)E,\phi_\tau+\theta_{\phi_\tau}(\lambda)\eta,\phi_\tau\\&\quad+\frac{\mathrm{poly}(\lambda)}{2\lambda}.\end{aligned} \tag{19}$$

Thus, it can be inferred that the possibility of the adversary, α, being able to differentiate between the actual view and the simulated view is extremely small in λ, with the assumption that the $PRF\phi_\tau$ and $PRFs\phi_1$, ϕ_2 are safe. In addition, based on Definition 3.3, our proposed method also successfully obtains forward privacy. The data leakage parameters $L_{\mathrm{Updt}}(\mathrm{op},\mathrm{in})$ then provides the specific data used, that is, $(\mathrm{op},\{\iota\delta,|W_{\iota\delta}|\},\mathrm{ph}(w))$. Thus, $\iota\delta$ signifies the document identifier, $|W_{\iota\delta}|$ represents the number of updated search terms within the document, and $\mathrm{ph}(w)$ is related to the previously recorded instances. The overall time complexity of the proposed method is

$$\begin{aligned}T_{\mathrm{total}}&=O\sum_{i=1}^{n}W_i+\sum_{j=1}^{q}[\ell+\log B+\delta+R_j\cdot\log R_j]\\&\quad+\sum_{u=1}^{d}(\log n+\omega)+\log k.\end{aligned} \tag{20}$$

Let $D=\{d_1,d_2,\ldots,d_n\}$ be the set of documents, with W_i representing the number of keywords in document d_i. For a conjunctive query $Q=\{w_1,w_2,\ldots,w_q\}$, let R_j be the number of documents matching keyword w_j. The system utilizes B blockchain blocks and handles an update set $U=\{u_1,u_2,\ldots,u_d\}$. Furthermore, ℓ is the output length of the PRF, δ is the latency for reading from the blockchain, ω is the overhead for updating the blockchain, and k is the number of distinct user roles. The system uses a private blockchain for a low-latency δ, scalable encrypted index and a public blockchain with layer 2 solutions for cost-effective data storage. A two-phase commit protocol ensures synchronization and consistency, despite potential public ledger delays. ■

VI. Performance Evaluation

A. Experimental Setup

The results were computed using a system configured with a 12th Generation Intel(R) Core(TM) i7-1115G4 processor running at a frequency of 3.00 GHz, 16 GB of RAM, and a 64-bit Windows 11 operating system. For all schemes, we executed hash and PRFs using SHA256. To make the ciphertext human-readable, we employed base64 encoding, implemented with the pycrypto library.[1] Two distinct datasets were used to validate encrypted searches across different data types: Chicago crime statistics and Enron emails. These datasets serve to evaluate the performance of encrypted search on unstructured, semistructured, and structured data, respectively. The system employs a hybrid blockchain approach, distributing storage and control tasks across both public and private blockchains to enhance scalability and minimize overhead. With a lightweight client footprint (0.68 MB) and fast update speeds ($\approx$ 170 ms), the system supports efficient encrypted searches on various datasets. Event-driven SCs and cache reuse techniques reduce synchronization delays, improving the performance of conjunctive queries and reducing search latency by up to 60% on the Ethereum platform. To manage costs, only essential metadata is recorded on-chain, and batch updates are used to minimize transaction fees. Experimental verification was conducted in four key areas: Index and Search Initialization, Search Operation Flow, Update Process, and Deletion Process.

B. Baselines

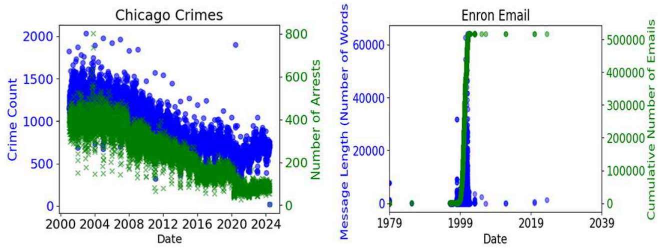
In addition, we compared the efficient update operation scheme with the conjunctive query strategies in DSSE with forward privacy [22] and [23]. In addition, we examined the verifiable DSSE with forward privacy technique from [24]. We compared our extended approach to other methods like VB-Tree [26] and ESVSE [21], which enables conjunctive queries in DSSE with forward and backward privacy, to evaluate its performance. The dual dictionary uses inverted and forwards indexes simultaneously [24]. Explicit and real-time data deletion enhances efficiency. The main advantage is forward security by encrypting new data with new keys related to previously used search tokens.

C. Dataset

In our experiment, we consider the following two datasets. The first dataset is Chicago Crime[2] includes 6 123 277 rows and 22 columns as well as accurately represents reported crimes in Chicago. The searchable term in this conventional database lacks intersections. Our initial query attribute is the object's description property, which has 173 discrete keywords (the x-axis in Fig. 3). Among the keywords, the least frequent has one record, while the greatest has 1 631 722 instances. In a

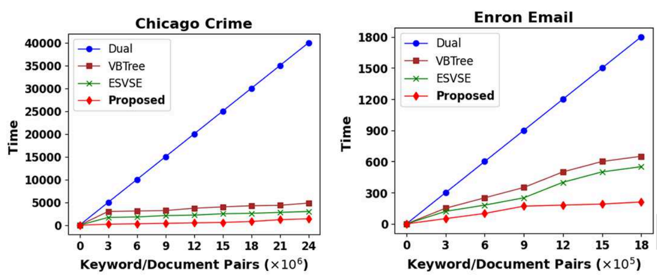
[1] https://pypi.org/project/pycrypto/

[2] https://data.cityofchicago.org/Public-Safety/Crimes-2001-to-Present/ijzp-q8t2/about_data



(a) (b)

Fig. 3. Statistical dataset representation. (a) Chicago crime. (b) Enron email.



(a) (b)

Fig. 4. Index-building time and cost. (a) Chicago crime. (b) Enron email.

database search situation, we also consider a nonskewed propagation by including a nonskewed attribute timestamp. The x-axis in Fig. 4 displays 58 403 keywords relevant to the time property. The lowest occurrence of a relevant term is 1 record, while the largest recurrence has 14 565 data.

Each of the 22 attributes displays the average and ideal rate of compression in Fig. 4(b). The Enron email dataset[3], that consists of 30 108 emails extracted from 150 Enron corporation employees' "sent mail" folder, is the additional dataset we use. Between 2000 and 2002, all of these emails have been sent. From the dataset, keywords were retrieved. The x-axis in Fig. 4(a) displays the 76 578 unique search phrases that make up the dataset. Out of these terms, the one with the lowest frequency is linked to just one document ID, while the one with the highest frequency is linked to 24 642 item IDs.

D. Index Creation and Updation Evaluations

The user interface and server storage analysis are shown in Table II. Our conjunctive query method uses less space in both cases. Fig. 4 shows the index construction time for two datasets. The Dual scheme [24] is shown, whereas VBTree [26] and ESVSE [21] define whole tree and leaf node construction, respectively. Figures shown use VBTree to represent the system. Tree node building in [26] was the most effective. There was just one hash function computation needed to create a keyword/search combination. Compared to [24], our technique uses just two computations [21].

The VBTree's leaf node building process is the slowest compared to the data structure described in [26]. To introduce intermediate nodes into a VBtree of degree L, L nodes are needed.

[3]https://www.kaggle.com/datasets/wcukierski/enron-email-dataset

TABLE II
PERFORMANCE COMPARISON OF STORAGE

AQ3

Scheme	Chicago Crime		Enron Email	
	Client	Server	Client	Server
Dual [24]	2.18	788	56.1	1185.2
VBTree [26]	1.22	541.8	33.7	1194.4
ESVSSE [21]	1.3	416.6	42.5	1228.5
This Study	**0.68**	**121.4**	**22.5**	**1028.5**

TABLE III
KEYWORD SEARCH TIME ANALYSIS (MEAN± SD)

Node(s)	50	100	150	200	250
Index Creation Time (s)	0.20	0.68	0.87	1.14	1.29
Standard Deviation (Index)	0.015	0.023	0.031	0.038	0.042
Search Time (s)	0.0214	0.0581	0.6532	0.1013	0.1369
Standard Deviation (Search)	0.002	0.0045	0.012	0.007	0.009

In comparison to [24], [21], and [26], the proposed method provides more pairs for the same dataset. Complete index development requires this final component. We ran experiments on 2.4×107 pairs during the Chicago crime experiment. [26] produced 337 774 922 nodes.

The large number of nonleaf nodes made it unsuitable for adjunct search. The index building time expenses in [21], [24], and [26], and this study were 18 546.1, 36 493.9, 16 163.2, and 14 409.6 s. Our study used 1.8 million Enron email pairs, while [26] produced 177 861 258 pairs. Previous research [21], [24], [26] and the proposed technique (1185.2, 1194.4, 1228.5, and 1028.5 s) required index building time. In the Chicago crime dataset, index-building analysis using [21], [24], [26], and the proposed method takes 788, 541.8, 416.6, and 121.4 s, respectively.

Our proposed method builds the backward and forward index in the same timeframe. Since the forward index is based on document keyword size, evaluation tests are run. Chicago crime and Enron email were selected from preexisting data for the test. Five groups of 2000 items made up the dataset. The node values of 50, 100, 150, 200, and 250 keywords per document in these categories were analyzed. According to Table II, a forward index is built in 0.25x milliseconds, where x represents the entire keyword length in the page. Table III analyses document updating effectiveness, revealing an average update time of 0.0214, 0.0581, 0.6532, 0.1013, and 0.1369 s for the entire test dataset. Our approach outperforms other schemes [24] and [21] in document updating efficiency, with index creation times of 0.20, 0.68, 0.87, 1.14, and 1.29 s, respectively.

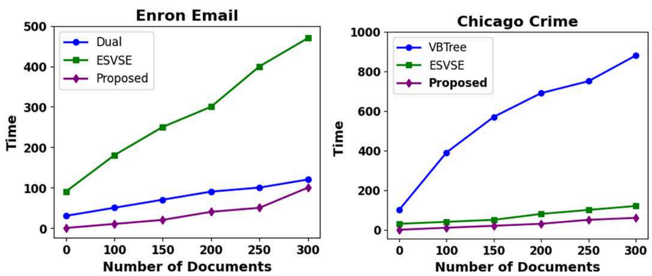
Retrieval system that focuses on a single keyword and utilizes straightforward index structures [21]. Regarding the conjunctive query technique, our approach outperforms the scheme when it comes to document updates [26]. The large quantity of indexes that must be constructed in results and poor updating efficiency as shown in Table IV.

E. Search Evaluation

An evaluation of the search approach is based on the building of a complete index, which utilizes a total of 1.8M $\times$ 106 pairings for the analysis of the Enron email and

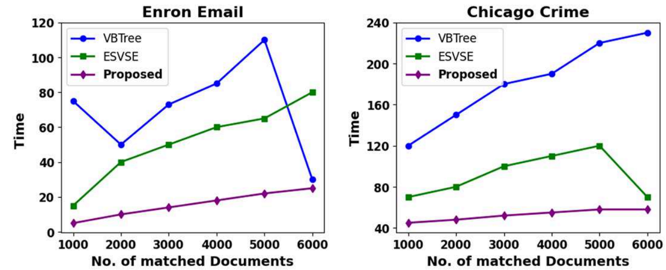
TABLE IV
EVALUATE THE EFFICIENCY OF UPDATES

Method	Chicago Crime		Enron email	
	Addition	Deletion	Addition	Deletion
Dual [24]	598.1	1592.8	562.2	952.3
VBTree [26]	256.2	547.1	361.3	763.7
ESVSSE [21]	536.3	1016.6	421.5	928.5
This Study	**156.4**	**167.2**	**151.3**	**161.8**



(a) (b)

Fig. 5. Single-keyword search evaluations. (a) Enron email. (b) Chicago crime.



(a) (b)

Fig. 6. 2-D searches using special keywords. (a) Enron email. (b) Chicago crime.

6.99M × 107 pairs for the criminal case of the Chicago crime. Initially, we conducted a trial of the search process using a single and conjunctive keyword. Subsequently, we proceeded to evaluate the effectiveness of the search process using 2-D and 3-D queries. Furthermore, we conducted experiments to evaluate the impact of the cache in our system and the efficiency when many processes are employed. The standard deviation is computed as $\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2}$, where x_i represents each observed value, μ is the mean, and $n = 300$ is the number of experimental runs using the Chicago Crime and Enron datasets. Fig. 5 displays the assessments of the single-term search procedure. Even though all searches in [26] began with a tree height of $\log_2 n - 1$, where n represents the total number of test files, search performance for a single keyword remained minimum compared to both [24] and the proposed method. This is because, in [26], the search needs to be performed around $\log_2 n$ times to locate a document. Furthermore, our approach achieved enhanced speed compared to the method described in [24] as a result of using a cache, that improved the execution of previous search outcomes.

Fig. 6 shows that the search results were unstable [21], [26]. Due to tree index data randomization, this instability exists. Effective query slicing is possible if the child nodes (documents) to be requested are mostly in the VBTree. The total number of tree nodes, including leaf and nonleaf nodes, that need access will be much fewer than the worst-case data. The total tree nodes that must be checked will be closer to the worst-case data if the leaf nodes (documents) that need to be queried are spread in the VBTree. We used specified terms in the test query to compare the effectiveness of our conjunctive searches with those in [21]. The query request included minimal terms, thus these keywords were chosen. These tests maintained the matching document's least commonly queried term quantity results of two-dimensional query studies in two datasets. The horizontal axis shows the number of extra keyword-matched pages.

The Chicago crime and Enron email dataset is used to evaluate three-dimensional queries with specified keywords due to dataset limitations. Fig. 7 shows the entire keyword-matched data on the x-axis. For a three-dimensional query request, a varied keyword provides 10 times more confirmed match data than the fixed one. The proposed method outperformed testing in search efficiency [21], [26]. The search performance was uneven, especially in the Enron email dataset [21], [26]. Results show that VBtree data distribution greatly impacts query performance. Note that optimization has limited potential to enhance datasets [21], [26]. The index data distribution is going to be random due to document addition and unexpected document content. To increase authenticity; the documents were randomly selected for testing without optimization. Under these settings, test results show uneven search efficiency [21], [26]. Due to the restricted number of test cases on the Enron email dataset, the Chicago crime dataset provides better findings, explaining this contradiction. In Fig. 7, the proposed method is evaluated with the ESVSE system [21], which provides support for conjunctive searches and achieves forward security. However, the inclusion of a time-consuming trapdoor permutation in the building of the search, which is based on RSA, greatly increases the search time cost of ESVSE compared to VBTree and our method. Based on the test results presented in both Figs. 6 and 7, it is evident that our protocol's conjunctive query performance remained consistent and improved, provided the less matched data amongst the searched terms kept constant.

Determine the least often searched phrase in Fig. 8 to estimate the conjunctive query token transmission cost to maintain 20 related data. Next, analyze the search token's transmission cost as the query dimension increases. Increased the query dimension to 4 to show that the query token's communication cost changes as the less commonly searched phrase matches more pages. The VBTree token throughput is proportional to the request size and maximum updating times for all evaluated keywords in conjunctive search, with ten updates for all analyzed keywords. The minimal search keyword and search size determine ESVSE token communication cost. Search token transmission costs are independent of the above criteria in the

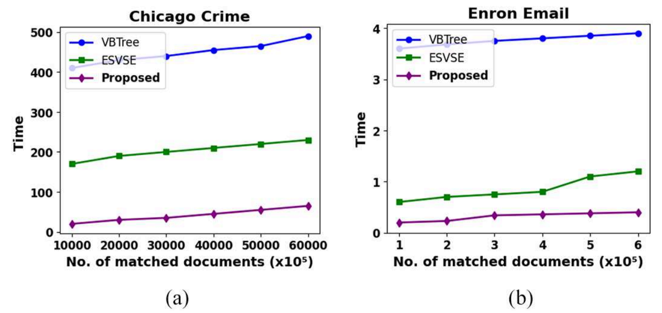


Fig. 7. 3-D keyword searches in Chicago and Enron datasets. (a) Chicago crime. (b) Enron email.

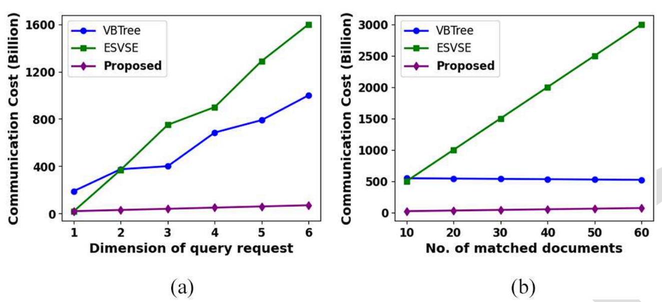


Fig. 8. Comparison of the costs of communication. (a) Chicago crime. (b) Enron email.

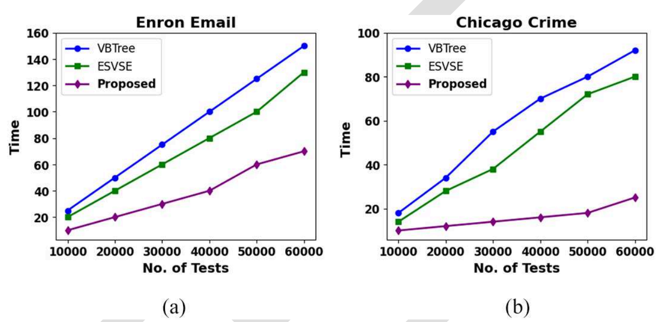


Fig. 9. Searches using 2-D with randomly generated keywords. (a) Enron email. (b) Chicago crime.

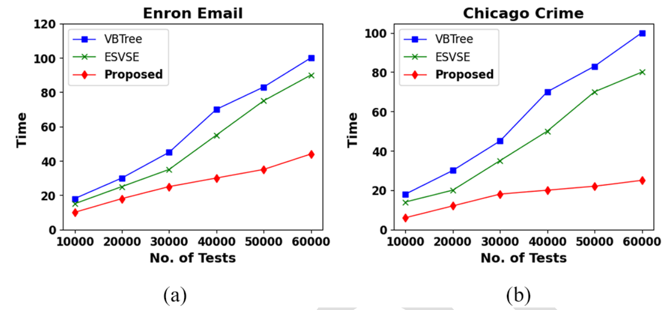


Fig. 10. Searches using 3-D with randomly generated keywords. (a) Enron email. (b) Chicago crime.

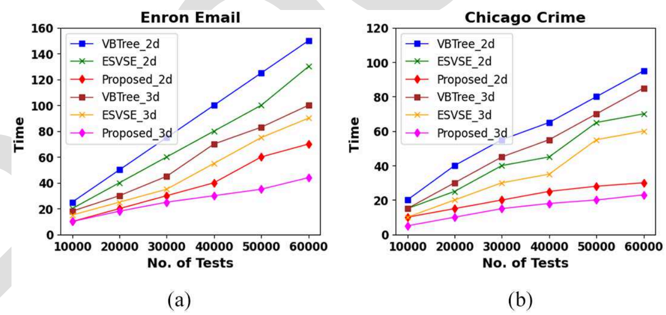


Fig. 11. Evaluation of 2-D and 3-D search keywords. (a) Enron email. (b) Chicago crime.

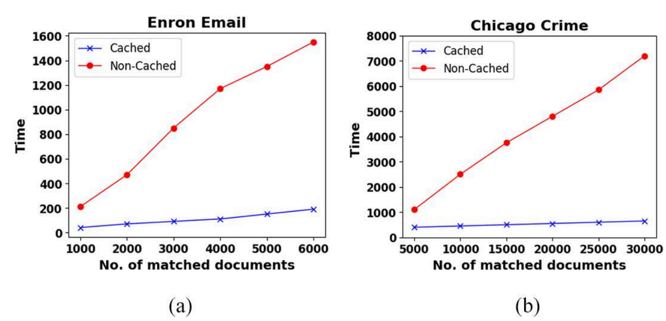


Fig. 12. Performance analysis of cached and noncached. (a) Enron email. (b) Chicago crime.

proposed method. In addition, it is smaller and more stable than VBTree and ESVSE.

To test conjunctive searches in a broad context, we randomly selected phrases and ran 60 000 tests on 2-D and 3-D queries using Chicago crime and Enron email datasets. As shown in Figs. 9 and 10, our search speed was twice as quick as VBTree and ESVSE for both 2-D and 3-D searches in an experimental environment. Comparison of 2-D and 3-D search results are shown in Fig. 11. Clearly, such artificial test results were consistent. Due to fewer matching search terms in 3-D searches than in 2-D queries, 3-D queries were faster.

F. Verification Process Evaluation

The user's verification process is analyzed. We compare the proposed method to the ESVSE [21]. Single-keyword search verification costs are evaluated between ESVSE and this study. First, ESVSE and the suggested method's evaluation cost increase positively linearly with document count. Evaluating verification data for all search term documents is required during verification. The proposed approach and ESVSE [21] have comparable verification efficiency, with the main time-consuming operation determining file authenticity taking about the same amount of time. The proposed method compares verification costs with single-keyword search and conjunctive search. Thus, conjunctive queries have twice the testing complexity of keyword queries. To ensure each item has the same calculation cost as a single-keyword query, conjunctive search computes two authentication data. Most caches perform better with more indexes. If not for the query cache, subsequent requests would have to fetch these indexes one by one, which is tedious. More cached indexes improve query efficiency.

As shown in Fig. 12, caching and optimized index structures can improve the speed of conjunctive queries, although they also provide more complexity to the system. Maintaining cache coherence and synchronized indexes can be difficult, especially in dynamic or large-scale systems. These limitations could affect the overall security, scaling, and maintenance cost of the system.

VII. Conclusion and Future Work

In this article, we propose a secure dynamic SE protocol that provides efficient index building, search, update, and deletion operations for cloud-based CPS systems. The protocol incorporates both inverted and forward index-building techniques. The protocol shows efficient document updating, achieving an average update time for both addition and deletion in the Chicago Crime and Enron Email datasets. The search process is more efficient than prior schemes, especially conjunctive queries, due to the use of caches and optimized index structures. We have proven that the protocol achieves adaptive security, incorporating leakage functions for the setup, search, update, and verification phases. Extensive experiments on the Chicago crime and Enron email datasets show the efficiency of the proposed scheme compared to existing methods, which can benefit modern CPS systems. The protocol represents a significant advancement in the field of secure and efficient SE for dynamic document collections. In the future, the middlebox for blockchain in a cloud computing environment will implement secure search using matrix queries and graph adjacency searches in conjunction with network function virtualization (NFV).

References

[1] T. Cao, J. Yi, X. Wang, H. Xiao, and C. Xu, “Interaction trust-driven data distribution for vehicle social networks: A matching theory approach,” *IEEE Trans. Comput. Social Syst.*, vol. 11, no. 3, pp. 4071–4086, Jun. 2024.

[2] R. Wang, C. Xu, and X. Zhang, “Toward materials genome big-data: A blockchain-based secure storage and efficient retrieval method,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 9, pp. 1630–1643, Sep. 2024.

[3] J. Xu, Y. Ming, Z. Wu, C. Wang, and X. Jia, “X-shard: Optimistic cross-shard transaction processing for sharding-based blockchains,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 35, no. 4, pp. 548–559, Apr. 2024.

[4] Q. Zhang, Z. Zhao, C. Li, and X. Huang, “Boosting performance of graph convolutional networks via generating pseudolabels and feature interaction,” *IEEE Trans. Comput. Social Syst.*, vol. 12, no. 4, pp. 1509–1522, Aug. 2025.

[5] S.-F. Sun et al., “Practical non-interactive searchable encryption with forward and backward privacy,” in *USENIX Netw. Distrib. System Secur. Symp.*, The Internet Soc., 2021. AQ4

[6] P. Datta, “Constrained pseudorandom functions from functional encryption,” *Theor. Comput. Sci.*, vol. 809, pp. 137–170, 2020. AQ5

[7] W. Hu et al., “An overview of hardware security and trust: Threats, countermeasures, and design tools,” *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 40, no. 6, pp. 1010–1038, Jun. 2021.

[8] H. Cui, Z. Wan, R. Gao, and H. Wang, “Outsourced privately verifiable proofs of retrievability via blockchain,” *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 4, pp. 1501–1514, Jul./Aug. 2024.

[9] K. He, J. Chen, Q. Zhou, R. Du, and Y. Xiang, “Secure dynamic searchable symmetric encryption with constant client storage cost,” *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 1538–1549, 2020.

[10] E. S. Hatay et al., “Transforming modern computing with quantum and AI: Vision and challenges,” *Int. J. Inf. Technol. Project Manage.*, vol. 16, no. 1, pp. 1–16, 2025.

[11] X. Zhou et al., “Information theoretic learning-enhanced dual-generative adversarial networks with causal representation for robust OOD generalization,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 2, pp. 2066–2079, Feb. 2025.

[12] M. M. Islam and Z. A. Bhuiyan, “An integrated scalable framework for cloud and IoT based green healthcare system,” *IEEE Access*, vol. 11, pp. 22266–22282, 2023.

[13] X. Zhou et al., “Reconstructed graph neural network with knowledge distillation for lightweight anomaly detection,” *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 35, no. 9, pp. 11817–11828, Sep. 2024.

[14] L. Rao, H. Zhang, and T. Tu, “Dynamic outsourced auditing services for cloud storage based on batch-leaves-authenticated Merkle hash tree,” *IEEE Trans. Services Comput.*, vol. 13, no. 3, pp. 451–463, May/Jun. 2020.

[15] X. Zhou et al., “Decentralized federated graph learning with lightweight zero trust architecture for next-generation networking security,” *IEEE J. Sel. Areas Commun.*, vol. 43, no. 6, pp. 1908–1922, Jun. 2025.

[16] W. Xu et al., “Multitask-based self-supervised learning for recommendation in social systems,” *IEEE Trans. Comput. Social Syst.*, early access, 2024.

[17] X. Zhou et al., “Hierarchical federated learning with social context clustering-based participant selection for internet of medical things applications,” *IEEE Trans. Comput. Social Syst.*, vol. 10, no. 4, pp. 1742–1751, Aug. 2023.

[18] X. Zhou et al., “Personalized federated learning with model-contrastive learning for multi-modal user modeling in human-centric metaverse,” *IEEE J. Sel. Areas Commun.*, vol. 42, no. 4, pp. 817–831, Apr. 2024.

[19] X. Zhou et al., “Federated distillation and blockchain empowered secure knowledge sharing for Internet of Medical Things,” *Inf. Sci.*, vol. 662, 2024, Art. no. 120217.

[20] H. Dou et al., “Dynamic searchable symmetric encryption with strong security and robustness,” *IEEE Trans. Inf. Forensics Security*, vol. 19, pp. 2370–2384, 2024.

[21] Z. Shi, X. Fu, X. Li, and K. Zhu, “ESVSSE: Enabling efficient, secure, verifiable searchable symmetric encryption,” *IEEE Trans. Knowl. Data Eng.*, vol. 34, no. 7, pp. 3241–3254, Jul. 2022.

[22] X. Song, C. Dong, D. Yuan, Q. Xu, and M. Zhao, “Forward private searchable symmetric encryption with optimized I/O efficiency,” *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 5, pp. 912–927, Sep./Oct. 2020.

[23] Y. Liu, J. Yu, M. Yang, W. Hou, and H. Wang, “Towards fully verifiable forward secure privacy preserving keyword search for IoT outsourced data,” *Future Gener. Comput. Syst.*, vol. 128, pp. 178–191, 2022.

[24] K. S. Kim et al., “Forward secure dynamic searchable symmetric encryption with efficient updates,” in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2017, pp. 1449–1463.

[25] Q. Tong et al., “Owner-free distributed symmetric searchable encryption supporting conjunctive queries,” *ACM Trans. Storage*, vol. 19, no. 4, pp. 1–25, 2023.

[26] Z. Wu and K. Li, “VBTree: forward secure conjunctive queries over encrypted data for cloud computing,” *VLDB J.*, vol. 28, no. 1, pp. 25–46, 2019.

[27] C. Guo, W. Li, X. Tang, K.-K. R. Choo, and Y. Liu, “Forward private verifiable dynamic searchable symmetric encryption with efficient conjunctive query,” *IEEE Trans. Dependable Secure Comput.*, vol. 21, no. 2, pp. 746–763, Jan. 2023.

[28] X. Liu, G. Yang, Y. Mu, and R. H. Deng, “Multi-user verifiable searchable symmetric encryption for cloud storage,” *IEEE Trans. Dependable Secure Comput.*, vol. 17, no. 6, pp. 1322–1332, Nov./Dec. 2020.

[29] Y. Sun, L. Han, J. Bi, X. Tan, and Q. Xie, “Verifiable attribute-based keyword search scheme over encrypted data for personal health records in cloud,” *J. Cloud Comput.*, vol. 12, no. 1, 2023, Art. no. 77.

[30] M. Morales-Sandoval, M. H. Cabello, H. M. Marin-Castro, and J. L. G. Compean, “Attribute-based encryption approach for storage, sharing and retrieval of encrypted data in the cloud,” *IEEE Access*, vol. 8, pp. 170101–170116, 2020.

[31] Y. Liang, Y. Li, K. Zhang, and Z. Wu, “VMSE: Verifiable multi-keyword searchable encryption in multi-user setting supporting keywords updating,” *J. Inf. Secur. Appl.*, vol. 76, 2023, Art. no. 103518.

[32] Y. Guo, C. Zhang, C. Wang, and X. Jia, “Towards public verifiable and forward-privacy encrypted search by using blockchain,” *IEEE Trans. Dependable Secure Comput.*, vol. 20, no. 3, pp. 2111–2126, May 2022.

[33] G. Duan and S. Li, “Verifiable and searchable symmetric encryption scheme based on the public key cryptosystem,” *Electronics*, vol. 12, no. 18, 2023, Art. no. 3965.

[34] Y. Miao et al., “Verifiable searchable encryption framework against insider keyword-guessing attack in cloud storage,” *IEEE Trans. Cloud Comput.*, vol. 10, no. 2, pp. 835–848, Apr./Jun. 2020.

[35] T. Wang et al., “An efficient verifiable searchable encryption scheme with aggregating authorization for blockchain-enabled IoT,” *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20666–20680, Oct. 2022.

[36] K. Alimohammadi, M. Bayat, and H. H. Javadi, “A secure key-aggregate authentication cryptosystem for data sharing in dynamic cloud storage,” *Multimedia Tools Appl.*, vol. 79, pp. 2855–2872, 2020.

[37] J. Liu, B. Zhao, J. Qin, X. Hou, and J. Ma, “Key-aggregate searchable encryption supporting conjunctive queries for flexible data sharing in the cloud,” *Inf. Sci.*, vol. 645, 2023, Art. no. 119336.

[38] J. Lee et al., “Blockchain-enabled key aggregate searchable encryption scheme for personal health record sharing with multi-delegation,” *IEEE Internet Things J.*, vol. 11, no. 10, pp. 17482–17494, May 2024.

[39] Z. Hou, J. Ning, X. Huang, S. Xu, and L. Y. Zhang, “Blockchain-based efficient verifiable outsourced attribute-based encryption in cloud,” *Comput. Standards Interf.*, vol. 90, 2024, Art. no. 103854.

[40] B. Mishra, D. Jena, and S. Patnaik, “Fine-grained access control of files stored in cloud storage with traceable and revocable multi-authority CP-ABE scheme,” *Int. J. Grid Util. Comput.*, vol. 14, no. 4, pp. 320–338, 2023.

[41] P. K. Donta, B. Sedlak, V. Casamayor Pujol, and S. Dustdar, “Governance and sustainability of distributed continuum systems: A big data approach,” *J. Big Data*, vol. 10, no. 1, 2023, Art. no. 53.

[42] I. Murturi, P. K. Donta, and V. C. A. Pujol, “Learning-driven zero trust in distributed computing continuum systems,” in *Proc. IEEE Int. Conf Dependable, Auton. Secure Comput., Int. Conf. Pervasive Intell. Comput., Int. Conf. Cloud Big Data Comput., Int. Conf. Cyber Sci. Technol. Congr.*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 0044–0049.

[43] M. Golec et al., “Artificial intelligence (AI): Foundations, trends and future directions,” *Telematics Inf. Rep.*, 2025, Art. no. 100116. AQ6

[44] C. Bicer, I. Murturi, P. K. Donta, and S. Dustdar, “Blockchain-based zero trust on the edge,” in *Proc. Int. Conf. Comput. Sci. Comput. Intell. (CSCI)*, Piscataway, NJ, USA: IEEE Press, 2023, pp. 1006–1013.

[45] F. Meng, “Online/offline attribute-based searchable encryption revised: Flexibility, security and efficiency,” *J. Syst. Archit.*, vol. 146, 2024, Art. no. 103047.

[46] D. Chinmaya, S. Satish, and P. K. Donta, “Securing clustered edge intelligence with blockchain,” *IEEE Consum. Electron. Mag.*, vol. 13, no. 1, pp. 22–29, Jan. 2024.

[47] H. Gao et al., “BFR-SE: A blockchain-based fair and reliable searchable encryption scheme for IoT with fine-grained access control in cloud environment,” *Wireless Commun. Mobile Comput.*, vol. 2021, no. 1, 2021, Art. no. 5340116.

[48] D. Zhang, S. Wang, Q. Zhang, and Y. Zhang, “Attribute based conjunctive keywords search with verifiability and fair payment using blockchain,” *IEEE Trans. Services Comput.*, vol. 16, no. 6, pp. 4168–4182, Nov./Dec. 2023.

[49] J. Niu, X. Li, J. Gao, and Y. Han, “Blockchain-based anti-key-leakage key aggregation searchable encryption for IoT,” *IEEE Internet Things J.*, vol. 7, no. 2, pp. 1502–1518, Feb. 2020.

[50] Q. Huang and H. Li, “An efficient public-key searchable encryption scheme secure against inside keyword guessing attacks,” *Inf. Sci.*, vol. 403, pp. 1–14, 2017.