SSEforIoT: Searchable Symmetric Encryption for Images on IoT Platforms

1st Ifeoluwapo Aribilola *
School of Computer Science,
University of Galway
Galway, Ireland
ifeoluwapo.aribilola@universityofgalway.ie
0000-0003-4488-2925

2nd Sofie Van Acker
Faculty of Engineering & Technology,
Technological University of the Shannon:
Midlands Midwest
Athlone, Ireland
sofieva@hotmail.com

3rd Amna Shifa *
School of Computer Science,
University of Galway
Galway, Ireland
amna.shifa@universityofgalway.ie
0000-0003-0872-6474

4th John G. Breslin
School of Engineering
University of Galway
Galway, Ireland
john.breslin@universityofgalway.ie
0000-0001-5790-050X

5th Mamoona Naveed Asghar
School of Computer Science,
University of Galway
Galway, Ireland
mamoona.asghar@universityofgalway.ie
0000-0001-7460-266X

***Abstract*—With the expansion of IoT technologies and their dependence on cloud providers for storage and computation, the security and privacy of data stored by IoMT users remain a major concern. Even simple operations, such as the search function, often require data owners to entrust the security of their sensitive data to third-party cloud providers. Similarly, there are concerns about the efficiency of classic encryption schemes in protecting users' sensitive data, as traditional encryption modes are seen as a tacit trade-off between security and utility. To address these challenges, this research investigates the feasibility and efficiency of applying a Symmetric Encryption (SE) scheme on digital images stored on a remote cloud platform within an IoT environment and proposes a searchable SE scheme that integrates a lightweight Advanced Encryption Standard in Galois Counter Mode to secure the image data. This scheme enables search-function operations to be performed directly on encrypted data without requiring decryption, thereby preserving data privacy. The proposed solution is evaluated through an in-house mobile application for data stored by a remote cloud database containing digital images, which can be retrieved through an annotations-based search function.**

***Index Terms*—Advanced Encryption Standard, Cloud storage, Homomorphic Encryption, IoMT, Image Annotation, Image Collection**

I. Introduction

Driven by rapid growth, multimedia has become central to many IoT applications, leading to the Internet of Multimedia Things (IoMT) [1]. In areas such as surveillance and security, massive image data is generated on devices with limited resources, making them dependent on cloud providers for storage and computation [2], [3]. IoMT faces unique challenges, including heterogeneous data, high throughput, QoS demands, and the CIA triad of security.

Effective IoMT security requires robust and sustainable encryption algorithms [4], [5]. Traditional schemes secure communication and storage but block computation on encrypted data. Since distributed and third-party cloud services are central to IoT, risks arise when secret keys must be shared with untrusted providers to enable processing [6].

Recent advancements in homomorphic encryption offer promising solutions to the challenges outlined above. Since the late 1970s, researchers have pursued the development of cryptosystems capable of performing accurate computations directly on encrypted data without requiring decryption. Modern schemes such as Brakerski/Fan-Vercauteren (BFV) and Cheon-Kim-Kim-Song (CKKS) enable privacy preservation and strengthen security when engaging with third-party cloud providers in the context of Big Data [7]. Despite their potential, these schemes face significant efficiency challenges due to their reliance on bootstrapping and modulus switching operations, which result in large ciphertext sizes. Furthermore, their applicability remains limited, as operations such as encrypted comparison, sorting, or evaluating regular expressions are not yet feasible within Symmetric Encryption (SE) frameworks [7].

The complexity and resource constraints in IoT and IoMT environments hinder the adoption of intensive, privacy-preserving security solutions. This paper introduces an SE scheme using client-side AES to secure cloud data, evaluating SE's feasibility and efficiency in IoT through simulated

This publication has emanated from research supported in part by the European Digital Innovation Hub Data2Sustain, co-funded by Ireland's National Recovery and Resilience Plan (the EU's Recovery and Resilience Facility), the Digital Europe Programme, and the Government of Ireland, and also by a grant from Taighde Éireann - Research Ireland under Grant Number 12/RC/2289_P2 (Insight).

Corresponding author *
All authors work equally on this manuscript.

scenarios. The key contributions of this research are:

1) A practical and efficient Searchable SE (SSE) scheme integrating lightweight AES-GCM (Advanced Encryption Standard – Galois/Counter Mode) for client-side encryption.
2) A custom mobile application to evaluate the feasibility and resource requirements of the proposed scheme.
3) Demonstration of secure cloud-based image retrieval without requiring decryption, simulating real-world scenarios.
4) Comprehensive assessment of computational efficiency, resource utilisation, and retrieval accuracy for encrypted images.

The remainder of this paper is organised as follows: Section II describes a comprehensive recent research contribution to this subject. In Section III a detailed description of the proposed scheme is presented, which includes the creation of an Android test environment to emulate an IoMT application to store digital images with annotations and metadata in a remote cloud solution and the retrieval of the images through an annotation-based search function. Section IV presents the results on mobile app design and implementation of the searchable symmetric encryption scheme. The performance analysis and limitations, along with future work extensions, are also presented in Section V. Finally, Section VI presents the concluding remarks.

II. Background and Related Studies

The widespread adoption of handheld devices equipped with multimedia capabilities has led to an exponential increase in the creation of digital images. Alongside visual content, standardised information such as camera settings, licensing details, geolocation, and time and date stamps is typically embedded within images [8]. However, this metadata is often lost when images are accessed, transferred, or stored outside their original source. To address this challenge, [9] proposed storing metadata in the Exif segment of the JPEG header to ensure its persistence within digital images. Building on this, recent research has explored the automatic embedding of metadata using machine learning (ML) and artificial intelligence (AI) techniques for image recognition [3]. Other approaches seek to integrate contextual data from nearby IoT devices to enable advanced search, query, and organisation functionalities. For example, [10] utilises embedded ubiquitous sensing technologies to generate context-rich, time-sequenced metadata. Similarly, [8] proposed a model that captures IoT data from Bluetooth Low Energy devices—including wearables, smartphones, and home-automation systems—to embed metadata directly into digital snapshots.

Given the portable and compact nature of IoMT devices, their memory and data processing capacities are inherently limited. As a result, most IoMT clients rely on cloud services for data storage and analytics to enhance efficiency and extend device functionality [11]. However, the interconnection of such devices, which frequently handle highly private and sensitive information, introduces significant security risks. To address these concerns, [12] proposed a multi-level encryption-based security system for surveillance videos to strengthen the protection of video data. Similarly, [13] introduced a secure service discovery model leveraging blockchain-enabled fog computing. In parallel, other studies suggest the use of computationally lightweight encryption ciphers within selective encryption schemes as a practical means to overcome the resource constraints inherent in IoT environments [4], [5].

Symmetric Encryption (SE) is a promising solution for securing cloud systems [14]. Homomorphic Encryption (HE) was first introduced in 1978 by [15] to enable computations on encrypted data without prior decryption. Since then, various schemes have supported modular multiplication, addition, and limited AND operations over ciphertexts [16]–[18]. Unlike partial or somewhat homomorphic schemes, SE supports unlimited operations [19]. A quick and efficient symmetric encryption method that enables ranked search through the k-means clustering algorithm, along with an asymmetric scalar-product-preserving encryption approach for encrypting indexes and queries, was proposed by [20]. [21] presents a tool for protecting private data in cloud databases using agents and APIs as an approach of SSE, preventing data leakage and privacy breaches while allowing users to store data and search keywords in their encrypted database.

On the server side, ciphertext batching via the Chinese remainder theorem has accelerated computations, while compression techniques using polynomial coefficients reduced data traffic [22]. However, such compression is unsuitable for client-side optimisation due to the lack of efficient unpacking methods. However, client-side optimisations primarily focus on reducing ciphertext size to improve the efficiency and speed of encryption and decryption processes. [23] proposed a transciphering approach that leverages an HE algorithm in conjunction with an SE scheme. SE offers significant advantages in this context, as the ciphertext size is equal to the plaintext size, enabling faster encryption and reducing storage requirements compared to alternative algorithms.

Only a few HE libraries, such as Microsoft SEAL (Microsoft, 2021), are available for Android, but their CKKS implementation yields approximate results, making them unsuitable for precise encrypted searches. To address this and ensure efficiency in resource-constrained IoT environments, this research uses client-side AES encryption to secure image locations and protect the images from unauthorised access.

III. Materials & Methods

This work builds upon previous advances by incorporating client-side AES-GCM encryption to create an efficient and practical framework for SE. Implemented in an Android test environment simulating an IoT application, it enables secure cloud storage and retrieval of digital images. The main operation is an unlimited search function performed directly on the ciphertext without changing the plaintext, fully adhering to the SE paradigm. The subsequent sections outline the implementation and assessment of the searchable symmetric encryption method designed for securing image retrieval.

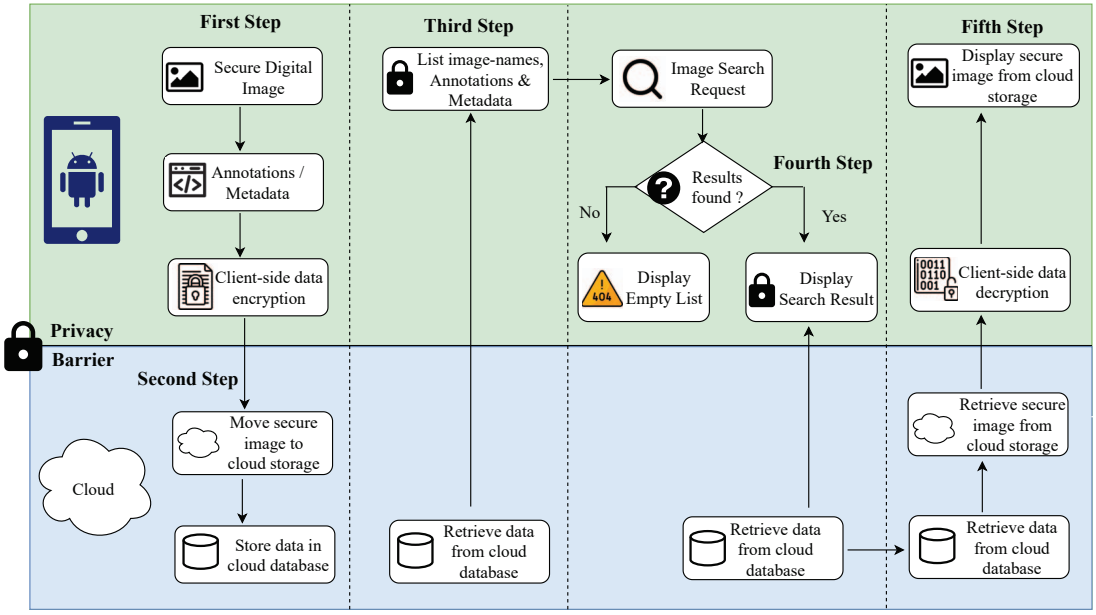


Fig. 1. Proposed Searchable Symmetric Encryption (SSE) scheme.

A. Proposed Searchable Symmetric Encryption (SSE) Scheme

At a high level, the proposed SSE scheme involves five main steps:

1) Select and annotate images, then encrypt the image URL on the local device.
2) Upload the encrypted image to cloud storage and store its metadata in the cloud database.
3) Display and search plaintext metadata by indexing (names, annotations, and other metadata) on the local device.
4) Show search results in plaintext from the cloud database.
5) Decrypt the image URL client-side and display the secured image.

"Fig. 1" illustrates the proposed scheme. The mobile app on an Android device lets users select images and add metadata (user ID, timestamp, keywords). The image URL is encrypted at the client-side before upload. The encrypted image is stored in cloud storage, while metadata and the encrypted URL are saved in the cloud database. Searchable plaintext metadata is listed locally for querying, and search results display relevant metadata. With proper authorisation and private AES keys, the image URL is decrypted locally, allowing secure image display.

The privacy barrier in "Fig. 1" highlights the scheme's ability to preserve privacy: unencrypted data is never exposed to external parties, including cloud providers. Sensitive fields are selectively encrypted, keeping images searchable while restricting access to authorised users with private AES keys. Unlike traditional solutions (e.g., Microsoft Azure), which require decryption for search and rely solely on provider access controls, this approach ensures stronger data privacy.

B. System Architecture

As illustrated in "Fig. 2", the system architecture centres on a smartphone device (client) that enables image capture and annotation through text fields to aid retrieval. When an image is selected, the app records the file extension, user ID, and upload timestamp. The user ID is assigned by the app, with authentication via email/password or Google/Microsoft accounts (detailed later).

The application extracts the URL of the selected image, encrypts it on the client side with a private key, and then uploads the image file, name, and encrypted URL to the cloud. The encryption keys are generated by the app and securely stored on the client device or key store.

On the cloud side, the server hosts the code binary and public keys, while the database records include the image name, annotations, timestamp, user ID, and encrypted image URL. The actual image and its metadata are stored in cloud storage.

C. Test Environment Set-up and Implementation Process

To address the research challenges of creating the test environment and implementing the proposed SSE scheme, the implementation process is divided into four main components. The first focuses on the development of the front-end Android mobile application, while the second addresses the design of the back-end cloud storage database. The third component involves building the front-end image gallery with an integrated search function for image retrieval. Finally, the fourth component implements and evaluates the SSE scheme.

1) Creation of the front-end android mobile application: The front-end application, *SSEforIoT*, is designed to emulate an IoT environment that requires image annotation and the uploading of digital images to a remote cloud database for

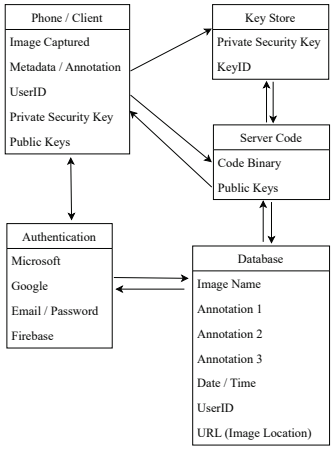
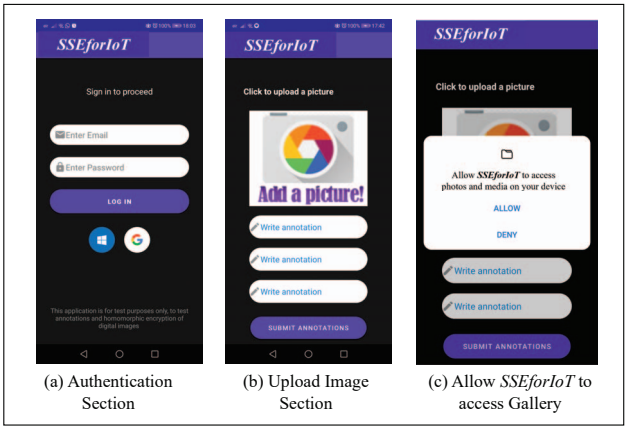


Fig. 2. System Architecture for SSE.

storage or further analysis, such as in surveillance systems. As shown in "Fig. 3", the application includes both the authentication activity and the image-upload activity, which enables users to retrieve digital images from the device's internal memory or gallery. The application begins with an authentication request managed through Firebase Authentication. The initial screen, illustrated in "Fig. 3", presents the standard email and password login option, while the two icons located beneath the "Log In" button provide alternatives for authentication via Microsoft or Google accounts.

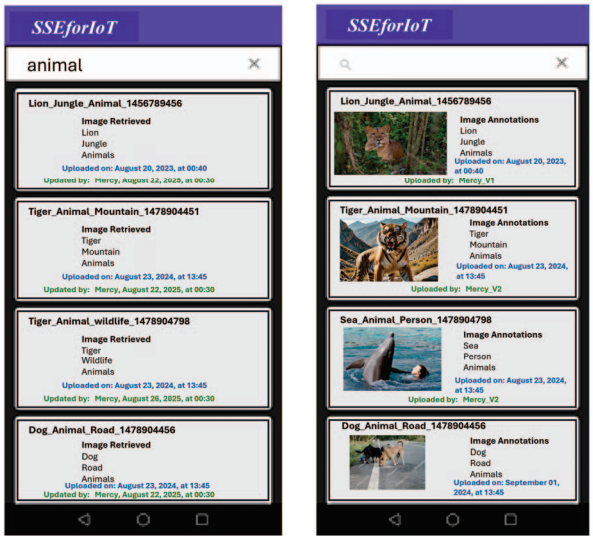


(a) Authentication Section (b) Upload Image Section (c) Allow *SSEforIoT* to access Gallery

Fig. 3. *SSEforIoT* Authentication and picture upload activities.

2) Creation of the back-end cloud storage database: M-IoT applications typically generate and transmit large volumes of data to remote cloud providers. To replicate this behavior, the test application *SSEforIoT* stores digital images along with their metadata and annotations in Firebase Realtime Database and Firebase Storage. The back-end database supports image retrieval by maintaining structured records that include annotations and relevant metadata for each image. Without the direct link to the storage location, the image cannot be retrieved by the front-end application.

3) Creation of the image gallery with search function: To enable image retrieval from the cloud storage back end, the



(a) Image search view (b) Image detail view

Fig. 4. Visual results of image search and image detail view.

SSEforIoT application includes an activity designed to display images along with their annotations and associated metadata. As shown in "Fig. 4", this information is presented in a user-friendly list or gallery view. To ensure optimal responsiveness, the gallery is implemented using a RecyclerView rather than a traditional ListView or GridView. Users type a keyword in the app's search bar positioned at the top of the gallery; the list (RecyclerView) is then filtered case-insensitively by entering those plaintext metadata fields. If no matches are found, the list remains empty. Else, selecting the found match opens a detailed view that displays the image, its name, annotations, upload date and time, user ID, and the unencrypted image location.

At this stage of development, client-side AES encryption has not yet been applied, meaning that images, annotations, and metadata remain fully visible within the gallery. The next section discusses the gallery implementation with secured images.

D. Implementation of the SSE Scheme

This stage aims to implement the proposed SSE scheme to ensure secure storage and retrieval of digital images in the cloud. As noted earlier, third-party cloud providers typically apply server-side encryption by encrypting data before writing it to disk and decrypting it upon access by authenticated or authorised users. In this work, server-side encryption is complemented with client-side encryption, whereby *SSEforIoT* applies encryption prior to transmitting data to the cloud, thereby securing it during transit.

The SSE scheme is realised in the *SSEforIoT* application through client-side AES-GCM encryption of the image URL during the image upload activity. AES-GCM employs counter (CTR) mode encryption while generating a Message Authentication Code (MAC) using 128-bit binary multiplication [24]. This approach ensures both confidentiality and integrity.

The encrypted element is the image URL only, which allows the system to maintain a searchable database while restricting

direct access to the images themselves. The SSE framework thus enables specific operations—most notably the search function—to be executed directly on encrypted data without requiring decryption, thereby preserving privacy. The app searches with plaintext metadata (image name, annotations, and other metadata). There is no encrypted keyword matching over ciphertext fields.

Furthermore, *SSEforIoT* securely stores the private encryption keys on the client device, with key management handled through the Android KeyStore. The algorithm for the SE scheme is illustrated in "Algorithms 1 and 2".

Algorithm 1: Performing SSE Scheme

Data: Load *SSEforIoT* on the mobile
while *SSEforIoT* == *True* **do**
  $image\_select \Leftarrow select\ image\ from\ gallery$
  $image\_annotate \Leftarrow Annotate(image\_select)$
  $image\_urlEncrypt \Leftarrow AES(selected\_image\_url)$
  $image\_encrypt \Leftarrow AES(image\_select)$
  $image\_cloudStore \Leftarrow Cloud(image\_encrypt)$

Algorithm 2: Retrieve image without encryption alteration

Data: From Cloud
$search\_image \Leftarrow imageSearch\ (using\ Name\ Annotation\ or\ Metadata)$
if *SSEforIoT* == *True* **then**
  $image\_display \Leftarrow imageDisplay(search\ results)$
  $image\_select \Leftarrow imageClick(choose\ image\_display)$
  $secureImage\_retrieved \Leftarrow imageRetrieved\_from\_cloud(image\_select)$
  $image\_decrypt \Leftarrow AES\_decrypt(secureImage\_retrieved)$
  $image\_Urldecrypt \Leftarrow AES\_decrypt(secureImage\_retrieved\_url)$
  $Original\_image \Leftarrow image\_decrypted\ andimage\_Urldecrypted$
else
  $Empty\_list \Leftarrow search\_image$

IV. Results and Discussion

The *SSEforIoT* app was developed in Android Studio 4.2 using Java, targeting devices with a minimum SDK of API 28 (Android 9.0 Pie). CPU Profiler measured memory, CPU, and network usage for key functions. The app uses Firebase Storage and Realtime Database to simulate cloud-based IoMT data storage.

This section evaluates the proposed SSE scheme for securing digital images in the cloud. Both server-side and client-side encryption are tested to assess resource requirements and feasibility in resource-constrained IoT environments.

1) Server-side security: The *SSEforIoT* application integrates multiple layers of server-side security features, including access control and encryption of data at rest, in transit, and in use. User authentication is managed through Firebase Authentication, supporting login via email/password, Google, or Microsoft accounts—all of which are based on OAuth 2.0 and OpenID Connect. These authentication providers enhance security by protecting users' long-term credentials and identity, eliminating the need for the mobile application itself to store sensitive credentials. Instead, the client is issued an access token, which is used to obtain authorisation, thereby separating the role of the client from that of the resource owner.

In addition to authentication, *SSEforIoT* enforces secure access control for stored data through Firebase Realtime Database and Cloud Storage Security Rules. When a user successfully authenticates via Firebase Authentication, the $request.auth$ variable in Cloud Storage Security Rules becomes an object containing the user's unique ID ($request.auth.uid$) and all relevant user information included in the authentication token ($request.auth.token$). This mechanism enables fine-grained, per-user access control, ensuring that only authorised users can access their data. If a user is not authenticated, $request.auth$ defaults to null, and access is denied [25].

Firebase stores records like name, annotations, timestamp, userId, encryptedUrl. The app first queries or fetches records by user scope (for example, orderByChild("userId").equalTo(currentUid) or a simple collection read), then filters on-device by keyword across name/annotations. The encryptedUrl is not queryable; it is decrypted locally only after a user selects a result.

"Table I" gives an overview of the main similarities and differences between authentication through Firebase Email/Password, Google and Microsoft.

In addition, all cloud providers implement 256-bit AES to protect the data at REST and Transport Layer Security (TLS) for data in TRANSIT. *SSEforIoT* on the other hand implements client-side encryption which secures the data in transit by applying encryption before it is transferred into storage. Firebase, AWS and Azure also offer similar centralised key management services to facilitate the creation and management of cryptographic keys. Google Transpiler and Microsoft SEAL both offer symmetric encryption libraries which incorporate the BFV and the CKKS encryption algorithms.

2) Client-Side Encryption: The efficiency of client-side encryption in *SSEforIoT* was evaluated by applying AES-GCM encryption with locally stored keys to the image URL during the image upload process. The image URL, stored as part of the database record, serves as the critical link between the image file in cloud storage and the corresponding metadata in the cloud database.

As illustrated in "Fig. 5 (a)", once the URL is encrypted, images are no longer directly populated in the gallery view. Nevertheless, users can still execute search queries and select items to open the detailed view. In this view, the image initially remains hidden, and the image location is shown in its encrypted form. When the user clicks the "Decrypt Image Location" button, decryption is performed—provided the user possesses the appropriate encryption keys. At that point, the image is rendered in the view, and the plaintext URL is

TABLE I
OVERVIEW OF AUTHENTICATION THROUGH FIREBASE EMAIL/PASSWORD, GOOGLE AND MICROSOFT.

Authentication Provider	Credentials stored	OAuth 2.0 & OpenID	Two-step Verification	Email Verif ication	Rules for Password Creation	Implementation
Firebase Email/Password	Yes	No	No	No	No	Easy
Google	No	Yes	No	Yes	Yes	Moderate
Microsoft	No	Yes	Yes	Yes	Yes	Difficult

displayed as shown in "Fig. 5 (b)". Importantly, throughout this entire process, the URL stored in the database record remains encrypted, thereby preserving data confidentiality.

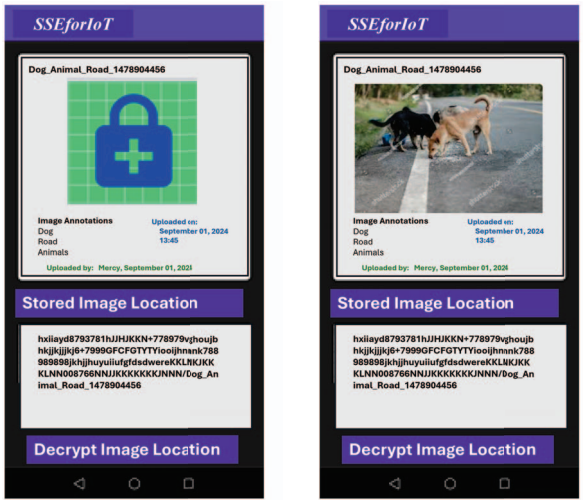
V. PERFORMANCE EVALUATION

This section evaluates the performance and robustness of the proposed SSE scheme, focusing on efficiency (time and resource use), accuracy, and reliability of retrieving secured cloud images, and security against potential threats.

A. *Efficiency Analysis*

Several tests have been undertaken to ensure the efficiency of the scheme in terms of time, as well as the resources, such as CPU, memory and network requirements for the client device.

1) ***Test Case 1****:* In test case 1, the efficiency of the scheme is tested by measuring the time required to complete the process of sending data to Cloud storage with and without client-side encryption. The calculation of the time needed to complete an upload to the Cloud database is coded into the activity. Initially, a sample of 200 images was uploaded through the *SSEforIoT* app. To increase the reliability of the results, pictures of different sized were selected. The smallest image had a size of 14.86KB and had a timeToStore of 1810 milliseconds, while the largest picture was 5.70MB in size and required 6596 milliseconds timeToStore. The average size of the images used in the test sample was 899.67kB, while the average timeToStore was 3248.2 milliseconds. This seems



(*a*) Encrypted Image detail view (*b*) Plaintext Image detail view

Fig. 5. Visual Results of Proposed SSE on Stored Images.

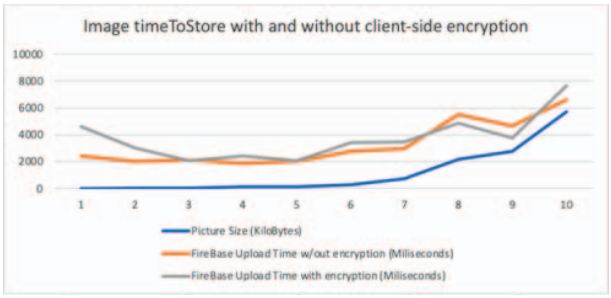


Fig. 6. Comparison of timeToStore with and without client-side encryption.

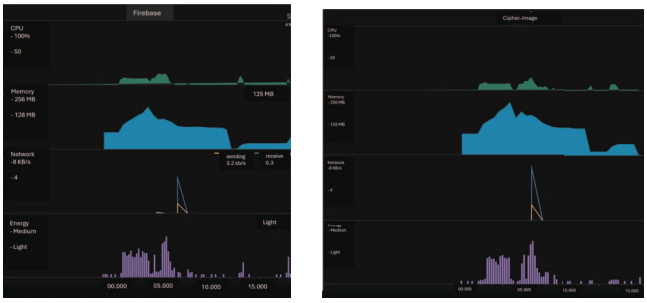
to suggest that the estimated timeToStore is roughly 3.61 milliseconds per kilobyte.

In order to compare different cloud storage solutions, a similar exercise was executed on Microsoft Azure Blob storage. The results of this test also validate that larger pictures take a longer time to upload. "Table II" compares TimeToStore for Firebase uploads without client-side encryption, and for MS Azure using 10 images from the 200 uploaded images. Uploads to MS Azure were faster, averaging 940.5 ms, compared to 3272.2 ms for Firebase.

TABLE II
COMPARATIVE UPLOAD TIME OF IMAGE DATASETS WITHOUT CLIENT-SIDE ENCRYPTION FOR FIREBASE AND AZURE.

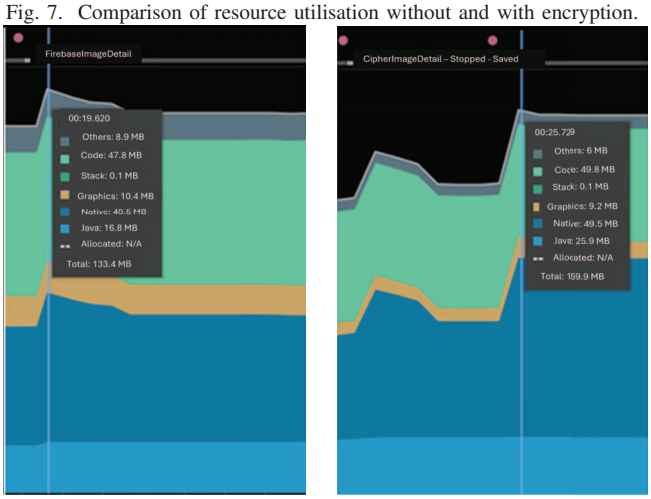
S/N	Image Name	Picture Size (Kilobytes)	Firebase Upload Time (Milliseconds)	MS Azure Upload Time (Milliseconds)
1	Stamen_Botany	16.53	2418	995.00
2	Fly_Insect_Animal	27.27	2007	509.00
3	Meerkat_Animal	28.47	2149	547.00
4	Belguim_Tourism	121.42	1880	531.00
5	Iceland_Arctic	143.01	2017	321.00
6	Lions_Couple	301.84	2548	653.00
7	Boy_Teddybear	736.12	2956	916.00
8	Gum_Water_Bird	2170	5493	1334.00
9	Turtles_Animal	2790	4658	1202.00
10	Lavenda_Flower	5700	6596	2397.00
	Total	**12034.66**	**32722**	**9405**
	Averages	**1203.466**	**3272.2**	**940.5**

To assess the impact of client-side AES encryption, the test was repeated with sample images. "Fig. 6" shows that encryption increases upload time by less than 500 ms, indicating minimal effect on Firebase TimeToStore. Despite the small sample size, this provides insight into cloud upload performance dynamics.



(a) Without encryption (b) With encryption

Fig. 7. Comparison of resource utilisation without and with encryption.



(a) Plaintext url (b) Encrypted url

Fig. 8. Peak in memory utilisation comparison without and with encryption.

*2) **Test Case 2**:* To compare efficiency with and without local encryption, the Android CPU profiler measured resource use from gallery access to Firebase upload. "Fig. 7(a) & (b)" shows minimal overhead from encryption: with client-side encryption, CPU peaked at 26%, memory at 204.3MB, and network at 5.1KB/s received and 1.5KB/s sent; without encryption, CPU was 21%, memory 200.3MB, 5.2KB/s received and 1.4KB/s sent.

"Fig. 8(a) & (b)" compares memory during image display. For unencrypted data, memory averaged 133.4MB (16.8MB Java, 49.5MB Native, 10.4MB Graphics, 47.8MB Code). For encrypted data, the peak occurred at decryption with 169.9MB (25.9MB Java, 78.9MB Native, 9.9MB Graphics, 49.8MB Code). Overall, results show negligible memory differences, confirming resource use is acceptable even for constrained IoMT devices.

B. Security Analysis

The robustness of the proposed SSE scheme was evaluated against common attacks:

*1) **Authentication Attack**:* Strong authentication is vital, as compromised credentials can expose all data. Appthority reported thousands of apps leaking data due to insecure Firebase use [26]. In this work, authentication is central: when users log in to *SSEforIoT* with Google credentials, OpenID verifies them in the background and issues a JSON Web Token (JWT) for access, as shown in 'Fig. 3 (a)". Microsoft login uses the MSAL library, which supports Azure AD and Microsoft Accounts, adding an extra verification step. In Android Studio, MSAL and Microsoft Graph are configured separately.

*2) **Man-in-the-Middle Attack (MITM)**:* Here an adversary intercepts data in transit. *SSEforIoT* prevents this by encrypting data client-side before cloud upload and keeping it encrypted throughout storage. Only users with AES keys can decrypt image URLs, protecting against MITM.

*3) **Replay Attack**:* In replay attacks, intercepted data is resent. Countermeasures include tagging encrypted images with timestamps or nonces. Our design records annotations, upload time, and user UID, enabling detection and rejection of replayed submissions.

*4) **Encryption Strength Analysis**:* The implemented scheme in *SSEforIoT* employs AES-256 in GCM mode for client-side encryption, which offers a 256-bit key size, providing an estimated 2^{256} possible key combinations, making brute-force attacks computationally infeasible under current technology. Each encryption operation generates a 128-bit authentication tag to verify data integrity and authenticity. The ciphertext expansion in AES-GCM is limited to the size of the authentication tag (16 bytes), keeping ciphertexts compact, which is well suited for IoT devices with storage and bandwidth constraints. Formally, the scheme provides:

1) **Confidentiality:** 256-bit symmetric security level (AES-256).
2) **Integrity and Authenticity:** 128-bit GCM tag ensuring protection against tampering and replay.
3) **Ciphertext Length:** $|C| = |P| + 16$ bytes (for authentication tag).

The security analysis demonstrates that the proposed SSE using AES-GCM is secured against brute-force, replay, and MITM attacks.

C. Comparative Analysis

The security and encryption methods of well-known cloud providers, such as Google Firebase, Amazon Web Services, and Microsoft Azure, are evaluated here. The *SSEforIoT* application uses Firebase for file storage and database management. Cloud security generally follows a shared responsibility model: providers secure infrastructure and configurations, while customers manage their own OS, databases, and applications, especially in PaaS or IaaS models. "Table III" compares security features of these providers and *SSEforIoT* regarding access control and encryption during storage, transit, and use, including symmetric encryption capabilities.

VI. Conclusions

As IoT systems expand globally, the massive amount of data from resource-limited devices must be transferred to cloud servers for processing and storage. With the rise of compact imaging and video devices, there is a shift from traditional IoT to multimedia-focused IoMT networks, which increases complexity due to the heterogeneous nature of data and the diversity of devices. This research work aimed to design, implement, and evaluate an SSE scheme. An assessment

TABLE III
OVERVIEW OF AUTHENTICATION THROUGH FIREBASE EMAIL/PASSWORD, GOOGLE, AND MICROSOFT.

	Access Control	Data Encryption at REST	Data Encryption in Transit	Centralised Key Management	Symmetric Encryption
Google Firebase	Firebase AIM & Security Rules	256-bit AES	TLS – HTTPS	Android Keystore	Google Transpiler
Amazon Web Services (AWS)	AWS IAM, ACLs & Bucket Policies	256-bit AES	TLS – HTTPS, CloudFront	AWS KMS	N/A
Microsoft Azure	Azure AIM & RBAC	256-bit AES	TLS – HTTPS, SMB 3.0	Azure Key Vault	Microsoft SEAL
SSEforIoT	**Firebase AIM & Security Security Rules**	**256-bit AES**	**TLS – HTTPS, Client-side Encryption**	**Android Keystore**	**AES Client-side encryption**

revealed that the protections offered by Google Firebase, Amazon Web Services, and Microsoft Azure are similar. To boost privacy, client-side encryption using AES-GCM was added in the *SSEforIoT* environment to encrypt image URLs. Its feasibility was tested in terms of storage time and resource use. Results showed minimal additional time for uploading encrypted images to Firebase storage. Memory, CPU, network, and power consumption grew slightly but remained manageable for resource-constrained IoMT devices. The scheme allows search operations directly on encrypted data without decryption, maintaining privacy. The homomorphic search can be repeated on ciphertexts without exposing data. Using AES in GCM mode avoids the large ciphertexts and noise issues typically associated with SE, which provides an efficient and practical solution for secure image storage and retrieval in IoT environments.

REFERENCES

[1] W. Stallings, B. L. Stallings, and Brown, *Computer Security: Principles and Practice*, 4th ed. Pearson, 2018.

[2] I. Aribilola, M. N. Asghar, N. Kanwal, M. Fleury, and B. Lee, "Securecam: Selective detection and encryption enabled application for dynamic camera surveillance videos," *IEEE Transactions on Consumer Electronics*, vol. 69, no. 2, pp. 156–169, May 2023.

[3] M. Milenkovic, *Internet of Things: Concepts and System Design*. Springer International Publishing, 2020.

[4] A. Shifa, M. N. Asghar, S. Noor, N. Gohar, and M. Fleury, "Lightweight cipher for h.264 videos in the internet of multimedia things with encryption space ratio diagnostics," *Sensors*, vol. 19, no. 5, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/5/1228

[5] I. Aribilola, S. H. Alsamhi, J. G. Breslin, and M. N. Asghar, "Supor: A lightweight stream cipher for confidentiality and attack-resilient visual data security in iot," *International Journal of Critical Infrastructure Protection*, p. 100786, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1874548225000472

[6] B. Li and D. Micciancio, "On the security of homomorphic encryption on approximate numbers," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12696 LNCS:648–677*, 2021.

[7] M. Alloghani, M. Alani, A.-J. M, B. D, M. T, H. J, J. A, and A., "A systematic review on the status and progress of homomorphic encryption technologies," *Journal of Information Security and Applications*, vol. 48, no. 102362, 2019.

[8] H. Tseng, C. Lee, C. Lin, and P. Chou, "Iot metadata creation system for mobile images and its applications," in *Proceedings - 11th IEEE International Symposium on Service-Oriented System Engineering, SOSE 2017:63–68*, 2017.

[9] N. Woods and C. Robert, "Encapsulation of image metadata for ease of retrieval and mobility," *Applied Computer Science*, vol. 15, p. 62–73, 2019.

[10] G. Wilkinson, T. Bartindale, T. Nappey, M. Evans, P. Wright, and P. Olivier, "Media of things: Supporting the production of metadata rich media through iot sensing," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, 2018.

[11] O. Ali, M. Ishak, and M. Bhatti, "Emerging iot domains, current standings and open research challenges: a review," *PeerJ Computer Science*, vol. 7, no. 1–49, p. 10 7717 – 659 –8, 2021.

[12] A. Shifa, M. Asghar, M. Fleury, N. Kanwal, M. Ansari, B. Lee, M. Herbst, and Y. Qiao, "Mulvis: Multi-level encryption based security system for surveillance videos," *IEEE Access*, vol. 8, p. 177131–177155, 2020.

[13] H. Liang, J. Wu, X. Zheng, M. Zhang, J. Li, and A. Jolfaei, "Fog-based secure service discovery for internet of multimedia things," in *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 16*, 2020.

[14] H. Yousuf, M. Lahzi, S. Salloum, and K. Shaalan, "Systematic review on fully homomorphic encryption scheme and its application," *Studies in Systems, Decision and Control*, vol. 295, p. 537–551, 2021.

[15] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, p. 120–126, 1978.

[16] F. Armknecht and A. Sadeghi, "A new approach for algebraically homomorphic encryption," 2008, iACR Cryptol. ePrint Arch.:422.

[17] A. Acar, H. Aksu, A. Uluagac, and M. Conti, "survey on homomorphic encryption schemes: Theory and implementation," *ACM Computing Surveys*, vol. 51, 2018.

[18] G. Kalyani and S. Chaudhari, "An efficient approach for enhancing security in internet of things using the optimum authentication key," *International Journal of Computers and Applications*, vol. 42, p. 306–314, 2019.

[19] F. Li, J. Ma, Y. Miao, X. Liu, J. Ning, and R. H. Deng, "A survey on searchable symmetric encryption," *ACM Comput. Surv.*, vol. 56, no. 5, Nov. 2023. [Online]. Available: https://doi.org/10.1145/3617991

[20] W. He, Y. Zhang, and Y. Li, "Fast, searchable, symmetric encryption scheme supporting ranked search," *Symmetry*, vol. 14, no. 5, 2022. [Online]. Available: https://www.mdpi.com/2073-8994/14/5/1029

[21] M. M. Silveira, D. S. Silva, S. J. R. Rodriguez, and R. L. Gomes, "Searchable symmetric encryption for private data protection in cloud environments," in *Proceedings of the 11th Latin-American Symposium on Dependable Computing*. New York, NY, USA: Association for Computing Machinery, 2023, p. 95–98. [Online]. Available: https://doi.org/10.1145/3569902.3570171

[22] A. El-Yahyaoui and M. El Kettani, "A noise-free homomorphic evaluation of the aes circuits to optimize secure big data storage in cloud computing," *Lecture Notes in Networks and Systems*, vol. 37, p. 420–431, 2017.

[23] K. Lauter, M. Naehrig, and V. Vaikuntanathan, "Can homomorphic encryption be practical?" in *Proceedings of the 3rd ACM workshop on Cloud computing security workshop - CCSW '11*, 2011.

[24] D. McGrew and J. Viega, "The galois/counter mode of operation (gcm). undefined," 2005.

[25] Cobb, "Cobb m.how was google firebase security bypassed?" 2018, available at. [Online]. Available: https://www.techtarget.com/searchsecurity/answer/How-was-Google-Firebase-security-bypassed

[26] A. Conway, "Millions of users' data leaked through misconfigured firebase backends," https://www.xda-developers.com/user-data-leak-misconfigured-firebase-backends/, Jul. 2018, accessed: 2025-9-22.