ELSA: a Keyword-based Searchable Encryption for Cloud-edge assisted Industrial Internet of Things

1st Jawhara Aljabri
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
j.aljabri.1@research.gla.ac.uk

2nd Anna Lito Michala
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
annalito.michala@glasgow.ac.uk

3rd Jeremy Singer
School of Computing Science
University of Glasgow
Glasgow, United Kingdom
jeremy.singer@glasgow.ac.uk

***Abstract*—The Industrial Internet of Things (IIoT) plays a powerful role in smart manufacturing by performing real-time analysis for large volumes of data. In addition, IIoT systems can monitor several factors, such as data accuracy, network bandwidth and operations latency. To perform these operations securely and in a privacy-preserving manner, one solution is to use cryptographic primitives. However, most cryptographic solutions add performance overhead causing latency. In this paper, we propose an Edge Lightweight Searchable Attribute-based encryption system (ELSA). ELSA leverages the cloud-edge architecture to improve search time beyond the state-of-the-art. The main contributions of this paper are as follows. First, we present an untrusted cloud/trusted edge architecture, which optimises the efficiency of data processing and decision making in the IIoT context. Second, we enhance search performance over current state-of-the-art (LSABE-MA) by an order of magnitude. We achieve this by improving the organisation of the data to provide better than linear search performance. We leverage the edge server to cluster data indices by keyword and introduce a query optimiser. The query optimiser uses k-means clustering to improve the efficiency of range queries, removing the need for linear search. In addition, we achieve this without sacrificing accuracy over the results.**

***Index Terms*—Industrial Internet of Things, Keyword-based searchable encryption,Edge-Cloud architecture,smart manufacturing.**

I. Introduction

The transformation of industrial manufacturing needs the support of data and systems technology to enhance the efficiency of manufacturing operations, improve product quality, and support smart decisions. This transformation through the Internet of Things has demonstrated significant improvements in sectors such as smart cities, smart homes and healthcare [1]. As a result, introducing the IoT to the industrial sector led to a new industrial revolution; Industry 4.0. A term *Industrial Internet of Things* (IIoT) has been used to collectively refer to proposed IoT solutions in this space [2], [3]. Khan et al. [4] have predicted that IIoT will have 75 billion deployed devices by 2025 and Accenture has predicted that the IIoT would add $14.2 trillion US dollars to the global economy by 2030 [5].

A typical use case for IIoT is remote maintenance, product traceability to manage the product life cycle and service optimization [6]. In this case, factories may need to share data with third parties such as insurance, consulting companies, customers and employees, requiring different users to see different levels of information. Thus, accessing all the data might compromise the factory while accessing some higher level of information might be beneficial [7]. To provide accessibility, data from multiple locations are collected and analysed on the cloud, often provided by third-party Infrastructure as-a-Service (IaaS) providers. Relying on IaaS providers raises concerns of data confidentiality, integrity, privacy and security. Cryptographic primitives such as AES and RSA are used in various implementations [8]. Often efforts concentrate on protecting the data-at-rest [9], [10]. This goal is challenging as it requires processing the encrypted data (not the plaintext) remotely in the cloud often through the use of searchable encryption (SE) [11]. However, intermediate steps of partial decryption often execute in the untrusted cloud environment. Thus, implementing an SE method with minimal leakage of unencrypted information is desirable. Additionally the search requires data to be in a predefined (partial) order. Moreover, to control and manage data access, IIoT systems must deploy access control (AC) policies [12]. This adds an even more significant challenge [13], [14]. Furthermore the keywords may not relate to the order the data is stored and are embedded in the ciphertext. Finally, all these challenges relate to the computation and power availability of IIoT devices, the real-time requirements imposed by the industrial processes, the complexity of SE and AC algorithms.

A. Contributions

In this work, we introduce a cloud-edge architecture IIoT that utilises keyword-based searchable encryption multi-authority (MA) access control (AC) scheme for IIoT devices assisted by a three-tier edge computing architecture. As accidents of data leakage in cloud storage happen frequently and have been considered as one of the security issues in cloud storage [15], we consider the devices in the cloud to execute in an untrusted environment and only allow it to process encrypted data. We consider the devices executing in the edge 'tier' to execute in a trusted environment. Any processing that involves the raw data or partial decryption executes on the edge server [16], [17]. Our contributions are as follows:

- We introduce an edge lightweight searchable attribute-based encryption system (ELSA), a cloud-edge architec-

ture that optimises the efficiency of data processing and decision making in the IIoT context.
- Our key novelty is the introduction of keyword indices at the trusted edge alongside a query optimiser, which uses a clustering algorithm; this improves the efficiency of range queries, removing the need for linear search.
- We improve search performance relative to state-of-the-art lightweight keyword-based searchable encryption with multi-authority access (LSABE-MA) [18] by an order of magnitude.

B. Organisation

The organisation of this paper is as follows. We briefly describe related work in Section II. We provide a description of LSABE-MA in Section III. ELSA and our optimisations to improve search time are discussed in Section IV. In Section V, we outline our evaluation methodology based on three factors: overhead times, search time, and accuracy. Results are presented in Section VI. Section VII discusses threats to validity. Finally, we present our conclusions in Section VIII.

II. Related Work

The first searchable encryption (SE) scheme was introduced by Song et al. [19] and the first asymmetrical SE scheme was presented by Boneh et al. [20]. Searchable encryption is the most common search method to retrieve files using keywords instead of retrieving all the encrypted files back. The subsequent SE schemes were designed to support a range of properties such as single keyword search [21] and multi-keyword search [22].

Zhou et al. [23] identified schemes that combined Public Key Encryption with Keyword Search (PEKS) with Attribute-Based Encryption (PEKS-ABE) for cloud-based applications. They identified that PEKS-ABE provides efficient data sharing and search capability, though the privacy of user keys needed improvement. However, they do not also apply the work to IIoT. In this case an edge server could provide a more trusted and privacy-preserving method for processing and storage of transactions that involve the use of private user keys.

The following works focus on improving either SE or AC for IIoT environments, but they do not combine them. Chen et al. [24] proposed lightweight searchable encryption for cloud-based IIoT applications with security improvements. Qi et al. [25] improve Ciphertext-Policy Attribute-Based Encryption (CP-ABE) in the following ways:

1) They are using a hybrid cloud infrastructure. They propose a public cloud to store encrypted IoT data and a private cloud to execute CP-ABE tasks over the data.
2) They guarantee data-privacy at the user level against the private cloud. The authors achieved this by proposing two encryption techniques. These techniques work by protecting IoT data privacy at the item level and preventing the user-key leakage problem.
3) They enable the private cloud to execute CP-ABE encryption/decryption tasks in batches, while executing the CP-ABE re-encryption tasks regardless of the size of IoT data. Thus, they improve the performance of IIoT applications.

Chen et al. [24] proposed lightweight searchable encryption for cloud-based IIoT applications with security improvements. To achieve more precise data retrieval, Miao et al. [26] proposed an improved ABE scheme with multi-keyword search to support simultaneous numeric attribute comparison, thereby greatly enhancing the flexibility of ABE encryption in a dynamic IoT environment. Furthermore, attribute-based multi-keyword search schemes were also investigated in [27]. Nevertheless, this CP-ABE scheme inevitably concentrates on the single authority environment in which a central authority (CA) essentially controls all attributes' authorisation. The single authorisation cannot effectively generate and manage the public/secret keys in the IIoT. However, these studies did not improve the bandwidth of data that is outsourced to the cloud, which is important to minimise the computational cost.

Moreover, many extensions of keyword-based searchable encryption work in recent years such as [28], [29]. They achieved decentralization by eliminating the central authority. However, their schemes suffer from high computational overhead and use expensive bilinear pairing operations. Considering the storage space constraint of lightweight IIoT devices, it is a serious issue.

Zhang et al. [18] proposed a lightweight SE-AC scheme by providing lower computational complexity. Moreover, their framework enhanced privacy by preventing leakage during data outsourcing to a cloud server. This scheme provides fine-grained AC, multi-keyword search, lightweight decryption, and a multi authority environment. They provide low latency as well as improved security against the chosen-keyword attack and the chosen plaintext attack. Their LSABE and LSABE-MA schemes can support single keyword and multi-keyword searching while maintaining the lightweight decryption on many practical testing platforms (PC, mobile phone, Single-board Computers). Moreover, their schemes meet the low-latency requirement of IIoT applications. Therefore, their schemes are suitable for practical IIoT environments. However, their work did not consider the accuracy and data bandwidth, which is regarded as requirements of IIoT applications. In addition, the encryption time for their scheme may not be suitable for real-time application requirements. Simultaneously, latency is an important metric in the encryption phase for the real-world IIoT environment. This work is based on sequential search of all the encrypted data records. Thus, searching over the encrypted privacy-sensitive data uploaded to the cloud can introduce latency that does not adhere to the real-time requirements of IIoT. Hence, we identify a deficiency in the searching method and associated time as well as the bandwidth utilised and we aim to improve the overall performance to meet IIoT requirements.

III. Background

We first conduct a reproduction study of LSABE-MA [18] over two parts, a server and a client app. We then enhance

this approach through use of an edge device that runs sketch algorithms and a query optimiser.

For the LSABE-MA scheme, the client app encrypts and transmits the encrypted data to the cloud by performing the following:

- Initialisation: The system is set up by the admin by taking as an input a security parameter κ. The outputs are the master secret key (MSK) and public parameters (PP)

$$(MSK, PP) \leftarrow Setup(\kappa)$$

  The Global Setup generates the global identity GID beside the PP and MSK
- Secret Key Generation for AC: The authority setup algorithm for each authority A_j generates an authority attribute public key $APK_{i,j}$ and attribute secret key $ASK_{i,j}$ for each attribute i. The secret key generation utilizes the master secret key, public parameter, global identity, and attribute secret key as parameters to generate the secret key for each specific authority.

$$SecKeyGene(MSK, i, GID, ASK_{i,j}) \leftarrow SK_{i,GID}$$

- Data Encryption: The data owner extracts the keyword set KW from file M to produce the ciphertext CT, containing the IIOT devices's reading and the encrypted keyword. The encryption process takes the following input: dataset M, access policy (A,q), keyword set KW, PP and the set of attribute public keys $APK_{i,j}$ for relevant authorities, to produce the ciphertext CT, which contains the encrypted secure index I and the encrypted file C_M.

$$Encrypt(M, (A,_P), KW, PP, APK_{i,j}) \leftarrow CT$$

- Data Transmission: the client sends the ciphertext to the cloud (server)
- Searching: the data user generates a trapdoor $T_{KW'}$ by using a set of keywords, PP and SK.
- Transformation Key: The data user also runs the transform key generation function, which takes the SK and a blind value z to generate the transformation key TK. The user then sends the TK with trapdoor query to the cloud.

The second part executes on the server. The server performs the following:

- Receive the encrypted data from the user
- Store the data in a database
- Receive a search trapdoor from the user and perform the search. The search function takes the trapdoor $T_{KW'}$ and the ciphertext as input. If the output of this function is "0", then the data was not found in the database. If the output of this function is "1", the cloud runs the transformation algorithm.

$$Search(CT, T_{KW'}) \leftarrow 0/1$$

- If the attributes included in transformation key TK satisfy the access policy in the ciphertext then the server runs the partial decryption on the result of the query using the transformation key.
- Return the result to the user.

When the user (the client) receives the data, they will decrypt it using the blind value z and the partially decrypted ciphertext to display it as an output.

$$Decrypt(z, CT_{out}) \leftarrow M$$

For these two parts the following features of the LSABE-MA were implemented: setup, key generation, encryption, trapdoor, transform key generation, search, transformation (partially decryption) and decryption.

IV. ELSA

A. Architecture

As illustrated in Fig. 1, our novel ELSA scheme leverages a cloud-edge architecture. We propose a lightweight searchable attribute-based encryption method on the edge. .

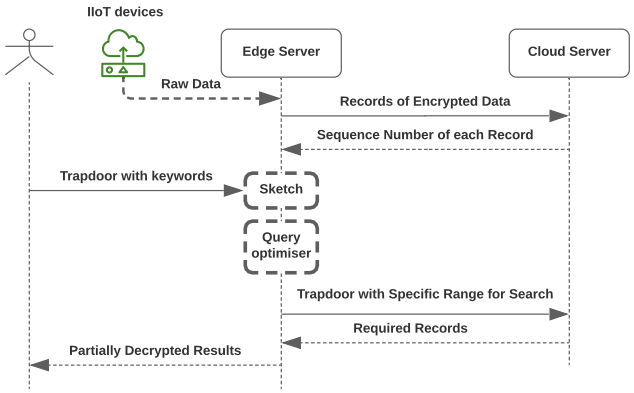

Fig. 1. The sequence diagram for ELSA which demonstrates the interactions between IIoT, Edge, Cloud, and the User.

In the cloud-based search model, query time increases as the number of ciphertexts increase. If a large number of users access the ciphertext at the same time, the server can be blocked or even crash [30]. In LSABE-MA, this is induced by the linear search performed on the cloud server over the entire database. Our proposed optimisations remove the need for a linear search through the introduction of clusters of data points relevant to existing keywords.

Further, a cloud-edge architecture is the most suitable platform for industrial applications. An edge computing entity can optimize the overall system computation, as it can be closely associated with IIoT objects while it can optimise the use of cloud services [31].

The ELSA method presented in Fig. 2, utilises the cloud-edge architecture to process the query over the encrypted data where the edge is the trusted environment. The cloud server is responsible for interacting with the full encrypted dataset and only handles encrypted data (Fig. 2 left). Moreover,

the use of the edge server can reduce the load on cloud communication and meet the IIoT privacy requirements, while reducing the latency for the user to acquire the requested data overall reducing core traffic. The edge (Fig. 2 middle) is responsible for handling incoming requests from the user, eliminating queries that would yield no result (bloom filter) and optimise queries that are propagated to the cloud. It also communicates directly with the IIoT devices and the data owner to establish the AC policies. Any partial decryption happens on the trusted edge server while user keys are only handled in this environment.

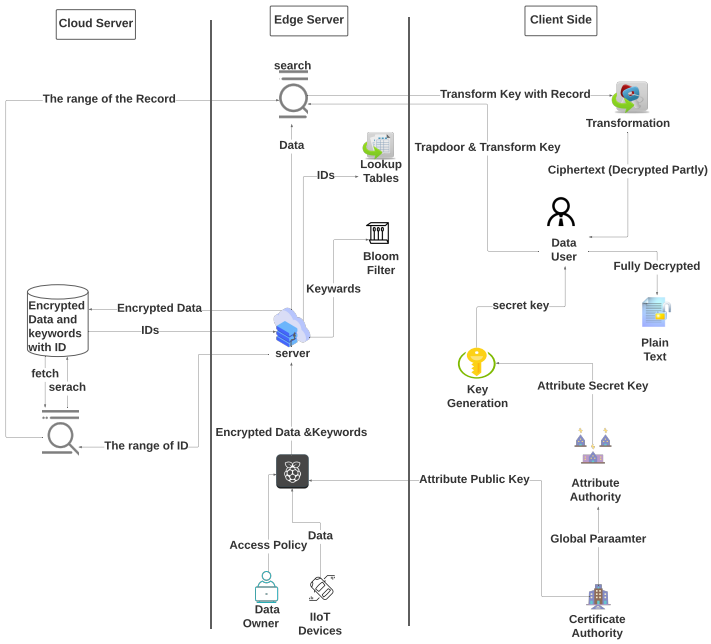

Fig. 2. Proposed ELSA Architecture separating domains of trust coinciding with data residence.

B. Generating Lookup Tables

To improve the performance, we enhance the architecture by introducing the edge server. The edge server is located within the smart-factory and introduces two significant optimisations to reduce latency of the encrypted data search. These are the sketch algorithms (probabilistic data structures), and the query optimiser which we describe further in this section. The workflow of ELSA is divided into 3 phases, as follows:

Phase 1: Generating keywords and ciphertext. The proposed architecture can collect raw data from various IIoT devices. The IIoT devices are connected to the edge Server. Regardless of the keyword generation mechanism (e.g., using feature extraction or thresholds), the edge generates the appropriate keyword set (KW) for each reading. The number of keywords associated with each reading is assumed to be predetermined at this point. For example in this paper we use the following: $\{CO_{2_{high}}$, $CO_{2_{low}}$,$CO_{2_{normal}}$, $humidity_{normal}$, $humidity_{high}$, $humidity_{low}$, $temperature_{normal}$, $temperature_{high}$, $temperature_{low}\}$. Then the edge encrypts the raw data with the keywords and appropriate user credential (CR) and generates a ciphertext (CT) as follows:

$$Encrypt(\{value, KW, sensorID, factoryID, CR\}) \leftarrow CT$$

Phase 2: Building and updating the lookup tables. The edge forwards the encrypted records to the cloud. The cloud then stores the encrypted record and generates a unique ID for each record. A unique ID for each encrypted record is sent back to the edge server. The edge is now responsible for maintaining the lookup tables which are updated whenever a new ID is received. The lookup tables store the ID and the associated keywords. The edge will also store the keywords in a sketch table which is a Bloom-Filter [32]. The sketch algorithm optimises access over summarised data through the use of estimated or probabilistic methods, in this case a Blook filter [33]. The Bloom filter [34] is based on a membership approach to test if an item exists in the stored data or no.The Bloom-filter simply answers the question of whether a specified item exists in the sketch or not. Thus, the sketch is used to determine whether the specified keyword exists in the database or not, which reduces the searching time. Finally the end user (Fig. 2 right) is responsible for generating the initial queries bu generating a trapdoor, executes decryption based to generate plain text if and only if the AC allows them access.

Phase 3: Searching. To initiate a search over the encrypted data the user (or client) will need to creates a request through a trapdoor function parameterised by a keyword and credentials. This process is the trapdoor generation. When a client generates a trapdoor, it must first send the trapdoor along with the required keywords to the edge server. For each keyword, the sketch algorithm is executed to confirm if the keywords were found in the look-up tables. If not found, the edge responds to the client with zero results and indicates that no data is available for the given keyword. If found, the query optimiser is executed. The optimiser decides if a query should be handled as a range or point query. When a client sends a query to the edge, the edge will fetch all the IDs associated with the query (keywords). The optimiser will then decide if the query should be executed as a point or a range query. The IDs are then send to the cloud and the cloud will only search within the specified IDs.

Range query. The optimiser uses the k-means clustering algorithm [35], [36] to cluster the IDs. It will be fastest to search over data if you have clustered data. When using non-clustered data, the server needs to do a sequential search over the entire data. The data storage is the clustered index, so searching by a clustered index eliminates the sequential search.

In ELSA, clustering data using the sklearn kmeans++ algorithm with default hyper-parameter settings. It is the simplest to implement and a lightweight clustering algorithm to deal with massive data reduction to centroids. K-means clustering is a popular unsupervised machine learning algorithm that groups n data items into k clusters. The user specifies the value of k. The centroid of each cluster is used to symbolize it. The first k records are normally utilized as the centroids in

the first iteration. The remaining records are clustered based on the smallest "distance" between each record and each centroid. The centroids are recalculated at the end of the first iteration which using the mean values of the attribute values for the records in each cluster. This step executes after each user query. Once the optimiser clusters the IDs, it sends the lower and upper bound of the IDs (each cluster) to the cloud. For example, a cluster that contains IDs 1,2,5,7 will send the values 1 and 7 to the cloud. Thus, reducing the query size. Note that the number of clusters has to be predefined, otherwise methods such as finding the optimal number of clusters can be adopted.

Point query. In some cases sending the lower and upper bound to the cloud uses more data than sending the IDs themselves. In this case, the optimiser will send the IDs to the cloud. For example, assume we have a query that requires the following IDs, 1,6,20,30. And the optimiser is required to find two clusters, i.e., cluster1 (1-20) and cluster 2 (30). Which means that in a range query, the cloud will search in records from 1-20, and record 30. That is a total of 21 records. In a point query, the server will only search the exact records, i.e., 1,6,20,30 (total of 4 records).

After the cloud searches for the specified IDs (using the LSABE-MA search function), the matched required records are then sent to the edge. Note that in this case, the cloud will search within the specified IDs while ensuring that the user has the appropriate permission (which is embedded in the trapdoor) to access these records. Finally, the edge will receive the records from the cloud, partially decrypt the records and forward them to the authorized users. We discuss the sketch and optimiser components of the search phase in the next section.

C. Searching

As mentioned in the previous section, the edge is using the record IDs to build the lookup tables. Using the IDs provides several advantages. It allows the data to be sorted, it reflects the insertion order, and groups data together into clusters. So, the goal here is to identify a subset of data to search within instead of performing sequential search on the entire data volume.

The query optimiser algorithm 1 enhances the search process on the edge by clustering the data utilising their ID and their associated keywords using the k-means algorithm.

Based on the density of the data around the keyword centroid the optimiser selects if a range of data points or if a single data point corresponds to the user query. A range query is executed when there are several data points in a dense cluster around the requested keyword. A point query (or a set of point queries) is executed when there are few and space data points corresponding to the requested keyword. The point query is used to return unique records of required data and avoid returning unrequired records from the cloud within a set range. The range query is used to particularly when dense data is to be return that were inserted in the database in a sequential range of. This avoids sequentially searching through all saved records.

Algorithm 1 Query Optimiser

Require: $numOfClusters$, IDs - The IDs to cluster
Ensure: Clusters ranges or IDs
1: **function** OPTIMISE($numOfClusters, IDs$)
2: $clusters \leftarrow kmeanCluster(numOfClusters, IDs)$
3: $itemsCount \leftarrow count(numOfClusters)$
4: $idsCount \leftarrow count(IDs)$
5: **if** $itemsCount \geq idsCount$ **then**
6: **return** $upperAndLowerBounds(clusters)$
7: **else**
8: **return** IDs

Moreover, as illustrated in Fig. 2, the edge server utilises a sketch algorithm. The sketch algorithm we are using in this case is the Bloom filter [34]. The Bloom filter works as follows: hash the key to be searched, then check all of the resulting array's index places. Return false if any bit is zero , indicating that the key is "missing". Return true if any bit is one, indicating that the key is "found". This algorithm reduces the overhead on cloud, in particular when the user runs a query that would return no results, while the size of this sketch on the edge is 158 bytes. Thus being very efficient in both execution time and resource use. However, the Bloom filter can introduce imprecision with the probability of a false positive being 0.01 in this case. If the number of unique keywords increase then the the number of hash function and the size of bloom filter will be increase. To calculate the bloom filter in ELSA, the number of unique keywords is nine, the probability of false positive is 0.01 and the number of hash function is one. So, the size of bloom filter is 100 Bytes. This step can eliminate queries that would potentially return zero results. Thus, ELSA eliminates redundant queries from being propagated to the cloud and causing a linear search over the full database only to return an empty set of results. Thus a false positive would be initiating such a search which is not detrimental to returning the correct response to the user. It would only add some amount of unnecessary processing.

V. EVALUATION

This section describes the evaluation setup, evaluation criteria, and presents the evaluation results. We evaluate ELSA (proposed method) by comparing its performance to the LSABE-MA scheme through reproducing the work presented in [18].

A. Evaluation Setup

1) Implementation and Dataset: In our evaluation, we first reproduced the state-of-the-art lightweight keyword-based searchable encryption scheme LSABE-MA [18]. The LSABE-ME [18] scheme and ELSA were implemented in Python 3.7. For the dataset, we used three different cases of data based on the percentage of representation for each keyword. These are the sparse dataset case, the medium density case and the dense case. The data consists of 200,000 unique temperature (temp), co_2, and humidity (hum) values. For evaluation purposes, we considered the $\{CO_{2_{high}}$,

$CO_{2_{low}}$,$CO_{2_{normal}}$, $humidity_{normal}$, $humidity_{high}$, $humidity_{low}$, $temperature_{normal}$, $temperature_{high}$, $temperature_{low}$} values as the keywords. This data is categorical, based on threshold values for ranges. The results are calculated by taking the average of 1000 runs for the three different cases. The dataset was synthetically generated as presented in Algorithm 2. For sparse data a probability of appearance was used to generate more data points associated with one keyword and very few data points associated with another. The latter being a more cumbersome task for the linear search approach in the LSABE-MA scheme.

Algorithm 2 Dataset generator

Require: len, $keywords$ - The keywords to associate with len data points, $probabilities$ - percentage of representation for each keyword

Ensure: dataset

1: **function** GENERATE(len, $keywords$, $probabilities$)
2: **while** $len - - \neq 0$ **do**
3: $value \leftarrow uniform\_rand()$
4: $keyword \leftarrow get\_next(keywords, probabilities)$
5: $datapoint \leftarrow \{value, keyword\}$
6: **return** $dataset$

The specific trapdoor generated for the evaluation in each case was constant and used in both the LSABE-MA and our proposed method ELSA.

2) Architecture: Fig. 3 illustrates the experimental setup used for LSABE-MA. We separated this scheme into two parts client and server. We run the client application on an edge device with Intel 2.3 GHz Core i9 processor and 16GB RAM. We deployed the server code on a docker container hosted on a DigitalOcean cloud provider (located in the UK). The plan for the cloud provider was CPU-Optimised, with 1 dedicated CPU, 2-32 vCPUs, 50 GB Memory, 2GB RAM/CPU and 2TB Bandwidth.

The following describes the procedure involved in evaluating LSABE-MA and our scheme (ELSA):

Setup and secret key generation - Client (1-2). The client setup and generates secret keys.

Encryption - Client (3). The client encrypts the sensor readings along with the appropriate keywords. The client then sends the encrypted data to the cloud, storing it on the cloud (step 4).

Searching - Client (5-6). The client generates trapdoors for the required keywords (query). Also, the client generates a transformation key used by the cloud to transform (partially decrypt) the encrypted results before sending them to the client.

Searching - Server (7-8). The server receives the trapdoor and transformation key and performs a search on the encrypted data. The results are then transformed (step 8) and returned to the client.

Decryption- Client (9). The client decrypts the data using the secret key only.

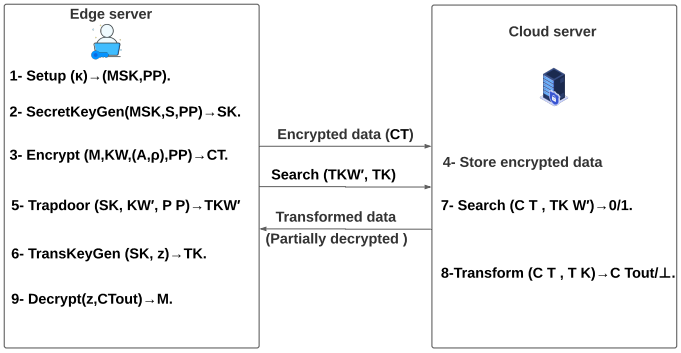

Fig. 3. Evaluation architect of LSABE-MA where the user is directly requesting information from the cloud using the ELSA edge Server as their end-point machine.

B. Criteria

As our goal is to develop a secure, fast and accurate solution for IIoT data. Thus, we focus on the following evaluation criteria:

Performance. We evaluate the performance of LSABE-MA and our method ELSA by measuring the execution time for various functions. Specifically we measure the execution of the key generation, encryption, search and decryption.

Accuracy. We measure the accuracy of the query by comparing the encrypted results of ELSA method and LSABE-MA against a plaintext version. We use the well defined Precision and Recall metrics [37] to measure the accuracy of the search. *Precision* is the percentage of relevant records among the total retrieved records, which is defined as:

$$Precision = \frac{RRT}{RRT - IRT} \times 100\%$$

Recall on the other hand is the percentage of relevant records in relation to the correct records (in the database), which is defined as:

$$Recall = \frac{RRT}{RRT - RRNT} \times 100\%$$

where:

- RRT = Number Of Relevant Records Retrieved
- IRT = Number Of Irrelevant Records
- $RRNT$ = Number Of Relevant Records NOT Retrieved

VI. RESULTS

A. Secret Key Generation

Fig. 4 demonstrates the key generation time for ELSA method and LSABE-MA by measuring the execution time in seconds. For both schemes, we measured the execution time for the function $(\mathrm{MSK}, i, \mathrm{PP}, \mathrm{GID}, \mathrm{ASK}_{i,j}) \rightarrow \mathrm{SK}_{i,\mathrm{GID}}$. The results shows that there is no significant difference between ELSA method and LSABE-MA demonstrating that adding the edge server has no detrimental effect to the overall performance.

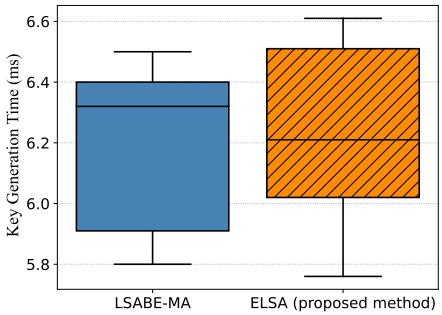

Fig. 4. Key Generation Time measured over 1000 repetitions for each case, reported collectively.

B. Encryption and Decryption

Fig. 5 and Fig. 6 show the encryption and decryption time respectively for both our scheme and LSABE-MA. We measure the execution time for both cases in terms of the function $(M, (A, \rho), \mathrm{KW}, \mathrm{PP}, \{\mathrm{APK}_{i,j}\}) \rightarrow \mathrm{CT}$. Decryption time for both approaches were measured using the same decryption function $(z, \mathrm{CT}_{\mathrm{out}}) \rightarrow M$. Again the introduction of the edge server has caused no detriment to the overall performance.

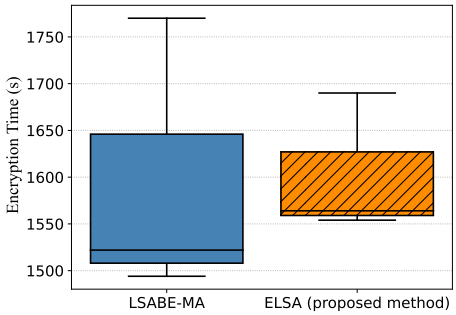

Fig. 5. Encryption Time for the full dataset measured over 1000 repetitions for each case, reported collectively.

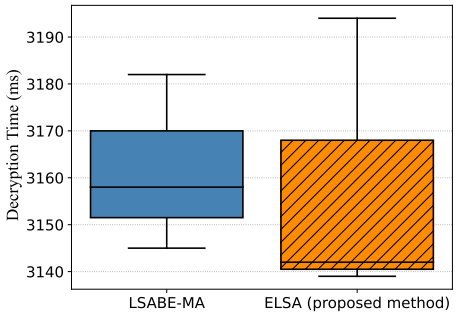

Fig. 6. Decryption Time of the returned query results measured over 1000 repetitions for each case, reported collectively. The results contain 100,000 datapoints in the dense case, 22,000 datapoints in the medium sparsity case and 10,000 in the sparse case.

C. Searching

To further evaluate the scalability of our approach, we measure the search time using various density datasets (shown in Fig. 7). The searched keyword represents 50% of the data size in the Dense Data, 11% in the medium density and 5% in the sparse data. As shown in Fig. 7, the searching time for LSABE-MA is consistent in all three cases, i.e., $386s$. ELSA attains $214s$ search time for the dense data, which is $1.8\times$ faster than LSABE-MA. ELSA has a $43s$ and $28s$ search time for the medium and sparse data, respectively. To conclude, ELSA achieves $1.8 - 14\times$ better performance than LSABE-MA. As the sparsity of the data increases the benefit of non-linear search is becoming apparent. In a dense dataset where the keyword repeats very often the search optimisation still returns a large range to be searched over. However, as the keyword appears less frequently in the sparse case, the smaller range queries combined with point queries significantly boost performance.

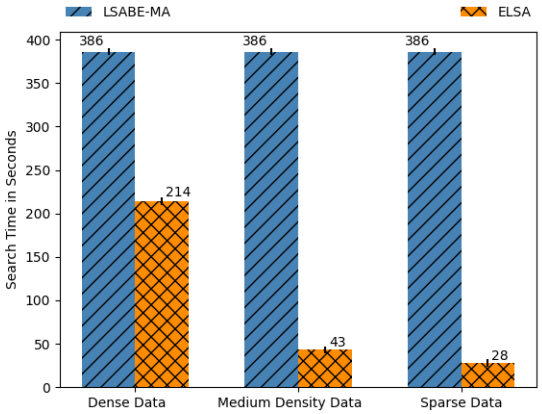

Fig. 7. Search time for three different cases measured at 1000 repetitions per case.

D. Edge and server operations and overhead times for ELSA

In our proposed method ELSA, the encryption process has one additional step. This additional step is to store data in the lookup table on the edge. So, the overhead of this step is illustrated in Table I as load data time. It is evident that this time is insignificant compared to the overall execution time and did not affect the user because of the computational capacity available to our edge server.

In addition, we have two operations in the search process: sketch and clustering algorithms. The sum of the execution time for both these operations are less than 0.1 ms as shown in Table I once again being insignificant compared to the overall execution time.

However, as ELSA scheme depends on several devices communicating we have further evaluated the execution time of each step in the process to identify bottlenecks and isolate the effect of networking or cloud operations that are beyond our control. In the Fig. 8, we illustrate the overall execution

TABLE I
MEAN OVERHEAD TIMES ON THE EDGE SERVER FOR THREE DIFFERENT CASES.

Factors	Overhead Times for ELSA		
	clustering(ms)	*data load(ms)*	*Sketch(ms)*
Execution Time	0.05	32.71	0.00007

time as well as details of the execution time for the various operations which take place on the edge and Cloud servers. These operations are the following: the total operations time as overall operation, encryption time ,the overall operation of the cloud server, edge storage time, cloud server - communication time, and cloud server - storage time. It is evident that the encryption process is one of the main contributors to the overall latency.

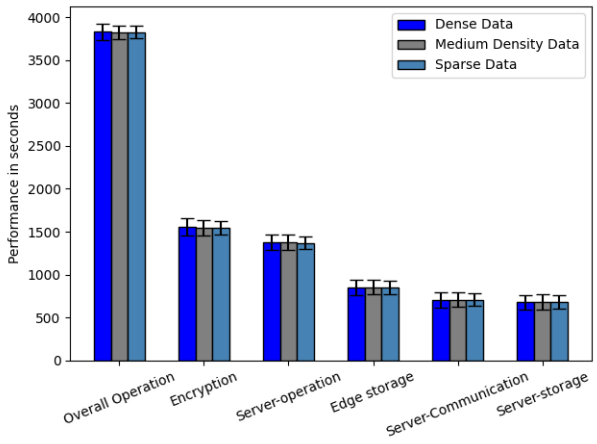

Fig. 8. The Time of Operations in Edge and Server for ELSA measured at 1000 repetitions for each case, reported collectively.

E. Overall Execution Time

Fig.9 compares our ELSA scheme with LSABE-MA scheme in overall execution time. This graph aims to determine the overall effect of search time improvements to the latency experienced by the user from initial query to final result. As demonstrated, the ELSA scheme reduced the overall execution time by 1.21x.

F. Lookup Table Size

To improve search time we have traded-off space used on the edge server to store the lookup tables. To investigate the potential limitation of ELSA we have measured the lookup tables size for our three cases. As presented in Fig. 10, the size of the lookup tables for dense data, medium density data and sparse data are 2.39, 2.15 and 2.25 Megabytes respectively. This did not prove to be a limitation for our use cases but might need to be further investigated in future work to identify any adverse effect on scalability of ELSA for big datasets in the IIoT context.

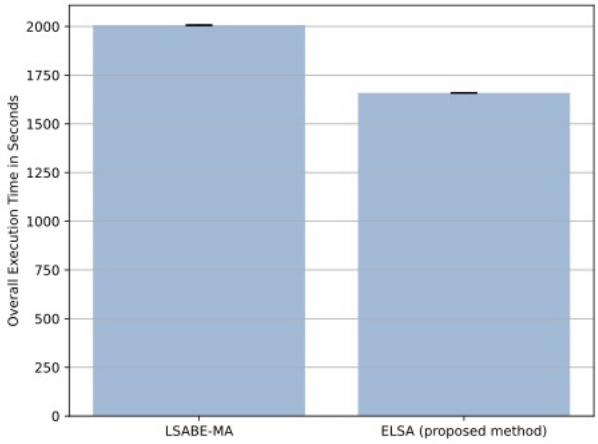

Fig. 9. Overall Execution Time in Seconds measured over 1000 repetitions for each case, reported collectively.

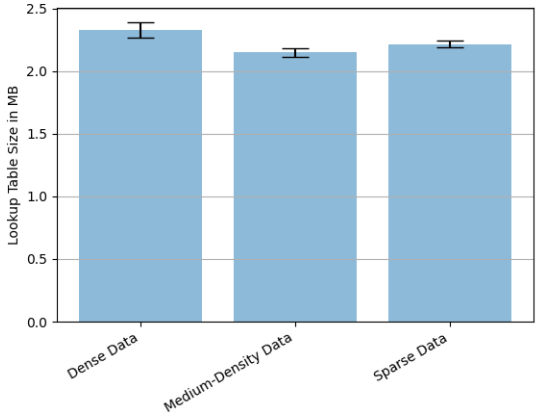

Fig. 10. Lookup Table Size for three different data cases measured at 1000 repetitions per case.

G. Accuracy

To evaluate the accuracy, we performed two queries. The first query contains one keyword: (co2-high), while the second query contains two keywords: (co2-low,co2-normal). Both queries use different keywords. As shown in Table II, both schemes offer 100% precision and recall for both cases. As a result our additional operations and optimisations did not introduce a detrimental effect on the accuracy.

TABLE II
SEARCH PRECISION AND RECALL (ACCURACY) FOR OUR ELSA METHOD AND LSABE-MA DEMONSTRATING LOSSLESS DATABASE PERFORMANCE.

Number of keywords	Proposed ELSA scheme		LSABE-MA	
	1	*2*	*1*	*2*
No. retrieved records	44,000	22,000	44,000	22,000
Precision	100%	100%	100%	100%
Recall	100%	100%	100%	100%

H. Comparison to original LSABE-MA results

In our implementation, we use one attribute, so we compare the results from our re-implementation with the original results from [18] for the one attribute case, which is the first point in their reported graphs. It is worth mentioning that their number of keywords is smaller (set to 5) compared to our re-implementation study.

In terms of encryption time, we find the LSABE-MA scheme takes 9.85s to encrypt the data, while in our Fig. 5 we reported 1550s on average. However in our case we measure the encryption of the full dataset while from the original paper it is unclear if they report the encryption of a single entry (one datapoint being added). If a single datapoint requires 9.85s then our implementation of the encryption for our 200,000 points is orders of magnitude faster. However, we are unable to verify this assumption.

Fig. 6 shows the decryption time of our replicated LSABE-MA, which took 3160 ms on average. In [18], their scheme requires 400 ms to decrypt the ciphertext, which, as they claim, does not significantly increase with the increased number of attributes. However, again it is unclear how many datapoints are returned in each case as it is dependent on the dataset, the keyword appearance frequency and the combination of keywords used for the query. As a result we cannot conduct any reasonable comparison between these two numbers. This is because, the authors of the original study did not explain the complexity of the dataset, the hardware configuration on the cloud and did not disclose the software they used.

From this analysis we can conclude that the large differences come from the number of records encrypted or decrypted in each case. Overall, it is unclear what is the size of the record encrypted in the original study, how large is the database that is searched over, and how many records are returned during the measured decryption step. Further it is unclear how long is the search time in the original study. Thus, no meaningful conclusions can be extrapolated from this comparison.

VII. Threats to Validity

Because of lack of access to the original source code for LSABE-MA, we have created a clean room re-implementation of the scheme. This means that there might be subtle differences to the implementation as initially presented [18]. However, we followed the description presented in the paper and replicated algorithms and formalism to the best of our capacity.

The evaluation is performed on a single fixed deployment scenario. We feel this scenario is sufficiently representative to allow us to hypothesise that the trends identified in the results would generalise.

Further the evaluation focused on a limited set of queries and respective searches. We have tried to cover different data characteristics within those to identify trends in the performance of ELSA. In further work we aim to implement a wider variety of queries to identify potential scalability thresholds and limitations.

Finally, ELSA uses k-means as the method to restrict the scope of the query search cloud-side. However, other clustering methods may apply in this scenario. We intend to investigate these in future work.

VIII. Conclusion

In this paper, we presented a new cloud-edge architecture for Keyword-based Searchable Encryption with an optimised query process. Compared with the state of the art, our scheme ELSA is advantageous with respect to maintaining the encryption and decryption process and achieving more efficient data sharing and data searching. In terms of performance, the experimental results show that our ELSA scheme effectively reduces the search time by up to 14× and overall performance by 1.21×. Compared to LSABE-MA the ELSA scheme significantly reduces the execution time and communication overhead by clustering the required data without sacrificing accuracy. This is achieved through the use of optimisations on the edge server that provide better than linear search performance at a trade-off of utilised storage space.

Acknowledgment

This work was funded under a scholarship TBU331 provided by University of Tabuk in Saudi Arabia.

References

[1] M. A. Khan and K. Salah, "IoT security: Review, blockchain solutions, and open challenges," *Future Generation Computer Systems*, vol. 82, pp. 395–411, 2018. [Online]. Available: https://doi.org/10.1016/j.future.2017.11.022

[2] M. Hermann, T. . Pentek, and B. Otto, "Design Principles for Industrie 4.0 Scenarios: A Literature Review," Tech. Rep. [Online]. Available: www.snom.mb.tu-dortmund.de

[3] P. Mathur, *IoT Machine Learning Applications in Telecom, Energy, and Agriculture*, 2020.

[4] W. Z. Khan, M. H. Rehman, H. M. Zangoti, M. K. Afzal, N. Armi, and K. Salah, "Industrial internet of things: Recent advances, enabling technologies and open challenges," *Computers and Electrical Engineering*, vol. 81, p. 106522, 2020. [Online]. Available: https://doi.org/10.1016/j.compeleceng.2019.106522

[5] Accenture, "Winning with the Industrial Internet of Things," p. 12, 2018.

[6] Y. Yu, R. Chen, H. Li, Y. Li, and A. Tian, "Toward Data Security in Edge Intelligent IIoT," *IEEE Network*, vol. 33, no. 5, pp. 20–26, 2019.

[7] S. T. H. E. Web and F. O. R. A.-r. Devices, "IoT FOR BUSINESS Take Manufacturing ' s Shift Your Manufacturing Shift to Lightspeed to Lightspeed," 2020.

[8] J. Blömer and N. Löken, "Cloud architectures for searchable encryption," in *Proceedings of the 13th International Conference on Availability, Reliability and Security*, 2018, pp. 1–10.

[9] M. S. Hossain and G. Muhammad, "Cloud-assisted Industrial Internet of Things (IIoT) - Enabled framework for health monitoring," *Computer Networks*, vol. 101, pp. 192–202, 2016.

[10] P. P. Jayaraman, X. Yang, A. Yavari, D. Georgakopoulos, and X. Yi, "Privacy preserving Internet of Things: From privacy techniques to a blueprint architecture and efficient implementation," *Future Generation Computer Systems*, vol. 76, pp. 540–549, 2017. [Online]. Available: http://dx.doi.org/10.1016/j.future.2017.03.001

[11] G. S. Poh, J.-J. Chin, W.-C. Yau, K.-K. R. Choo, and M. S. Mohamad, "Searchable symmetric encryption: designs and challenges," *ACM Computing Surveys (CSUR)*, vol. 50, no. 3, pp. 1–37, 2017.

[12] K. Pothong, I. Brass, M. Carr, L. Tanczer, S. Security, R. Nicolescu, B. Craggs, E. Lupu, A. Rashid, C. Maple, S. Wakenshaw, M. Taddeo, J. Lindley, S. Cannizzaro, R. Procter, and P. Coulton, "Editors of the Cybersecurity of the Internet of Things: PETRAS Stream Report 03 Privacy and Trust 05 Adoption and Acceptability," 2019.

[13] E. Sisinni, A. Saifullah, S. Han, U. Jennehag, and M. Gidlund, "Industrial internet of things: Challenges, opportunities, and directions," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4724–4734, 2018.

[14] H. F. Atlam, A. Alenezi, R. K. Hussein, and G. B. Wills, "Computer Network and Information Security," *Computer Network and Information Security*, vol. 1, pp. 26–35, 2018. [Online]. Available: http://www.mecs-press.org/

[15] D. W. Chadwick, W. Fan, G. Costantino, R. De Lemos, F. Di Cerbo, I. Herwono, M. Manea, P. Mori, A. Sajjad, and X.-S. Wang, "A cloud-edge based data security architecture for sharing and analysing cyber threat information," *Future Generation Computer Systems*, vol. 102, pp. 710–722, 2020.

[16] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE internet of things journal*, vol. 3, no. 5, pp. 637–646, 2016.

[17] L. Zhou, M. H. Samavatian, A. Bacha, S. Majumdar, and R. Teodorescu, "Adaptive parallel execution of deep neural networks on heterogeneous edge devices," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing*, 2019, pp. 195–208.

[18] K. Zhang, J. Long, X. Wang, H.-N. Dai, K. Liang, and M. Imran, "Lightweight Searchable Encryption Protocol for Industrial Internet of Things," *IEEE Transactions on Industrial Informatics*, vol. 3203, no. c, pp. 1–1, 2020.

[19] D. X. Song, D. Wagner, and A. Perrig, "Practical techniques for searches on encrypted data," in *Proceeding 2000 IEEE symposium on security and privacy. S&P 2000*. IEEE, 2000, pp. 44–55.

[20] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano, "Public key encryption with keyword search," in *International conference on the theory and applications of cryptographic techniques*. Springer, 2004, pp. 506–522.

[21] R. Chen, Y. Mu, G. Yang, F. Guo, X. Huang, X. Wang, and Y. Wang, "Server-aided public key encryption with keyword search," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 12, pp. 2833–2842, 2016.

[22] H. Li, Y. Yang, T. H. Luan, X. Liang, L. Zhou, and X. S. Shen, "Enabling fine-grained multi-keyword search supporting classified sub-dictionaries over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 13, no. 3, pp. 312–325, 2015.

[23] Y. Zhou, N. Li, Y. Tian, D. An, and L. Wang, "Public key encryption with keyword search in cloud: A survey," *Entropy*, vol. 22, no. 4, pp. 1–24, 2020.

[24] B. Chen, L. Wu, N. Kumar, K.-K. R. Choo, and D. He, "Lightweight Searchable Public-key Encryption with Forward Privacy over IIoT Outsourced Data," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. c, pp. 1–1, 2019.

[25] S. Qi, Y. Lu, W. Wei, and X. Chen, "Efficient Data Access Control with Fine-Grained Data Protection in Cloud-Assisted IIoT," *IEEE Internet of Things Journal*, vol. 4662, no. c, pp. 1–1, 2020.

[26] Y. Miao, J. Ma, X. Liu, X. Li, Z. Liu, and H. Li, "Practical attribute-based multi-keyword search scheme in mobile crowdsourcing," *IEEE Internet of Things Journal*, vol. 5, no. 4, pp. 3008–3018, 2018.

[27] Y. Miao, X. Liu, R. H. Deng, H. Wu, H. Li, J. Li, and D. Wu, "Hybrid keyword-field search with efficient key management for industrial internet of things," *IEEE Transactions on Industrial Informatics*, vol. 15, no. 6, pp. 3206–3217, 2019.

[28] Y. Miao, R. H. Deng, X. Liu, K.-K. R. Choo, H. Wu, and H. Li, "Multi-authority attribute-based keyword search over encrypted cloud data," *IEEE Transactions on Dependable and Secure Computing*, vol. 18, no. 4, pp. 1667–1680, 2019.

[29] Q. Huang, G. Yan, and Y. Yang, "Privacy-preserving traceable attribute-based keyword search in multi-authority medical cloud," *IEEE Transactions on Cloud Computing*, 2021.

[30] "Neat: A resilient deep representational learning for fault detection using acoustic signals in iiot environment."

[31] L. D. Xu and L. Duan, "Big data for cyber physical systems in industry 4.0: a survey," *Enterprise Information Systems*, vol. 13, no. 2, pp. 148–169, 2019.

[32] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Communications of the ACM*, vol. 13, no. 7, pp. 422–426, 1970.

[33] G. Cormode, "Sketch techniques for approximate query processing," *Foundations and Trends in Databases. NOW publishers*, 2011.

[34] L. Liu and M. T. Özsu, *Encyclopedia of database systems*. Springer New York, NY, USA:, 2009, vol. 6.

[35] G. Gan, C. Ma, and J. Wu, *Data clustering: theory, algorithms, and applications*. SIAM, 2020.

[36] L. Rokach, "A survey of clustering algorithms," in *Data mining and knowledge discovery handbook*. Springer, 2009, pp. 269–298.

[37] M. Buckland and F. Gey, "The relationship between recall and precision," *Journal of the American society for information science*, vol. 45, no. 1, pp. 12–19, 1994.