MBNA Publishing House Constanta 2021

Proceedings of the International Scientific Conference SEA-CONF

SEA-CONF PAPER • OPEN ACCESS

Searchable Symmetric Encryption for Cloud Software in Maritime Industry

To cite this article: Marius Iulian MIHAILESCU, Stefania Loredana NITA and Ciprian RACUCIU, Proceedings of the International Scientific Conference SEA-CONF 2021, pg.165-176.

Available online at www.anmb.ro

ISSN: 2457-144X; ISSN-L: 2457-144X

doi: 10.21279/2457-144X-21-022

Searchable Symmetric Encryption for Cloud Software in Maritime Industry

Marius Iulian Mihailescu[1], Stefania Loredana Nita[2], Ciprian Racuciu[3]

[1]*Spiru Haret* University,
Scientific Research Center in Mathematics and Computer Science
[2]Instiute for Computers, Integrated Systems Department
[3]Computer Science Department, *Titu Maiorescu* University

m.mihailescu.mi@spiruharet.ro

Abstract. Due to technological advance of software applications in cloud computing, maritime software industry has a special spot. Starting with 2020, because of the cybersecurity threats and the attacks that took place (e.g., FireEye, SolarWinds), it is very important to redesign and adopt a critical thinking about data privacy, to implement serious security policy and develop high secure software code. The importance of software reliability and secure communication between shore and sea represents an important challenge on which we need to focus. This article comes as an encouragement and its aim is to provide an approach for providing a certain level of confidence of the data that are exchanged between shore-side and sea (e.g., documents, messages). The approach is defined by a searchable symmetric encryption scheme that will allows a party to outsource the documents to another party in a private and dedicated manner. In this way the users will have the possibility to access their data by searching directly over the encrypted data using specific keywords.

1. Introduction

Searchable Encryption (SE) a unique mechanism for allowing encrypted data to be outsourced to an untrusted and potential third-party service provider [26], giving to the user the possibility to search directly over the data that are encrypted [30]. SE represents a particular case of homomorphic encryption [40].

Let's imagine the following general scenario [26], we assume that we have a data owner (DO) that has some documents $D = \{d_1, \dots, d_i\}$ and he wants to store them on a server (S), but in the same time there is another user, let's call him Bob, that he wants to access those documents using a query. If we want to achieve this goal, the DO has to encrypt the documents using $Bob's$ public key and then the DO will store those documents on the server. When Bob wants to access some documents, he will initiate a query (q) based on a keyword (kw) or keywords $K = \{kw_1, \dots, kw_n\}$, he will generate a value using a function called trapdoor (T) based on his kw or K and his private key (prv_k). Once Bob will compute the T function, he will be submitted to the server where there is an algorithm/service that will perform the searching operation and the output will be represented by the documents which meet the criteria given by Bob.

Another scenario, which is more practical [26], can be seen or imagined in a company that wish to develop a software solution that is based on social security numbers (SSNs) that are obtained from the clients. One of the best practice's states that an SSN should ne encrypted if it is not involved in a data

process. At the same time, the employees will have access to the client's SSN's when they will search for a client account – a typical case of plaintext. A normal question is being raised: *what will happen if the employees will not be able to see the SSN in plaintext?* This is the point where a searchable encryption scheme comes and provides solution for the current situation and solve the mystery.

Based on the above scenarios, we can say that SE [3] represents a mechanism that gives the possibility for a user to initiate queries and to search on encrypted content for a specific data/document(s). SE is also a very sensitive mechanism, a wrong implementation or designing process of the algorithms, could be quite devasting for a complex infrastructure (e.g., cloud computing [32], big data [33] etc.). Also, SE has a large area of domains with complex systems and infrastructures in which it can be applied with success, such as healthcare, physics, meteorology, agriculture, IoT etc.

The main objective of our work is to provide a practical approach that it can be used to implement searchable encryption for maritime software application in complex infrastructure, such as cloud computing.

The paper structure. The paper is divided in five sections as follows:

- *Section 1 – Introduction*. The section provides a short introduction for searchable encryption and presents two scenarios where searchable encryption can be applied practically.
- *Section 2 – Searchable encryption components*. In this section the main components of a searchable encryption are presented, a set of questions for implementation are presented with the goal to make the process of implementation much clearer. There are two sub-sections, *2.1. Entities* that describe the role of each entity/participant in the searchable encryption scheme, and *2.2. Types* that present two categories of the searchable encryption techniques (SSE – searchable symmetric encryption and PEKS – public encryption with keywords [7-10]).
- *Section 3 – Basic example of implementation.* The section describes a simple example of implementation, providing an intuitive source code of how an implementation of searching and querying operations should look like.
- *Section 4 – Proposed idea.* The section will discuss in depth the implementation of several algorithms from a searchable encryption scheme.
- *Section 5 – Conclusions.* The section summarizes the main achievements, what challenges were experienced and how they were fixed.

2. Searchable encryption components

We will use the term *entities* to describe the *persons (e.g., Data Owner, Data User, Server etc.)*.

A SE scheme can be characterized by two components: *entities* and *algorithms*. Each of these components have a precise *role* and *objective.* In Section 2.1 we will discuss about the entities, what they represent and what is their main goal/objective in a SE scheme.

2.1. Entities

Starting from the implementation process of a software application or system, there are couple of aspects that are recommended to be clarified before the implementation should start. This being said, the following questions are valid:

- Who will operate the application?
  Answer in accordance with a searchable encryption scheme: Data User, Data Owner.
- Who will provide maintenance?
  Answer in accordance with a searchable encryption scheme: Data Owner, Administrator.
- What types of data are we dealing with and who can access them?
  Answer in accordance with a searchable encryption scheme: encrypted documents that have a keyword (kw) *or keywords* (K) *attached. Also, the keywords are encrypted and stored accordingly.*

- Where the data will be stored?
  Answer in accordance with a searchable encryption scheme: Server, Cloud environment [2], Big Data [19], fog computing, edge computing [18, 20] etc.
- What security mechanisms should be addressed?
  Answer in accordance with a searchable encryption scheme: searchable encryption (SE), symmetric searchable encryption (SSE) or public encryption with keywords (PEKS).
- What infrastructure will be used for the entire software system (e.g., cloud computing, big data, fog computing [38], edge computing etc.)?
  Answer in accordance with a searchable encryption scheme: Any architecture that respects the principles of a secure architecture and provides the proper security mechanisms that will guarantee the confidentiality, integrity and authenticity of the data. A searchable encryption scheme should be flexible, easy to adapt to the software applications and to the types of data and data formats [36].

These questions surround the SE mechanism and the following entities are involved within the entire process of implementation:

- *Data Owner (DO).* As we mentioned in the beginning, a DO has a set of n documents, $n \in D$, where $D = \{d_1, \dots, d_n\}$. Each document is described by a keyword (k) or set of keywords (K). Before proceeding further with storing those documents on the server, the DO will encrypt the documents based on a cryptographic algorithm. We will consider DO as a trusted entity.
- *Data User (DU).* A data user represents a person that is authorized to launch the search process based on queries (Q). This process will use a keyword or set of keywords and generate a trapdoor value. The value will be used to search in the encrypted content. The DU will have the possibility to decrypt the received documents based on the private key.
- *Server (S).* The role of the server is to store the encrypted data and to execute the algorithms for searching using the value from the trapdoor. The server represents a critical point, but for our work we will consider it as a semi-trusted or honest-but-curious. This means that it will execute the search algorithms as instructed and provide an analyze of the data that was given to it.

A very good observation that can be mentioned at this point is based on the fact that the DO can be considered a DU.

2.2. Types

Generally speaking, there are two important categories of searchable encryption schemes that are currently having an applied potential in real case scenarios: *symmetric searchable encryption (SSE)* [21] and *public encryption with keywords (PEKS)* [21].

The first idea of SE scheme for cloud environments was proposed by Song *et al.* in 2000 [5], their contribution representing a pioneering work in the field of provably secure *SE*. In [26], Bosch *et al.* provide a survey about provably secure searchable encryption based on the two main SE techniques: *SSE* and *PEKS.* Their survey is focused on making a comparison between *SE* security, efficiency and functionality. It is easy to follow, being written in a language that makes it easy to understand with a very good foundation for security background. A *SE* are based on different techniques. The techniques covered by their survey are: Searchable Symmetric Encryption (SSE), Public Key with Keyword Search (PEKS), Identity-based Encryption (IBE), Hidden Vector Encryption (HVE), Predicate Encryption (PE), and Inner Product Encryption (IPE).

Another technique that it seems to be powerful enough and has potential in real applications is represent by Multi-Keyword Rank Searchable Encryption (MRSE) [41].

In 2019, Su *et al.* [1] propose a verifiable multi-key searchable encryption (VMKSE) scheme for cloud computing based on multi-key searchable encryption (MKSE). Their scheme is based on fine-grained data sharing [4] for the authorized users. The main contributions are promising, as the scheme is based on Garbled Bloom Filter and they define a security model with a strong security and

efficiency analysis for their proposed scheme, making their idea gaining trust and to give a chance to be implemented in a real environment.

In a symmetric searchable encryption scheme (SSE) the key used for encryption is used as well as for decryption, being carried for other algorithms from the scheme. In public encryption schemes with keyword search (PEKS), two cryptographic keys are used, public key (pub_k) and private key (prv_k).

Next, we will continue with the general steps (algorithms) of a **symmetric searchable encryption (SSE)** scheme and we will provide a short analysis of the main goals of each of the algorithms. Usually, a SE has four main algorithms and depends on the situations (complexity, complex processes, functionalities, complex architectures etc.) extra- algorithms can be added. This being said, the main four algorithms that should be included in such scheme, are:

1. *KeyGeneration.* The algorithm is invoked and run by the *DO*. To call this algorithm, the *DO* need to provide the security parameter ω. Based on ω, the algorithm will output the private key (prv_{key}).
2. *BuildIndex.* The algorithm is creating an index structure [6] and is being run by the DO. One of the input parameters is represented by the secret key (s_k) and the set of documents D. The output will be an index structure I.
3. *Trapdoor.* This algorithm is run by DU. Based on the kw, the algorithm will generate a trapdoor value. Using the trapdoor value, the algorithm will proceed with searching process. The trapdoor function will need for the searching process the s_k. The output is represented by a trapdoor value that usually is noted as T_{kw}.
4. *Search.* The algorithm usually is performed by the server. The input value for the algorithm is represented by the T_{kw} and I. An important observation is that the search algorithm does not make a simple matching of T_{kw} and I.

3. Basic example of implementation

Even that searchable encryption has an amazing potential, theoretical and practical, sometimes is quite difficult to provide an implementation from scratch in regular software applications with respect for improving the quality of the software [35, 37]. This is happening due to the roughness of the algorithms – if we have a wrong implementation of the algorithms, then we will have a wrong encryption, leading in the end to a very poor security and many resources that are consumed.

Depends on what is the goal of what we want to protect and the infrastructure in which the software application will be deployed, the practical implementations will be different. There are couple of practical implementations, such as Crypteron[1] [14] or CryptDB[2].

Most of the practical implementations, such as CryptDB for example are vulnerable to attacks on the database server, but also on the encrypted query and encrypted results. In case of man-in-the-middle attacks or exploiting the vulnerabilities of a server using dedicated tools (e.g., Metasploit, Kali Linux etc.), the results will not be so promising, and this will be shown in a future contribution. An interesting contribution for insider threats based on natural language processing and personality profiles is discussed in [31]. Insider threats should be treated with a maximum responsibility by the institutions as an untrained employee can cause security breaches and data leakages.

In the following example, we will show how we can implement a searchable function for accessing some documents related in a software application for maritime industry.

Below, in Code Listing 3.1 we can see an example of practical implementation for defining a query and receiving the messages, the implementation is done using C# 8.0 programming language [39]. These two functions represent *Step 4 - Search* from the general algorithms mentioned above. The implementation shown as an example is on the user side software application. The DU is the one who initiate the below implementation.

---

[1] Crypteron, https://www.crypteron.com/

[2] CryptDB, http://css.csail.mit.edu/cryptdb/

Code Listing 3.1. A simple implementation for searchable encryption *Search* algorithm

```
❶   public class MaritimeDocuments
❷   {
❸      public int DocumentID {get; set ;}
❹      [Secure]
❺      public string Keyword {get; set;}
❻      [Secure(Opt.Search)]
❼      public string query_based_on_keyword =
                  SE_SecureSearch.GetPrefix("trapdoor value");
❽     var results = connection_database.MaritimeDocuments
              .Where(p =>
              .DocumentKeywords.StartsWith(query_based_on_keyword));
❾   }
```

As we can observe from Code Listing 3.1, at a first sight it is quite simple to give an implementation for searching process, but if we pay attention to the implementation we can observe as well how easy is attack such implementation using software obfuscation attacks and techniques []. The main lines on which we need to focus are ❼ and ❽.

In Line ❼ we define a string variable that represents the query using a specific keyword (kw) or set of keywords (K), and by using *GetPrefix()* with the trapdoor value we will be able to get the first records that meets the criteria (trapdoor value). The trapdoor value is computed as we shown in Section 2. In Line ❽ we define an object that based on the content stored in *query_based_on_keyword* string variable from Line ❼ it will return the documents that match the query. In Line ❻, *Opt.Search* option will help developers to enable the searchable property for *DocumentKeywords* as shown in Line ❽. Before the search is being initiated, the keyword from the query is being processed by *SE_SecureSearch.GetPrefix* from Line ❼.

4. Proposed idea

In the following example, we have brought into discussion an idea of applied searchable encryption for a software maritime application (web or desktop) that can be adjusted and properly configured for a cloud architecture.

As we mentioned before, it is very important to be aware about the participants into the system. Next example will show and demonstrate theoretically and practically how strong as security [11-13], powerful as computations, and complex as implementation a searchable encryption can be.

The following scenario will be used to design the algorithms (which represents the theoretical background) and based on the algorithms we will provide a sketch for the implementation. The implementation sketch can be adjusted depends on the requirements of the maritime software application/system.

The scenario is based on the participants that interacts with the software system and infrastructure, as follows:

- the DU is in the possession of a set of documents D, making sure that the system is ready to generate the required keys, to provide encryption for them and send them to the cloud in order to store them;
- the DO will have the chance and possibility to submit the queries for searching process to cloud [15, 16, 17];
- the *cloud infrastructure* [34] will store the documents as encrypted and it will call the search algorithm;

The typical searchable encryption scheme proposed for a maritime software system contains a number of six algorithms, as follows:

1. $Key_{gen}(\omega) \rightarrow (pub_{key}, prv_{key})$: based on the security parameter (ω) two keys are generated, public key (pub_{key}) and private key (prv_{key}).
2. $E(D_i, pub_{key}) \rightarrow C_i$: the encryption algorithm will encrypt the set of documents using the pub_{key} and it will output a set of encrypted documents (C_i).
3. $B_I(D_i, kw, pub_{key}) \rightarrow I$: based on a certain document D_i, the keyword (kw) associated with the document and pub_{key}, the algorithm will output the index structure I that represents the association link between the document(s) and keyword(s).
4. $T(kw, prv_{key}) \rightarrow t_{kw}$: the trapdoor algorithm receives as input two parameters, kw and prv_key, based on which he will output the trapdoor value (t_{kw}).
5. $S(t_{kw}, pub_{key}, I) \rightarrow C$: the searching algorithm receives as input the t_{kw}, pub_{key} and I. The output of the algorithm is represented by the encrypted documents $C = \{C_{i_1}, \dots, C_{i_{kw}}\}$ with their kw or K associated.
6. $D(C, prv_{key}) \rightarrow D$: the role of the algorithm is to provide the decryption of the document or set of documents based on the C and prv_{key}. The output has the following representation $D = \{D_{i_1}, \dots, D_{i_w}\} \subset S$, D being the decrypted documents.

Before proceeding further with the implementation of the algorithms, a set of guidelines should be taken into consideration before doing so. Those guidelines are:

- The software architecture(s) (services etc.) and hardware infrastructure (server, database etc.);
- The hardware infrastructure and how it is represented for the maritime software application and how the security and cryptographic mechanisms are used [35-38].
- Providing a separate and independent implementation of the algorithms with respect for architecture, in such way that each of the algorithms from the searchable encryption scheme.

In the followings we will present the main important points of the implementation approach. For more details about our approach, a GitHub[3] repository is available. The implementation is purely indicative, representing a starting point for future improvements, quite easy to be adjusted for other systems as well.

In Code Listing 4.1. we will give an implementation for $Key_{gen}(\omega) \rightarrow (pub_{key}, prv_{key})$ algorithm, which is the first step of the proposed searchable encryption scheme.

Code Listing 4.1. A simple implementation for searchable encryption *key generation* algorithm

```
public class KeyGeneration  {
      //** Step 1-the algorithm from KeyGeneration step (algorithm)
      //** are runned and invoked by the data owner

     //** the function will return the policy,
     //** as a content or file
     public string GetPolicy(IServiceCollection policyService)  {
         policyService.AddAuthorization(policyChoices => {
             policyChoices.AddPolicy("Policy content", policy
                                 =>policy.Requirements.Add(new
                                 UserPolicy()));
         });
         policyContent = policyChoices.ToString();
```

[3] SE_Sketch GitHub Repository, https://github.com/mmihailescu-hub/SE_Sketch

```
      }
      //** getting server identity can be tricky and it has
      //** different meanings, such as the name of computer, IP etc.
      //** For the current example we will use the hardware ID
      public string GetServerIdentity() {  }

      //** class constructor
      public KeyGeneration(){}

      //** generation of secret key, server key and public parameters
      //** "#" represents the separator
      public string ReturnParameters() {
       StringBuilder sbParameters = new StringBuilder();

       sbParameters.Append(ownerSecretKey + "#" +
                                        serverKey + "#" +);
                                        publicParameters);  }
}
```

Let's continue with the implementation and look over the third algorithm of the search encryption, which is represented by building the index $B_I(D_i, kw, pub_{key}) \rightarrow I$ and illustrated in Code Listing 4.2. For this work we skipped the second algorithm that is represented by the encryption of the documents due to the fact that it does not bring any genuine to the implementation as most of the cryptographic algorithms are known.

Code Listing 4.2. A simple implementation for searchable encryption *build index* algorithm

```
public class BuildIndex
{
       //** Step 2
       //** the algorithm from BuildIndex step (algorithm)
       //** are runned and invoked by the data owner

       //** constructor of the class
      public void BuildIndex(){}

      //** the function centralize the build index parameters
      //** after their initialization and processing
      public void UseBuildIndexParameters()
      {
            LinkedList descriptionDataSet =
                                        new LinkedList();
            string ownerPrivateKey = string.Empty();
            string outputIndex = string.Empty();
      }
```

```
      //** simulation of getting the data set and their
      //** descriptions
      public LinkedList GetDataSet() {
       for(int i=0; i<dataSet.Length; i++)
            {
              LinkedList ll = new LinkedList();
              ll.Items.Add(dataSet[i],description[i]);
       }
      }

      //** getting the private of the owner
      public string ownerPrivateKey() {
       string privateKey = string.Empty();

       //** get the private key and work with it arround
       return privateKey;
      }

      //** get the index
      public string Index() {
       string index = string.Empy();

       //** implement the query for getting
       //** or generating the index
       return index;
      }

}
```

One of the most important and vital steps of the discussed searchable encryption scheme is represented by the fifth algorithm which consists in the search procedure - $S(t_{kw}, pub_{key}, I) \rightarrow C$ (see Code Listing 4.3).

Code Listing 4.3. A simple implementation for searchable encryption *key generation* algorithm

```
public class Search
{
       //** Step 5
       //** the algorithm from Search step (algorithm)
       //** are runned and invoked by the server

       //** the constructor of the Search class
       public void Search() {}

       public string SearchQuery() {
              string query = string.Empty();
```

```
            //** take the search query
            return query;
        }

        public string Index() {
            string index = string.Empty();

            //** take the search query
            return index;
        }

        public string ReturnResult() {
         string result = string.Empty();
         string setOfIdentifiers = string.Empty();

         //** based on the search query and index,
         //** get the set identifiers of the data items
         setOfIdentifier = "query for identifiers";

         //** build the result. "#" is the separator for
         //** illustration purpose only
         result = SearchQuery + "#" + Index;

         return result;
        }

}
```

5. Conclusions

The current work represents a practical and unique approach for searchable encryption and it can be used in real environments, such as maritime software applications.

The main idea of our work has been focused on providing a unique encryption mechanism for documents, that are exchanged between different users of a maritime software infrastructure using cloud computing as environment. Currently, the scheme has been tested on a virtual machine, such as VMware.

The main challenges that we have experienced were based on huge amount of time dedicated for processing multiple queries (~200 queries per second). These situations were remediated by using a proper load balancing mechanism. For example, if we will implement the scheme for other fields, such as biology or health, the queries will suffer significant delays in being processed. In physics, where the amount of data that are resulting as output from different experiments [22-24] the processing time is exponentially increasing.

As for future research directions, we have a list of tasks, such as improving the execution time (make it faster); providing complex encryption mechanism for documents that are stored in complex infrastructures; providing support for IoT environments and devices [25]; storing and securing complex media files and guaranteeing extra security mechanisms [27, 28]; and providing completely anonymous authentication mechanisms for users and for different systems, such as e-passports or e-lottery systems [29].

References

[1] Y. Su, J. Wang, Y. Wang and M. Miao, "Efficient Verifiable Multi-Key Searchable Encryption in Cloud Computing," in IEEE Access, vol. 7, pp. 141352-141362, 2019. DOI: 10.1109/ACCESS.2019.2943971.

[2] Yanjiang Yang, Haiyan Zhu, Haibing Lu, Jian Weng, Youcheng Zhang, Kim-Kwang Raymond Choo, Cloud based data sharing with fine-grained proxy re-encryption, Pervasive and Mobile Computing, Volume 28, 2016, Pages 122-134, ISSN 1574-1192. DOI: 10.1016/j.pmcj.2015.06.017.

[3] Wang, Y., Wang, J. & Chen, X. Secure searchable encryption: a survey. J. Commun. Inf. Netw. 1, 52–65 (2016). DOI: 10.1007/BF03391580.

[4] J. Shao, R. Lu and X. Lin, "Fine-grained data sharing in cloud computing for mobile devices," 2015 IEEE Conference on Computer Communications (INFOCOM), Hong Kong, China, 2015, pp. 2677-2685. DOI: 10.1109/INFOCOM.2015.7218659.

[5] Dawn Xiaoding Song, D. Wagner and A. Perrig, "Practical techniques for searches on encrypted data," Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, Berkeley, CA, USA, 2000, pp. 44-55. DOI: 10.1109/SECPRI.2000.848445.

[6] Goh, E. J. (2003). Secure indexes. IACR Cryptology ePrint Archive, 2003, 216.

[7] Boneh D., Di Crescenzo G., Ostrovsky R., Persiano G. (2004) Public Key Encryption with Keyword Search. In: Cachin C., Camenisch J.L. (eds) Advances in Cryptology - EUROCRYPT 2004. EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-24676-3_30.

[8] J. Li, Q. Wang, C. Wang, N. Cao, K. Ren and W. Lou, "Fuzzy Keyword Search over Encrypted Data in Cloud Computing," 2010 Proceedings IEEE INFOCOM, San Diego, CA, USA, 2010, pp. 1-5. DOI: 10.1109/INFCOM.2010.5462196.

[9] J. Bringer, H. Chabanne and B. Kindarji, "Error-Tolerant Searchable Encryption," *2009 IEEE International Conference on Communications*, Dresden, Germany, 2009, pp. 1-6. DOI: 10.1109/ICC.2009.5199004.

[10] Junzuo Lai, Xuhua Zhou, Robert Huijie Deng, Yingjiu Li, and Kefei Chen. 2013. Expressive search on encrypted data. In *Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security* (*ASIA CCS '13*). Association for Computing Machinery, New York, NY, USA, 243–252. DOI: 10.1145/2484313.2484345.

[11] Raphael Bost. 2016. ∑οφος: Forward Secure Searchable Encryption. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (*CCS '16*). Association for Computing Machinery, New York, NY, USA, 1143–1154. DOI: 10.1145/2976749.2978303.

[12] Javad Ghareh Chamani, Dimitrios Papadopoulos, Charalampos Papamanthou, and Rasool Jalili. 2018. New Constructions for Forward and Backward Private Symmetric Searchable Encryption. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security* (*CCS '18*). Association for Computing Machinery, New York, NY, USA, 1038–1055. DOI: 10.1145/3243734.3243833.

[13] Zuo C., Sun SF., Liu J.K., Shao J., Pieprzyk J. (2019) Dynamic Searchable Symmetric Encryption with Forward and Stronger Backward Privacy. In: Sako K., Schneider S., Ryan P. (eds) Computer Security – ESORICS 2019. ESORICS 2019. Lecture Notes in Computer Science, vol 11736. Springer, Cham. DOI: 10.1007/978-3-030-29962-0_14.

[14] Crypteron Documentation, https://www.crypteron.com/docs/.

[15] C. Ma, Y. Gu and H. Li, "Practical Searchable Symmetric Encryption Supporting Conjunctive Queries Without Keyword Pair Result Pattern Leakage," in *IEEE Access*, vol. 8, pp. 107510-107526, 2020. DOI: 10.1109/ACCESS.2020.3001014.

[16] Fu, S., Zhang, Q., Jia, N. *et al.* A Privacy-preserving Fuzzy Search Scheme Supporting Logic Query over Encrypted Cloud Data. *Mobile Netw Appl* (2020). DOI: 10.1007/s11036-019-01493-3.

[17] Boneh D., Di Crescenzo G., Ostrovsky R., Persiano G. (2004) Public Key Encryption with Keyword Search. In: Cachin C., Camenisch J.L. (eds) Advances in Cryptology - EUROCRYPT 2004. EUROCRYPT 2004. Lecture Notes in Computer Science, vol 3027. Springer, Berlin, Heidelberg. DOI: 10.1007/978-3-540-24676-3_30.

[18] B. Ernest and J. Shiguang, "Privacy Enhancement Scheme (PES) in a Blockchain-Edge Computing Environment," in IEEE Access, vol. 8, pp. 25863-25876, 2020. DOI: 10.1109/ACCESS.2020.2968621.

[19] N. S. Loredana, "On recommendation systems applied in big data," *2016 8th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*, Ploiesti, 2016, pp. 1-6. DOI: 10.1109/ECAI.2016.7861068

[20] M. Shahriar Rahman, A. Al Omar, M. Z. A. Bhuiyan, A. Basu, S. Kiyomoto and G. Wang, "Accountable Cross-Border Data Sharing Using Blockchain Under Relaxed Trust Assumption," in IEEE Transactions on Engineering Management, vol. 67, no. 4, pp. 1476-1486, Nov. 2020. DOI: 10.1109/TEM.2019.2960829.

[21] Christoph Bösch, Pieter Hartel, Willem Jonker, and Andreas Peter. 2014. A Survey of Provably Secure Searchable Encryption. ACM Comput. Surv. 47, 2, Article 18 (January 2015), 51 pages. DOI: 10.1145/2636328.

[22] V. Marascu, C. Stancu, V. Satulu, A. Bonciu, C. Grisolia, G. Dinescu, Material erosion and dust formation during tungsten exposure to Hollow-Cathode and Microjet Discharges, APPLIED SCIENCES-BASEL, Volume: 10, Issue: 19, Article Number: 6870, Published: OCT 2020. DOI: 10.3390/APP10196870.

[23] V. Marascu, A. Lazea-Stoyanova, A. Bonciu, V. Satulu, G. Dinescu, Tungsten particles fabrication by a microjet discharge, MATERIALS RESEARCH EXPRESS, Volume: 7, Issue: 6, Article Number: 066509, Published: JUN 2020. DOI: 10.1088/2053-1591/AB955D.

[24] V. Marascu, A. Lazea-Stoyanova, C. Stancu, G. Dinescu, The influence of plasma operation parameters on synthesis process of copper particles at atmospheric pressure, PLASMA PROCESSES AND POLYMERS, Volume: 15, Issue: 1, Article Number: e1700091, Published: JAN 2018. DOI: 10.1002/PPAP.201700091.

[25] Panda, P.K., Chattopadhyay, S. A secure mutual authentication protocol for IoT environment. J Reliable Intell Environ 6, 79–94 (2020). DOI: 10.1007/s40860-020-00098-y.

[26] Mihailescu Marius Iulian, Nita Stefania Loredana, and Pau Valentin Corneliu. E-Learning System Framework using Elliptic Curve Cryptography and Searchable Encryption. In Proceedings of International Scientific Conference for e-Learning and Software for Education, Volume 1, Pages: 545-552, 2020. DOI: 10.12753/2066-026X-20-071.

[27] Dăscălescu, A.C., Boriga, R.E. A novel fast chaos-based algorithm for generating random permutations with high shift factor suitable for image scrambling. Nonlinear Dyn 74, 307–318 (2013).DOI: 10.1007/s11071-013-0969-6.

[28] Radu Boriga, Ana Cristina Dăscălescu, Adrian-Viorel Diaconu, "A New One-Dimensional Chaotic Map and Its Use in a Novel Real-Time Image Encryption Scheme", Advances in Multimedia, vol. 2014, Article ID 409586, 15 pages, 2014. DOI: 10.1155/2014/409586.

[29] Florin Medeleanu, Ciprian Răcuciu, Madlena Nen, Zieduna Liepe & Narcis Florentin Antonie (2019) Fair e-lottery system proposal based on anonymous signatures, Applied Economics, 51:27, 2921-2933, DOI: 10.1080/00036846.2018.1563671.

[30] Mihailescu, Marius Iulian, and Stefania Loredana Nita. *Pro Cryptography and Cryptanalysis: Creating Advanced Algorithms with C# and .NET*. Apress, 2021. DOI: 10.1007/978-1-4842-6367-9.

[31] S. Eftimie, R. Moinescu and C. Răcuciu, "Insider Threat Detection Using Natural Language Processing and Personality Profiles," 2020 13th International Conference on Communications (COMM), Bucharest, Romania, 2020, pp. 325-330, DOI: 10.1109/COMM48946.2020.9141964.

[32] Opris, V.N. & Opris, M.E. 2016, "EXPERT SYSTEMS RUNNING ACROSS MULTIPLE

CLOUDS. A SUSTAINABLE PERSPECTIVE", Scientific Bulletin "Mircea cel Batran" Naval Academy, vol. 19, no. 2, pp. 585, 2016. DOI: 10.21279/1454-864X-16-I2-076.

[33] Opris, V.N. & Racuciu, C. 2015, "THE EXPERT SYSTEMS ANALYSIS USING THE CONCEPT OF BIG DATA AND CLOUD COMPUTING SERVICES", Scientific Bulletin "Mircea cel Batran" Naval Academy, vol. 18, no. 2, pp. 46-50, 2015. DOI: 10.21279/1454-864X-17-I1-0 8.

[34] L. A. Dumitru, S. Eftimie, M. I. Mihailescu, S. L. Nita, V. Opris and C. Racuciu, "A novel architecture for authenticating scalable resources in hybrid cloud," 2016 International Conference on Communications (COMM), Bucharest, Romania, 2016, pp. 251-254. DOI: 10.1109/ICComm.2016.7528254.

[35] Albeanu G., Madsen H., Popențiu-Vlădicescu F. (2020) Computational Intelligence Approaches for Software Quality Improvement. In: Pham H. (eds) Reliability and Statistical Computing. Springer Series in Reliability Engineering. Springer, Cham. DOI: 10.1007/978-3-030-43412-0_18.

[36] F. Popențiu-Vlădicescu, G. Albeanu and H. Madesn, "Reliability of Modern Engineering Systems - Towards a Safer World," 2019 2nd International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2019, pp. 1-5. DOI: 10.1109/ICOMET.2019.8673474.

[37] Popentiu-Vladicescu, F., Albeanu, G., & Madsen, H. (2019). Improving software quality by new computational intelligence approaches. In Proceedings of 25th ISSAT International Conference on Reliability & Quality in Design (pp. 152-156). [RQD25-152]

[38] F. Popentiu-Vladicescu and G. Albeanu, "Software reliability in the fog computing," 2017 International Conference on Innovations in Electrical Engineering and Computational Technologies (ICIEECT), Karachi, 2017, pp. 1-4. DOI: 10.1109/ICIEECT.2017.7916578.

[39] Mihailescu, Marius Iulian and Stefania Loredana Nita. "CSAP: Cyber Security Asynchronous Programming With C++20 and C# 8 for Internet of Things and Embedded Software Systems." Examining the Impact of Deep Learning and IoT on Multi-Industry Applications, edited by Roshani Raut and Albena Dimitrova Mihovska, IGI Global, 2021, pp. 249-269. DOI: 10.4018/978-1-7998-7511-6.ch014.

[40] Kuchta V., Sharma G., Sahu R.A., Markowitch O. (2018) Multi-party (Leveled) Homomorphic Encryption on Identity-Based and Attribute-Based Settings. In: Kim H., Kim DC. (eds) Information Security and Cryptology – ICISC 2017. ICISC 2017. Lecture Notes in Computer Science, vol 10779. Springer, Cham. DOI: 10.1007/978-3-319-78556-1_5.

[41] YANG, Yang; LIU, Ximeng; and DENG, Robert H.. Multi-user multi-keyword rank search over encrypted data in arbitrary language. (2017). *IEEE Transactions on Dependable and Secure Computing*. 17, (2), 320-334. Research Collection School Of Computing and Information Systems. DOI: 10.1109/TDSC.2017.2787588.