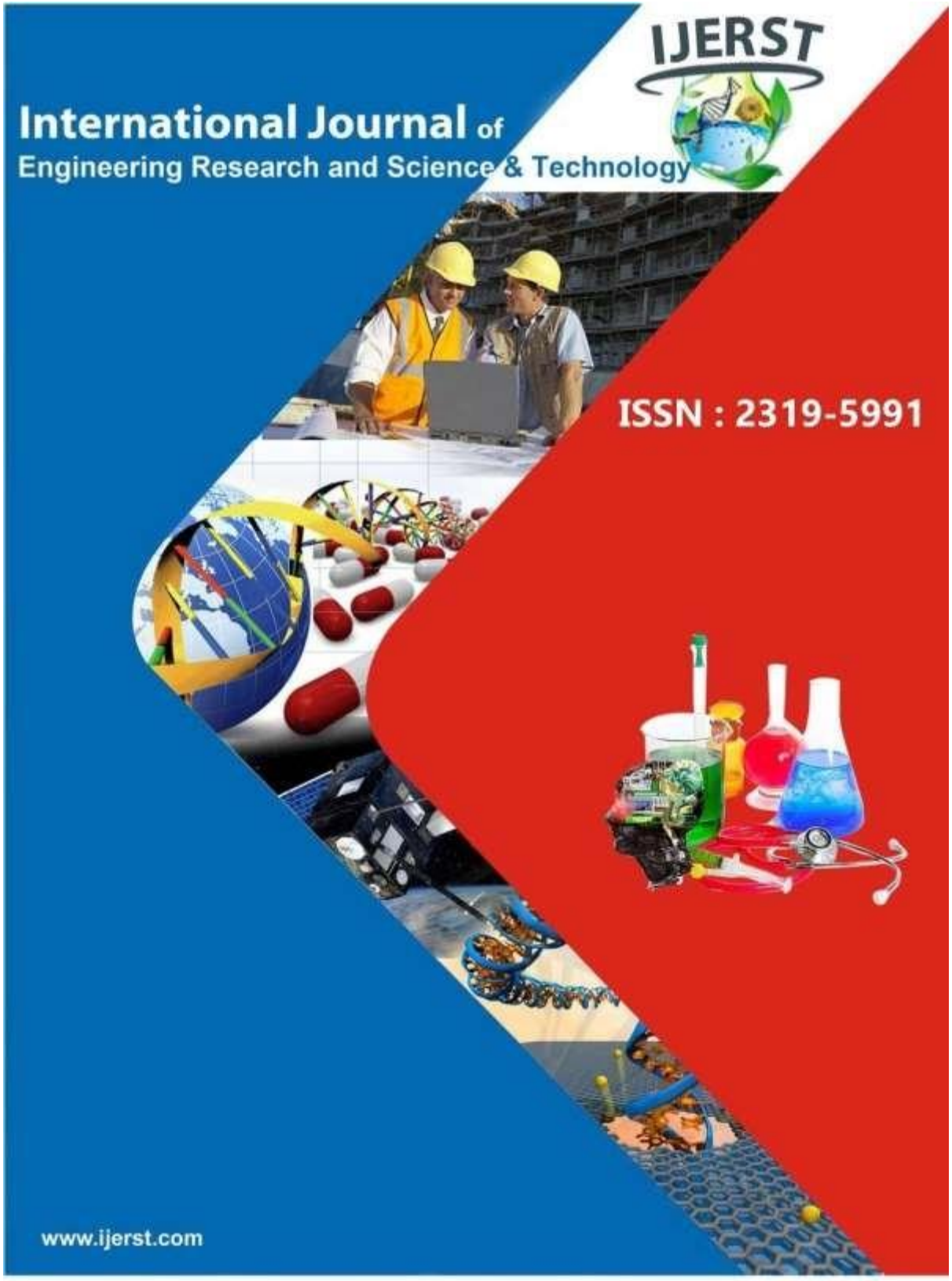


Email: editor@ijerst.com or editor.ijerst@gmail.com

International Journal of Engineering Research and Science & Technology

ISSN 2319-5991 www.ijerst.com

Vol. 19, Issuse.2,April 2023

SMARTCRYPT: SECURE STORING AND SHARING OF TIME SERIES DATA STREAMS IN IIOT

Dr. Y. Jayababu [1],Puvvala Sai Krishna [2],Abhishek Gedela [3],Golagani Girish Kumar [4],Kasturi Swetha Sri Anjani [5],Rapeti Phaneendra Charan [6]

Article Info

Received: 07-01-2023 *Revised: 09-02-2023* *Accepted: 01-03-2023*

ABSTRACT

To provide ubiquitous access, scalability and sharing possibilities, the Industrial Internet of Things (IIoT) applications utilize the cloud to store collected data streams. However, secure storing and sharing of the massive andcontinuously generated data poses significant privacy risks, including data breaches. This paper proposes SmartCrypt, a data storing and sharing system that supports analytics over the encrypted time series data. SmartCrypt enables users to secure and fine grain sharing of their encrypted data using a novel symmetric homomorphic encryption scheme. Simulation results show that SmartCrypt reduces query time by 17% and improves throughput by 9% over the benchmark scheme.

INTRODUCTION

In smart manufacturing, Industrial Internet of Things (IIoT) devices produce a significant amount of time series data related to production, monitoring and maintenance that need to be processed and stored. Due to storage and processing constraints of IIoT devices, nowadays the data storage and processing functionalities are mostly shifted to the time series database in cloud platform, e.g., Azure Time Series Insight. Further, processing and storing the data in the cloud platform enhances ubiquitous access, scalability and sharing possibilities. However, secure data storing in the cloud poses significant privacyrisks, including unauthorized access of production line efficiency data. To address privacy risks, encrypted databases have appeared as a promising solution.

The main advantage of this approach is that it allows data owners and third-party services to query encrypted data while maintaining both functionality and confidentiality. Recently, research in this domain has led to several encrypted databases, e.g., relational databases and batch analytics.

Secure time series data storing in the cloud and sharing them with third-party services come with unique performance and challenges that current encrypted data processing systems fail to meet.

PROFESSOR
DEPT OF COMPUTER SCIENCE AND ENGINEERING
PRAGATI ENGINEERING COLLEGE(A),SURAMPALEM(EAST GODAVARI)A.P,INDIA

To address these challenges, numerous databases have been devised, particularly for time series data. However, all these databases incur significant overhead during encrypted data processing. Besides, another key challenge in smart manufacturing is that privacy should co-exist during queries on data statistics, e.g., finding the standard deviation, which usually indicates sharing data to be examined by third-party services. Further, data sharing must be fine grained as it is often unnecessary to provide third-parties free access to the data.

Instead, data owners might like to (i) share only statistical computation of the data, e.g., mean, sum, max, min, (ii) restrict the granularity at which such statistical computations are reported, e.g., per-minute, per hour, (iii) restrict the time interval over which queries are generated, e.g., February 2021, and (iv) a combination of earlier three choices. We believe that support for encrypted query processing should go together with access control to restrict the scope of data that users may query. The sharing procedure for data stream stored in the time series databases is considerably distinct from traditional databases. Particularly, in smart manufacturing, various levels of production process continuously push data streams to the cloud, where numerous services can subscribe to access and analyze data streams. Furthermore, often there is a requirement to aggregate and analyze time series data from different production processes collaboratively.

1. LITERATURE SURVEY

TITLE: ―Big data analytics over encrypted datasets with seabed,‖

ABSTRACT:Today, enterprises collect large amounts of data and leverage the cloud to perform analytics over this data. Since the data is often sensitive, enterprises would prefer to keep it confidential and to hide it even from the cloud operator. Systems such as CryptDB and Monomi can accomplish this by operating mostly on encrypteddata; however, these systems rely on expensive cryptographic techniques that limit performance in true ―big data‖ scenarios that involve terabytes of data or more.

This paper presents Seabed, a system that enables efficient analytics over large encrypted datasets. In contrast to previous systems, which rely on asymmetric encryption schemes, Seabed uses anovel, additively symmetric homomorphic encryption scheme (ASHE) to perform large-scale aggregations efficiently. Additionally, Seabed introduces a novel randomized encryption scheme called Splayed ASHE, or SPLASHE, that can, in certain cases, prevent frequency attacks based on auxiliary data.

TITLE: ―InfluxDB Cloud,‖

ABSTRACT:The recent great technological advance has led to a broad proliferation of Monitoring Infrastructures, which typically keep track of specific assets along time, ranging from factory machinery, device location, or even people. Gathering this data has become crucial for a wide number ofapplications, like exploration dashboards or Machine Learning techniques, such as Anomaly Detection. Time-Series Databases, designed to handle these data, grew in popularity, becoming the fastest- growing database type from 2019. In consequence, keeping track and mastering those rapidly evolving technologies became increasingly difficult. This paper introduces the holistic design approach followed for building NagareDB, a Time-Series database built on top of MongoDB—the most popular NoSQL Database, typically discouraged in the Time-Series scenario. The goal of NagareDB is toease the access to three of the essential resources needed to building time-dependent systems: Hardware, since it is able to work in commodity machines; Software, as it is built on top of an open- source solution; and Expert Personnel, as its foundation database is considered the most popular NoSQL DB, lowering its learning curve. Concretely, NagareDB is able to outperform MongoDB recommended implementation up to 4.7 times, when retrieving data, while also offering a stream-ingestion up to 35% faster than InfluxDB, the most popular Time-Series database. Moreover, by relaxing some requirements, NagareDB is able to reduce the disk space usage up to 40%.

TITLE: ―SHAMC: A secure and highly available database system in multi-cloud environment,‖

ABSTRACT:Data owners outsource their databases into the cloud to enjoy the quality services provided by the cloud service providers. However, using cloud database makes the private data vulnerable and exposed to the attackers including malicious insiders. Many researchers try to find the way to encrypt the cloud database and execute queries securely on the ciphertext. In this paper, we propose a secure and highly available cloud database system in the multi-cloud named SHAMC. Specifically, we use the idea of secure multiparty computation and homomorphic encryption to store data and execute queries direct on the ciphertext. Besides, the entire database is stored in multiple clouds to avoid service interruption as well as solve the problems of permanent failure and vendor lock-in. We implement the prototype of

SHAMC which supports all queries in TPC Benchmark™ H (TPC-H) on the top of the commercial cloud. SHAMC is proved to be highly available and cost-efficient. The evaluation shows it has an acceptable query overhead which is superiorto other encrypted cloud databases.

2. EXISTING SYSTEM

Most of the existing state-of-the-art security solutions are designed for relational databases instead of time series database. Although, the researchers in have proposed a security mechanism for storing and sharing of data streams, it is vulnerable to the malleability attacks. Besides, the existing time series databases failed to provide suitable access policies to allow data owners a fine-grain protection during selective and secure sharing of data streams with third-party services in multi-user smart manufacturing settings.
Our main contributions in this paper are three-fold.
(i) We design SmartCrypt, a symmetric homomorphic encryption-based access control technique for flexible and fine-grain sharing of encrypted data streams.
(ii) (ii) We introduce a Homomorphic Message Authentication Code (HomMAC) based verification technique that supports source authentication and provides data integrity checks.
(iii) (iii) Our experimental results show that SmartCrypt significantly improves the query time, latency and throughput compared to the state-of-the-art realization, TimeCrypt.

3.2 PROPOSEDSYSTEM

This section provides details about our proposed scheme.
A. Storing of Encrypted Data In this section, we illustrate how SmartCrypt encrypts and stores the time series data streams in the cloud server.
• Symmetric Homomorphic Encryption: Let mi be a message to be encrypted from the message space [0, M − 1], i.e., mi ∈ [0, M − 1] and size of mi is an integer, where i = 1, . . . , n. Let ki be a randomly generated secret key stream used to encryptmi , where ki ∈ [0, M − 1]. To encrypt mi , the data owner computes the ciphertext as ci = Eki (mi) = (mi + ki) mod M. In contrast, to retrieve mi for given ki , one can perform decryption as mi = Dki (ci) = (ci − ki) mod M.

B. Sharing of Encrypted Data In SmartCrypt, our objective is to allow data streams access permissions to the third-party services at arbitrary intervals or temporal ranges like from 13.00-Feb 05 till 12.00-Feb 06 2021. To achieve this, SmartCrypt partitions the data streams into fixed-length time segments or chunks of size Δ, e.g., 20 sec. Each chunk is then encrypted using a separate key from thekey stream, subsequently indexed by the time window of the chunk. To achieve fine-grain access control and enable data owners to share encrypted data streams of desire intervals, we devise a hierarchical key derivation tree.

SYSTEM ARCHITECTURE .
BelowdiagramdepictsthewholesystemarchitectureofSmartCrypt – Secure Storing and Sharing of Time Series Data Streams in IIOT.

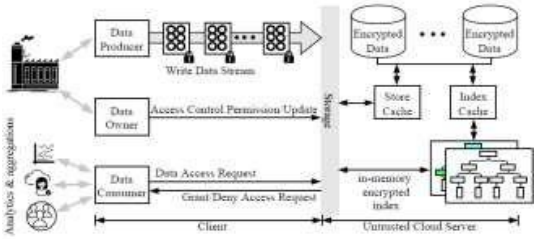
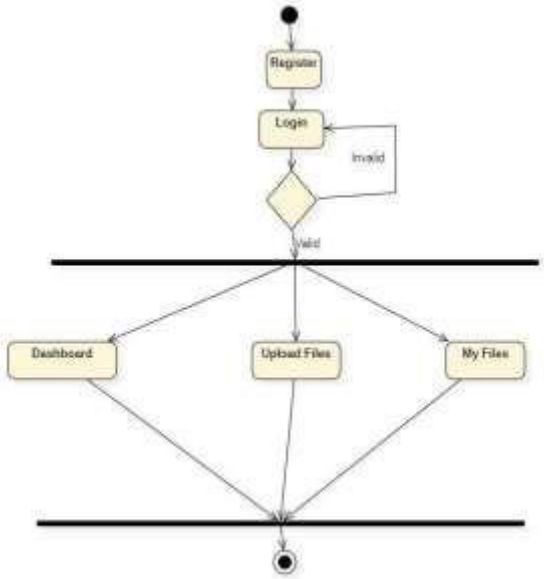


Fig. 1. The SmartCrypt architecture.

Activity Diagram

A graphical representations of work process of stepwise exercises and activities with supportfor decision, emphasis and simultaneousness, used to depict the business and operational well-orderedstreamof parts ina framework furthermoredemonstratesthegeneral streamof control.



5.SYSTEM IMPLEMENTATION

There are 4 modules:

1.Data Producer: It is the set of IIoT devices, e.g., appliances, services, which generate time series data. The main function of this actor is to ingest time series data into the cloud server and run SmartCrypt's client

library, which manages data stream pre-processing and encryption.

2. Data Consumer: These are entities like third-party services who are authorized to access data owner's time series data and produce an added value, e.g., aggregate numerous data streams for monitoring, analyzing and visualizations.

3. Data Owner: It owns the data stream and grants access permissions to its generated data stream. Data owners determine policies to selectively expose their data streams to data consumers. Based on the defined access policy, database server grants or denies data stream access requests.

4. Database Server: It is mainly responsible for storing encrypted data stream and giving access to data consumers following policies as defined by the data owner.

In SmartCrypt, cloud server performs analytical and statistical queries over the ciphertext and sends back the ciphertext to the data consumer. Only the data consumer, which owns the correct keys can decrypt the ciphertext and obtain statistical result (e.g., mean/max/min) and analytic (e.g., trend detection). To augment fast queries and analytics, SmartCrypt creates in-memory encrypted indices.

TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTING

■ Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

■ Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

■ Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

7.RESULTS

Fig. 7.1 Home page of the project

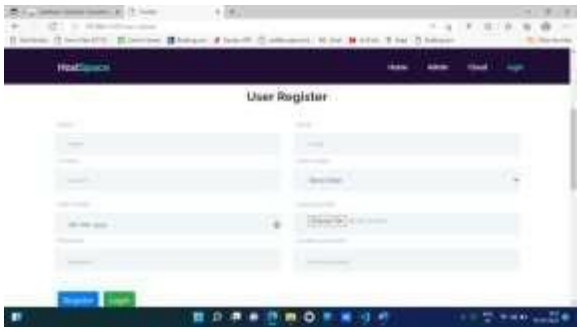


Fig. 7.2 This fig showsthe User Register page

Fig7.3 This fig showsthe Dashboard in Admin page

Fig. 7.4 Datanode View Real-Time File Migration Analysis

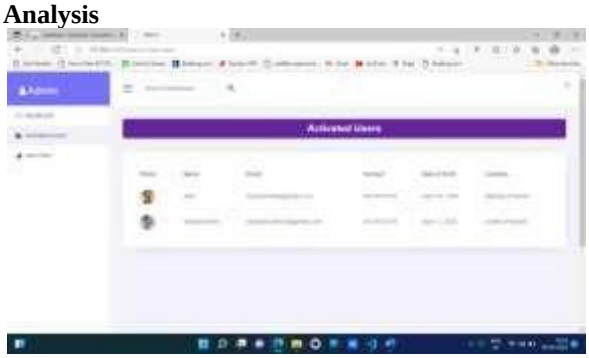


Fig7.5 This fig showsthe Activated Users in Admin page

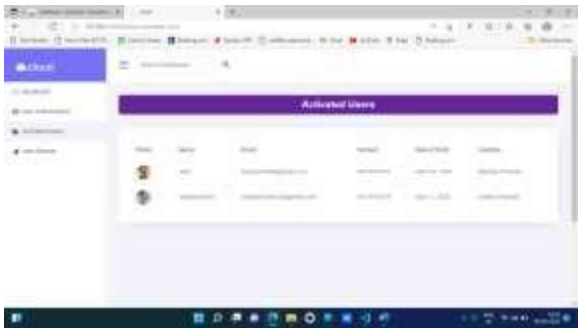


Fig. 7.6 This fig showsthe Activated Users in Cloud

8. CONCLUSION & FUTURE WORK

In this paper, we proposed SmartCrypt, a data storing and sharing system that supports analytics over massive encrypted time series data. We introduced a novel symmetric homomorphic encryption-based access control technique tailored for time series data. SmartCrypt enables the execution of analytics over encrypted data streams and empowers users toimpose access restrictions on encrypted data streams considering their access control and privacy preferences. Our evaluation on real-world datasets show the feasibility of SmartCrypt as an authorization service for secure and fine-grain sharing, and performing analytics on large-scale time series data.

REFERENCES

[1] A. Papadimitriou, R. Bhagwan, N. Chandran, R. Ramjee, A. Haeberlen, H. Singh, A. Modi, and S. Badrinarayanan, ―Big data analytics over encrypted datasets with seabed,‖ in Proc. of 12th USENIX OSDI, 2016, pp. 587–602.

[2] InfluxDB, ―InfluxDB Cloud,‖ [Online]: Accessed on February 06, 2021, https://www.influxdata.com/.

[3] L. Wang, Z. Yang, and X. Song, ―SHAMC: A secure and highly available database system in multi-cloud environment,‖ Future Generation Computer Systems, vol. 105, pp. 873–883, 2020.

[4] Y. Hu, S. Kumar, and R. A. Popa, ―Ghostor: Toward a secure datasharing system from decentralized trust,‖ in 17th USENIX NSDI, 2020, pp. 851–877.

[5] L. Burkhalter, A. Hithnawi, A. Viand, H. Shafagh, and S. Ratnasamy, ―Timecrypt: Encrypted data stream processing at scale with cryptographic access control,‖ in Proc. of 17th USENIX NSDI, 2020, pp. 835– 850.

[6] D. Catalano and D. Fiore, ―Practical homomorphic message authenticators for arithmetic circuits,‖ J. of Cryptology, vol. 31, no. 1, pp. 23–59, 2018.

[7] A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias, ―Delegatable pseudorandom functions and applications,‖ in Proc. of 20th ACM CCS, 2013, pp. 669–684.

[8] W. Kleiminger, C. Beckel, and S. Santini, ―Household occupancy monitoring using electricity meters,‖ in Proc. of ACM UbiComp, 2015, pp. 975–986