Security and Privacy Solutions associated with NoSQL Data Stores

Gerasimos Vonitsanos*, Elias Dritsas*, Andreas Kanavos*, Phivos Mylonas†, Spyros Sioutas*
*Computer Engineering and Informatics Department
University of Patras, Patras, Greece
{mvonitsanos, kanavos, sioutas}@ceid.upatras.gr, eldritsas@gmail.com
†Department of Informatics
Ionian University, Corfu, Greece
fmylonas@ionio.gr

***Abstract*—Technologies such as cloud computing and big data management, have lately made significant progress creating an urgent need for specific databases that can safely store extensive data along with high availability. Specifically, a growing number of companies have adopted various types of non-relational databases, commonly referred to as NoSQL databases. These databases provide a robust mechanism for the storage and retrieval of large amounts of data without using a predefined schema. NoSQL platforms are superior to RDBMS, especially in cases when we are dealing with big data and parallel processing, and in particular, when there is no need to use relational modeling. Sensitive data is stored daily in NoSQL Databases, making the privacy problem more serious while raising essential security issues. In our paper, security and privacy issues when dealing with NoSQL databases are introduced and in following, security mechanisms and privacy solutions are thoroughly examined.**

***Index Terms*—NoSQL Databases, NoSQL Security, Secure Databases, Policy Update, Access Control**

I. Introduction

The advances in cloud computing technology and distributed web applications, along with the ever-increasing large volume of data for storage and further processing, has rendered necessary the adoption of non-relational databases, known as NoSQL or "Not only SQL" [20]. It is widely known that the traditional SQL database is not able to cope with Big Data [25] as NoSQL systems, nowadays, are experimenting with an increase in popularity [18]. In recent years, many NoSQL databases have made their appearance; for example, Cassandra and MongoDB are two popular ones, to name a few. Some useful features of NoSQL databases are the high availability, scalability, better performance, as well as the ability to store and process large-scale semi-structured and/or unstructured data faster than traditional RDBMS [18], [25].

However, due to the ever increasing use of NoSQL databases, a significant amount of sensitive data is exposed to a number of security vulnerabilities, threats and risks. Lack of encryption support and poor authentication between servers and clients are some of the leading security issues in NoSQL Databases. Also, it should be noted that simple authorization is provided without support for role-based access control (RBAC) and so, there is no protection for injections and denial of service attacks [12].

Brewer in [6] made a conjecture about the trade-offs in the development of a distributed database system, thus introducing the CAP (Consistency, Availability, Partition Tolerance). A formal version of Brewer's conjecture is officially published as the CAP theorem in [11]. Specifically, the CAP theorem indicates that no shared data system could provide at the same time more than two out of the three properties, including consistency, availability, as well as partition tolerance.

Regarding organizations, Amazon developed the Dynamo technology [10], whereas Google produced the distributed storage system Bigtable [7]. These particular technologies have inspired many NoSQL applications installed in companies, like Facebook or Twitter. Modern companies are dealing with data that are not relational and need superior databases than traditional ones, which encounter scalability and availability problems because of their data size. There are already several authorization models in relational databases where views are usually utilized. In this way, SQL queries are used to display a specific state of a specified part of the database [3]. Some NoSQL Databases managed by Big Data use new authorization models, which are specifically designed for structure, speed, and a huge amount of data. These models include key-value, wide column, and document-oriented authorization. In addition, the storage and retrieval of these records are achieved through a unique key for each record while providing a swift search [8].

In this work, we present security and privacy issues in NoSQL databases and further examine it to propose the most efficient security mechanisms and privacy solutions. More to the point, data protection and access control are some of the key issues of security in NoSQL, while several security threats for NoSQL databases are considered, such as distributed environment, authentication, fine-grained authorization and protection of data at rest and in motion.

The remainder of this paper is organized as follows. Section II presents a survey of existing related works concerning mechanisms to overcome security issues. Moreover, Section III overviews a comparative study between Relational and NoSQL Databases, while in Section IV, our security and privacy-preserving mechanisms are proposed. Finally, in Section V, the summary of the proposed paper is presented.

II. Related Work

Many early papers that issued the relationship between Relational and NoSQL databases were given an overview of NoSQL database, as well as its types and characteristics. They were so enthusiastic about NoSQL and how it declined the dominance of SQL [21], [26]. However in [4], there was discussion about structured and non-structured database; there it was also explained how the use of NoSQL databases, like Cassandra, improved the performance of the system. In addition, it can scale the network without changing any hardware or needing to alternate the server infrastructure. This results in improving the network scalability, with low-cost commodity hardware.

In [14], a survey paper regarding relational databases is introduced along with NoSQL features and shortcomings. In addition, these shortcoming and issues of the NoSQL databases have been mentioned in [16]; complexity, consistency as well as limited eco structures are considered as serious concerns. Also in [19], the authors state that the demand for relational database will not go away anytime soon, and it will exclusively serve in line of application that will support business operations. However, NoSQL databases will serve the large, public and content centric applications. Another similar work is the one presented in [20], whaere an extensive analysis for security issues with NoSQL Database, like Cassandra and MongoDB, is considered.

Several solutions have also been proposed to improve privacy-preserving in NoSQL databases. More specifically, in Arx, a proxy is employed in order to rewrite NoSQL queries at the trusted premises. A back-end component, deployed at the untrusted premises, is used to perform computation over encrypted data [23]. In terms of BigSecret system, standard encryption is used for protection of the stored data, while the indexes are encoded using special techniques to allow comparisons (pseudo-random functions) and range queries (order-preserving partitioning) [22]. Authors in [27] employ algorithms of searchable encryption to build a privacy-preserving key-value store on top of the Redis database. In this approach, the values are protected with symmetric encryption, while the keys are secured with pseudo-random functions. In another solution, SafeRegions combines secret sharing and multiparty computation to perform secure NoSQL queries on three independent and untrusted HBase clusters providing thus simultaneously secure computation over the stored values and security guarantees similar to standard encryption [24].

III. Comparison of Relational and NoSQL Databases

During the last decades, relational databases, sub-divided into groups known as tables, have been used with the aim of storing structural data. The units of data in each table are known as columns, and each unit of the group is known as a row. Also, the columns in a relational database have relationships amongst them. This situation tends to change over the last years due to the rise of large web applications, which outputs a huge amount of data that traditional relational databases cannot handle any more [9]. NoSQL databases are sometimes referred as "Not only SQL" so as to give some emphasis on the fact that they may support query languages that are SQL-like. Nowadays, it is stated that NoSQL databases have more to offer than just presenting solutions to scale problems, while also providing many important advantages [9] like the following:

- The data representation is schema-less, and there is no need to define a certain structure from the beginning since new fields at run-time can be added.
- The speed, even with a small amount of data, can be processed in milliseconds instead of hundreds of milliseconds.
- The elasticity of the applications due to the scalability features that NoSQL databases offer.
- Reduce in development time, as developers do not have to deal with complex SQL queries and difficult joints so as to collate the data from different tables into a new view.

Some of the differences between relational and NoSQL databases are listed in the following paragraphs.

A. Reliability of Transactions

The ACID (atomicity, consistency, isolation, durability) model is fully supported by the design of relational databases, providing high reliability in transactions unlike the NoSQL databases.

B. Scalability Issues and Cloud Support

The primary purpose of cloud technology is to provide services to end-users. NoSQL databases are fully compatible with cloud environment requirements as they can analyze not only raw structured data, but also semi-structured or unstructured data from different sources, since they are not compliant with the ACID model. On the other hand, the relational databases do not provide data search on full content and their characteristics are now designed for cloud use.

It is possible that the need for scalability is one of the most significant problems of relational databases as they rely on vertical scalability to upgrade the performance. More specifically, this upgrade method requires the purchase of expensive equipment such as RAM, processors, SSD hard drives, etc. and in some cases, this is not easily achieved due to each system constraints. Also, the possibility of horizontal scaling is not supported by the addition of extra nodes and therefore, cannot support demanding online applications with many users and distributed data. However, NoSQL databases support only horizontal scaling since they do not deal with relational data.

C. Complexity and Big Data Management

The complexity of NoSQL databases is less than that of relational databases as it is not necessary to create tables to record data, but the modeling by considering a query method can be used. Also, the development of a database structure on a relational database is always considered a complicated task compared to the abstract model of a NoSQL database, where

data can be stored regardless of whether they are structured, unstructured, or semi-structured.

NoSQL databases have a valuable role in Big Data management since they are well-suited for storing or retrieving data in high speed across distributed nodes, thus taking advantage of multi-core GPU architectures. In relational databases, where accuracy is more important than speed, the data should be stored in tables' rows and columns, while the scalability is always considered a big issue. In the case of supporting conventional applications with small datasets, they are the most reasonable choice, but slitting the data across different servers increases the arduousness requiring complex SQL queries for joining the data again.

D. Data Model

Sets in mathematics are the driving force for relational database; all the data are represented as mathematical n-ary relations, where an n-ary relation is a subset of the Cartesian product of N domains. The data are represented as tuples inside the database and are further grouped into relations. The relation (represented by table) contains a set of tuples (represented by rows); where the column in the relation table utilizes the sequence of attributes, the type of an attribute is identified by the domain, which is the set of values that have a common meaning. This data model is very specific and well organized, while the columns and the rows are described by a well-defined schema.

NoSQL databases can employ many modelling techniques like graph, key-value stores and document data model. In terms of classification procedure, NoSQL is named after their data model but in some cases, NoSQL database system can be identified by using two or more of the data models that represent their data. NoSQL data model does not utilize the table as storage structure of the data and this is considered the main feature that distinguishes the NoSQL from relational databases. Furthermore, it is schema-less and as a result, can handle the unstructured data like word, pdf, images, as well as video files, in a very efficient method.

E. Data Warehouse and Crash Recovery

Regarding data warehousing, relational databases gather data from many sources and the oversize of stored data results in big data problems. To name a few, some problems are the performance degradation when utilizng an OLAP (Online Analytical Processing), statistical process or Data Mining. On the other hand, NoSQL databases are not designed when considering data warehouse applications, because designers are focused on scalability, availability and high performance.

Crash recovery is implemented in relational databases via the recovery manager, which is responsible for ensuring durability and transaction atomicity by using log files and ARIES algorithm. The crash recovery in NoSQL databases depends on replication to recover from the crash.

F. Privacy and Security

Most relational databases do not provide any feature regarding embedding security in the database itself. As a result, this requires developers to impose directly security systems in the middleware. Classic cryptography mechanisms and encryption protocols, such as asymmetric key encryption schemes, digital signature schemes, zero-knowledge Proof of Knowledge, as well as commitment schemes, which are based on SRSA (Strong RSA), bilinear maps [2], discrete logarithm, homomorphic encryption, fully or not [1], have been widely considered for securing communication and ensuring data confidentiality in relational databases.

Nonetheless, one of the most serious shortcomings of NoSQL databases is considered the fact that data files are not by default encrypted, but such a process takes place in the application layer before sending data to the database server. Although there are solutions that provide encryption services, these lack horizontal scaling and transparency required in the NoSQL environment.

Furthermore, only a few NoSQL databases provide encryption mechanisms to protect user-related sensitive data. By default in NoSQL databases, the inter-node communication is not encrypted and does not support SSL (Secure Sockets Layer) client-node communication (as in relational databases), breaking the network security [25]. Also, there is no integration of authentication or authorization mechanisms. The distributed environments increase attack surface across several distributed nodes and enforcing integrity constraints is much complex in NoSQL databases. In general, only a few categories of NoSQL databases provide mechanisms to employ encryption techniques protecting data at rest.

IV. Proposed Security and Privacy Solutions

Below are our proposed security and privacy solutions for NoSQL data stores.

A. Pseudonyms-based Communication Network

In the context of this system, users can have access to multiple services by inserting their credentials only once, that is when they are initially connected to the system. Such a system is called anonymous because users can be known only through their pseudonyms, and the transactions carried out by the same user cannot be linked as their identity is disclosed. For this reason, it is considered the best means in terms of user protection. Furthermore, it is based on two vital protocols, the RSA (Rivest–Shamir–Adleman) and the Diffie Hellman. Its structure and operations extend Brand's credential system [5], whereas it consists of four parties: the users U, a central Identity Provider denoted as IP, the Service Providers SPs, and the organization for issuing and validating credentials. Users are entities that receive credentials and are known to Service Providers only through their pseudonyms.

The central Identity Provider creates its own public and secret key, denoted as (P, S) respectively, and uses its secret key to digitally sign its sensitive data. Each credential is encoded with $m + 1$ attributes, denoted as $y_1, y_2, \ldots, y_m, t$ (where t is the credential issuing time). The IP decides on the $\mathbb{G}_q$, a finite cyclic group of prime order q, to which the

random generators $g_1, g_2, \ldots, g_m, g_{m+1}, h_0$, involved in keys generation, belong to. Specifically,

$$S = (y_1, y_2, \ldots, y_m, t, s) \tag{1}$$

$$P = g_1^{y_1} g_2^{y_2} \ldots g_m^{y_m} g_{m+1}^{t} h_0^{s}, \tag{2}$$

where $s \in \mathbb{Z}_q$ keeps secret.

Under the Discrete Logarithm assumption in $\mathbb{G}_q$, these keys are unique. The IP is responsible for the distribution of the digital pseudonyms $p_1, p_2, \ldots, p_m$ to any user. An organization can issue a credential to a pseudonym, and the corresponding user can prove its ownership to another organization (who knows it by a different pseudonym), by just revealing the ownership of the credential.

Additionally, the Credential Authority CA prevents the sharing of credentials or pseudonyms and guarantee that users who enter the system have a public and secret key that makes them unique to the system. Another entity in the system is the Verifier V, whose role is to certify the validity of the user credentials and to communicate with either the Issuing Authority or the Credential Authority to inform that the user is not the owner of the credential that is presenting. A user, in terms of a digital credential, transmits the public key and the CA's digital signature derived from a Proof of Knowledge, through which they prove that they know the secret key and the attributes in the digital credential that satisfy the particular attribute property they are revealing.

Each pseudonym and credential belong to well-defined users. More in detail, it is impossible for different users to collaborate and show some of their credentials in a Service Provider as well as to obtain a credential for a user that they could not obtain (coherent credentials). As organizations are autonomous and separable entities, they can select their public and secret key independently of the other entities, so as to ensure the security of these keys and facilitate the keys management system.

The pseudonyms system can protect user privacy and provide security, as in such a system, an organization can not find out anything about a user other than the ownership of a set of credentials. Specifically, two pseudonyms that belong to the same user cannot be linked (unlinkability) and identified as in the Brand's system, except for specific conditions. In order to be efficient, any communication in the system involves as few entities as possible along with the minimum amount of information. If a user holds a credential, this can be shown multiple times without the need to reissue (and consequently resign) it.

When a user has access to a service, they are validated by proving that they know the secret key of their pseudonym, without revealing it, thus preventing pseudonyms repetition. Also, for each pseudonym that a Service Provider associates with a user, it requires the user to unveil a different encoded random number of their pseudonym each time and thus, ensuring the unconditional unlinkability of their pseudonyms. Although the Identity Provider blindly encodes the random numbers in all of a user's pseudonyms, that are uniquely related with them, if a user makes abuse of the service, the SP can blacklist and reveal numbers. In following, it is able to globally revoke their pseudonyms and abolish their access to any of the services they have previously had.

Finally, users, under Discrete Logarithm, can conclusively prove that their encoded numbers do not belong to the SP's blacklist, while using this one as input on a zero-knowledge proof and without revealing any information about their identity. Hence, this technique does not impact users' privacy and does not strengthen the SP and IP.

B. Monitoring, Filtering and Blocking

As mentioned above, available applications designed to monitor NoSQL databases cannot detect and then disable malicious jobs and queries. The Kerberos central authentication system can be easily bypassed via advanced scripts, and in general, the level of monitoring is limited to data processing mainly in the API [15].

In a cloud environment, no information regarding the communication of nodes in the cluster or user connection details or data altering information (even editing or deleting), is recorded. In general, since there are no log files, a challenging problem is to identify incidents of data breach or malicious data loss in the cluster [13].

Real-time security mechanisms exist in big data technologies, resulting in high-speed data analysis. Therefore, the detection of anomalies is real-time implemented and the recording of security analytics can be frequently updated [12]. Some monitoring tools are available but are limited to controlling user requests at the API level. In general, neither the characteristics of a malicious query in big data technologies are defined, nor complete monitoring tools to disable these malicious queries, exist. One technique could be an initial authentication via Kerberos and in following, a second level authentication for accessing MapReduce [17].

V. Conclusions

In this paper, we have discussed major security concerns regarding NoSQL databases. Data protection and access control can be considered some of the key issues of security in NoSQL technology. Reasons for security threats in various NoSQL databases have also been thoroughly discussed in the current work, like privacy of user data, distributed environment, authentication, fine-grained authorization and access control, safeguarding integrity as well as protection of data at rest and in motion.

In NoSQL databases, Kerberos is used to authenticate the clients and data nodes. Specifically, in order to ensure fine-grained authorization, data are grouped according to their security level. On the other hand, Cassandra uses TDE technique to protect data at rest, whereas administrators must implement controls for ensuring that application and users have only access to the data they need in order to maintain a secure MongoDB deployment. Various techniques for mitigating the attacks on NoSQL databases have also been discussed

along with proposed security and privacy solutions of NoSQL databases.

REFERENCES

[1] M. Ahmadian, F. Plochan, Z. Roessler, and D. C. Marinescu. Securenosql: An approach for secure search of encrypted nosql databases in the public cloud. *International Journal of Information Management*, 37(2):63–74, 2017.

[2] E. Bangerter, J. Camenisch, and A. Lysyanskaya. A cryptographic framework for the controlled release of certified data. In *12th International Workshop on Security Protocols*, volume 3957, pages 20–42, 2004.

[3] E. Bertino and R. S. Sandhu. Database security-concepts, approaches, and challenges. *IEEE Transactions on Dependable and Secure Computing*, 2(1):2–19, 2005.

[4] A. Bhatewara and K. Waghmare. Improving network scalability using nosql database. *International Journal of Advanced Computer Research*, 2(4):488, 2012.

[5] S. Brands, L. Demuynck, and B. D. Decker. A practical system for globally revoking the unlinkable pseudonyms of unknown users. In *12th Australasian Conference on Information Security and Privacy (ACISP)*, volume 4586, pages 400–415, 2007.

[6] E. A. Brewer. Towards robust distributed systems. In *19th Annual ACM Symposium on Principles of Distributed Computing*, page 7, 2000.

[7] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A distributed storage system for structured data. *ACM Transactions on Computer Systems (TOCS)*, 26(2):4:1–4:26, 2008.

[8] P. Colombo and E. Ferrari. Fine-grained access control within nosql document-oriented datastores. *Data Science and Engineering*, 1(3):127–138, 2016.

[9] A. Davoudian, L. Chen, and M. Liu. A survey on nosql stores. *ACM Computing Surveys*, 51(2):40:1–40:43, 2018.

[10] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels. Dynamo: Amazon's highly available key-value store. In *21st ACM Symposium on Operating Systems Principles (SOSP)*, pages 205–220, 2007.

[11] S. Gilbert and N. A. Lynch. Brewer's conjecture and the feasibility of consistent, available, partition-tolerant web services. *SIGACT News*, 33(2):51–59, 2002.

[12] N. Gupta and R. Agrawal. Chapter four - nosql security. *Advances in Computers*, 109:101–132, 2018.

[13] V. N. Inukollu, S. Arsi, and S. R. Ravuri. Security issues associated with big data in cloud computing. *International Journal of Network Security and Its Applications (IJNSA)*, 6(3):45, 2014.

[14] N. Jatana, S. Puri, M. Ahuja, I. Kathuria, and D. Gosain. A survey and comparison of relational and non-relational database. *International Journal of Engineering Research and Technology (IJERT)*, 1(6):1–5, 2012.

[15] P. Kadebu and I. Mapanga. A security requirements perspective towards a secured nosql database environment. In *International Conference of Advance Research and Innovation*, 2014.

[16] N. Leavitt. Will nosql databases live up to their promise? *IEEE Computer*, 43(2):12–14, 2010.

[17] S. LLC. Securing big data: Security recommendations for hadoop and nosql environments. 2012.

[18] M. Mohamed, O. G. Altrafi, and O. Ismail. Relational vs. nosql databases: A survey. *International Journal of Computer and Information Technology*, 3(3):598–601, 2014.

[19] C. Nance, T. Losser, R. Iype, and G. Harmon. Nosql vs rdbms - why there is room for both. In *Southern Association for Information Systems Conference*, 2013.

[20] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov. Security issues in nosql databases. In *IEEE 10th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*, pages 541–547, 2011.

[21] R. P. Padhy, M. R. Patra, and S. C. Satapathy. Rdbms to nosql: Reviewing some next-generation non-relational database's. *International Journal of Advances in Engineering, Science and Technology (IJAEST)*, 11(1):15–30, 2011.

[22] E. Pattuk, M. Kantarcioglu, V. Khadilkar, H. Ulusoy, and S. Mehrotra. Bigsecret: A secure data management framework for key-value stores. In *6th IEEE International Conference on Cloud Computing*, pages 147–154, 2013.

[23] R. Poddar, T. Boelter, and R. A. Popa. Arx: A strongly encrypted database system. *IACR Cryptology ePrint Archive*, 2016:591, 2016.

[24] R. Pontes, F. Maia, J. Paulo, and R. M. P. Vilaça. Saferegions: Performance evaluation of multi-party protocols on hbase. In *35th IEEE Symposium on Reliable Distributed Systems Workshops (SRDS)*, pages 31–36, 2016.

[25] H. Shahriar and H. M. Haddad. Security vulnerabilities of nosql and sql databases for mooc applications. *International Journal of Digital Society (IJDS)*, 8(1), 2017.

[26] V. Sharma and M. Dave. Sql and nosql databases. *International Journal of Advanced Research in Computer Science and Software Engineering*, 2(8), 2012.

[27] X. Yuan, X. Wang, C. Wang, C. Qian, and J. Lin. Building an encrypted, distributed, and searchable key-value store. In *11th ACM on Asia Conference on Computer and Communications Security (AsiaCCS)*, pages 547–558, 2016.