IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 9, 2021, accepted February 18, 2021, date of publication February 24, 2021, date of current version March 5, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3062034

Privacy-Breaching Patterns in NoSQL Databases

KANIKA GOEL AND ARTHUR H. M. TER HOFSTEDE
Queensland University of Technology, Brisbane, QLD 4000, Australia
Corresponding author: Kanika Goel (k.goel@qut.edu.au)

ABSTRACT NoSQL databases disrupted the database market when first introduced. Their contemporary relevance has increased further in the era of big data due to the demands placed on (real-time) analytics. NoSQL databases are well placed to meet these demands due to their performance, availability, scalability, and storage solutions. Unfortunately, to achieve these features, compromises have been made with respect to security and privacy. Growing community awareness and unease combined with increased legislative requirements around data privacy have made such compromises less palatable, risky, or downright unacceptable. And though there is a growing body of knowledge related to data privacy in NoSQL databases, it is diverse and fragmented, and does not adequately address the challenges arising from the current environment. This paper aims to systematically examine various privacy weaknesses of NoSQL databases in the form of patterns. The patterns are shown to manifest themselves in well-known NoSQL databases and this evaluation can be used for benchmarking purposes. Through a survey it is demonstrated that the patterns have been observed in practice and are perceived as relevant. The pattern collection forms a repository of knowledge that can serve as a starting point for future privacy-related research for NoSQL databases through its identification of key problems, trade-offs, existing solution mechanisms, and its provision of terminology.

INDEX TERMS NoSQL, Databases, privacy, patterns.

I. INTRODUCTION

Technological advancements have resulted in an inconceivable growth in data. The massive amounts of data generated on an everyday basis today have become the wealth of organisations, harnessed for improved decision making. NoSQL databases are part of the aforementioned technological advancements. NoSQL database is a term used for modern web-scale databases that offer performance, storage, availability, and scalability [1]. These databases have become popular because of their simple yet flexible architecture and their ability to handle a large amount of unstructured data [2], [3]. However, to offer these benefits, NoSQL databases often compromise on privacy-related features [4]–[7].

Data privacy refers to having the privilege of control on how data is collected and disclosed. It is a 'data-owner-oriented' concept focusing on data owners, who may be individuals or groups, with the aim of maintaining their privacy when using their data for analytics [6], [8]. Not maintaining data privacy can have grave consequences. Recently, a university professor in Melbourne was forced to quit because of data related to Medicare and pharmaceutical benefits scheme of over 2.5 million Australians being re-identifiable when used for analysis. Using publicly available data, one could determine the identity of the individual in the dataset [9]. Societal concerns related to data privacy are reflected by an increasing number of privacy regulations surrounding the use of data and the occurrences of egregious and high profile privacy breaches. Clearly the need for data privacy cannot be ignored or downplayed so easily.

With the growing awareness of the need for data privacy, an emerging stream of privacy-related research can be observed in the field of NoSQL databases [10]. And while this is clearly a trend in the right direction, the research efforts are still quite diverse and fragmented and not yet amounting to a cohesive program of work. As such, no single resource exists that provides a comprehensive understanding and treatment of the various threats that NoSQL databases can be exposed to in relation to data privacy. Therefore, the research question we aim to address is, "*What data privacy-related issues exist in NoSQL databases?*" To address this question, a systematic analysis of privacy-related literature in the context of NoSQL databases is conducted. We use a patterns-based approach [11] to delineate core recurring privacy-breaching issues, which we refer to as *privacy-breaching patterns*. Patterns have the advantage of being specific to a problem at hand, but also general enough

The associate editor coordinating the review of this manuscript and approving it for publication was Vlad Diaconita.

to address future problems [12]. We collate and analyse the literature to provide pattern descriptions, their manifestation, their effect, methods of detection, and example strategies for mitigation. A patterns-based approach has been shown to be successful in capturing recurring phenomena [13], [14] and has the advantage of being technologically independent, sufficiently precise, and extensible.

The privacy-breaching patterns presented in this paper offer a repository of knowledge about a range of privacy-related issues that NoSQL databases can be exposed to. Expressing privacy-breaching issues as patterns also makes them more accessible to developers, administrators, and users in the area of NoSQL databases. The pattern collection enables benchmarking, i.e., selection of a suitable database for a particular application and provides a foundation for future work in the area of data privacy and NoSQL databases.

The remainder of the paper is organised as follows. Related work is highlighted in Section II. Patterns are introduced in section III. In Section IV, eight major NoSQL databases are evaluated in terms of their vulnerability to the patterns. An initial empirical evaluation is presented in Section V. This evaluation focuses on the pervasiveness and usefulness of the patterns. Section VI summarises the paper and provides avenues for future work.

II. BACKGROUND AND RELATED WORK

Recent technological developments have resulted in unprecedented volume of data generated by social networking websites, sensor networks, Internet, healthcare applications, and many other companies. This unstructured and voluminous amount of data that is generated is also referred to as big data [15]. Big data has become an active research area since the past few years, with data privacy being an important area of contribution [10].

A. DATA PRIVACY

Data privacy is the relationship between collection and dissemination of data, technology, and the public expectation of privacy, legal, and political issues surrounding them [16]. Data privacy enforces appropriate methods to collect, store, and analyse data such that sensitive information about users is not exposed. According to Katal *et al.* [17] user's privacy can be breached under the following circumstances:

- When personal information is combined with external datasets that may reveal new personal information about individuals.
- When personal information is analysed to add value to the business but it may result in knowing about personal characteristics of an individual.
- When appropriate data management and governance principles are not applied to sensitive user information.

For example, Amazon and Google can learn about the shopping preferences and browsing habits of users [18]. Location based service providers collect, store, and process information about the locations of the individual, which if not handled properly can reveal personal information [8]. Furthermore, with more data being available on clouds, data phishing is getting popular, which can breach the privacy of individuals [10]. According to a 2019 Cisco data privacy survey, 84% respondents commented that they care about their data privacy and 80% said that they were willing to take actions to protect it [19]. In addition to safeguarding data from these issues, there is also a need to comply with data protection regulations, such as general data protection regulation (GDPR) [20].

B. DATA PRIVACY IN RELATIONAL DATABASES

Relational databases are databases that store data in the form of tables. The columns hold the attributes of the table, and the rows contain the values of the attributes. Relational databases abide by ACID (atomicity, consistency, isolation, and durability) properties, which allows secure and reliable transactions. Relational databases were developed in the 1970s and focused on reducing data duplication [21].

Relational databases facilitate the implementation of reliable restrictions to ensure data privacy [22]. Relational database management systems have features such as role-based security, user-level permissions to allow access to data, transfer of encrypted messages, support to control access to a particular row or column, although these features require significant licensing fees [23]. Regardless, some data privacy issues such as SQL injections and insider attacks still exist, for which a number of techniques have and are being developed.

While relational databases are ubiquitous, they are not capable of handling huge amounts of unstructured data or big data. First, they have a rigid schema which data is expected to follow. Further, scalability is required to store (ETL) massive amounts of data. Relational databases offer vertical scalability, however, it can be expensive as it utilises many cores and/or CPUs that share RAM and disks [24]. Furthermore, the inbuilt security features and operations undertaken in the database (e.g., Cartesian product) make them slow on big data [25]. This led to the introduction of NoSQL databases.

C. DATA PRIVACY IN NoSQL DATABASES

With rise of big data, databases to store huge amount of data and process it effectively were required. This resulted in the introduction of NoSQL, or ''not only SQL'' databases. As opposed to relational databases, NoSQL databases are distributed in nature and offer high concurrent reading and writing with low latency, efficient big data storage, high scalability and availability, and lower management and operational costs [26]. Developed in late 2000s, NoSQL databases have a flexible schema and provide horizontal scalability, which is less expensive than vertical scalability that relational databases offer. However, in order to provide performance, availability, and scalability, NoSQL databases often compromise on privacy related features [3].

Greater community awareness of privacy-related issues as well as societal regulations on appropriate use of data have resulted in technical developments in the domain of NoSQL databases with a focus on security and privacy-related features. Prior research illustrates a range of security and privacy issues NoSQL databases are exposed to. While some papers focus on security issues related to particular NoSQL databases, such as MongoDB and Cassandra [3], [27], others provide a comprehensive overview of privacy-preserving mechanisms that can be deployed when analysing big data [2], [6]. Frameworks that support privacy-preserving big data analytics have also been proposed [28], [29], though not specific to NoSQL databases. These frameworks and techniques elaborate on the different data encryption techniques that can be used to store data (e.g., symmetric key encryption) and the privacy models that can be employed to prevent unauthorised access to data (e.g., k-anonymity, l-diversity) [28], [29].

There has been some research conducted specifically in the context of NoSQL databases on different techniques that can be used to detect and mitigate privacy-related issues. However, these techniques are database specific. For example, for detection, a technique to detect anomalous behaviour when conducting Map/Reduce queries on Hadoop has been proposed [30], [31], and for mitigation, a strong encryption technique to protect data at rest has been tested and advocated [32], [33]. Overall, prior work highlights the significance of data privacy in NoSQL databases, and this is evidenced by an increasing amount of published research in this area. However, the resulting work is scattered, not systematic, and does not yet add up to a coherent whole. To the best of our knowledge there is no resource which captures and systematically outlines the data privacy related issues faced by NoSQL databases.

In this paper, we review and synthesise prior literature to distill a number of privacy-breaching issues in NoSQL databases as privacy-breaching patterns. Together these form a repository for current knowledge in relation to data privacy issues in NoSQL databases. They provide a structure around key challenges and can be added to over time.

III. PRIVACY-BREACHING PATTERNS

This section presents six patterns that were distilled from the literature. The patterns were obtained through a narrative literature review of peer reviewed papers in the area of privacy and NoSQL databases. A narrative literature review aims to synthesise what has been written about a particular topic, which in this case is privacy issues in NoSQL databases. The initial keywords used to retrieve relevant articles were ''privacy'' AND ''NoSQL'' OR ''NoSQL databases''. The keyword ''NoSQL databases'' was also replaced with specific names of databases such as MongoDB and Cassandra as many studies were found to be conducted on particular databases. Recurring themes related to privacy-related aspects in NoSQL databases were collated and synthesised to reveal six privacy-breaching patterns. Documentation of NoSQL databases were also reviewed to complement the findings. Furthermore, since NoSQL databases are distributed systems, literature related to distributed systems, e.g., cloud computing and cloud architecture, was reviewed to augment our findings. Each pattern consists of six components: description (a brief explanation of the pattern), example (an illustration explaining the pattern), effect (impact of the pattern), manifestation (how the pattern can occur in NoSQL databases), detection (examples of how the pattern can be identified), and mitigation (examples of how the pattern can be prevented).

A. PATTERN 1 - MALICIOUS QUERY INTRODUCTION

Malicious Query Introduction happens when a person with malicious intent intrudes into the system and modifies a NoSQL query so that it can read or modify a NoSQL database or change data in a web application in an unintended manner. Malicious queries enable users to manipulate the back-end of NoSQL databases by adding, modifying, or deleting data. Malicious query modification in NoSQL databases can result from (i) NoSQL injection attacks [27] or (ii) insider attacks [34], [35].

Example: MongoDB uses the $ where operator (well-known from SQL) as a simple filter or to check for a constraint. An attacker could insert an arbitrary script (e.g. expressed in JavaScript) in the $ where clause and through its execution, triggered by evaluation of the query, obtain access to sensitive data. Furthermore, people internal to the organisation (e.g., contractors) may also gain access to sensitive information if appropriate access privileges are implemented.

Effect: Malicious Query Introduction may open up access to sensitive data to attackers or unauthorised people inside the organisation (insiders) through the use of arbitrary scripts. In both cases, the privacy of data is at risk. Malicious Query Introduction can also enable denial of service attacks. Furthermore, the injection of malicious code may restrict access to data for a long time [36]. For example, a command like 'while(1)' will force the target server to use 100% of its processor time to process the infinite loop thus inhibiting access to data.

Manifestation: Malicious Query Introduction can be a result of NoSQL injection attacks or insider attacks. NoSQL injection attacks involve injecting malicious code in the executable code allowing the attacker to gain authority and hence access to the information in the database. Hackers typically insert malicious code in the input boxes of web applications or in the URL of such applications [27]. Ron *et al.* [37] describe 5 classes of injection attacks relevant to NoSQL databases: (i) tautologies- injecting code in conditional statement, (ii) union queries- combines queries to bypass authentication and extract data, (iii) javascript injections- allows execution of javascript to perform complicated tasks, (iv) piggybacked queries- exploits assumptions in the interpretation of escape sequences' special characters to insert additional queries, and (v) origin violation- uses HTTP REST APIs to access database from other domain. In insider attacks, people in

the organisation abuse their power and use queries to gain unauthorised access to information [34].

Detection: Malicious Query Introduction remains a threat to the security of NoSQL databases, but there has been some prior work aiming to address the issue. In Islam *et al.* [38] a tool is described for detecting NoSQL injections using supervised learning. The tool was tested on MongoDB and CouchDB. They identify a number of features that point to NoSQL injection. In Eassa *et al.* [39] a method named DNIARS is proposed that compares patterns from static and dynamic NoSQL statement structures to determine the possibility of a NoSQL injection attack. DIGLOSSIA is introduced in [40] and it identifies tainted characters in a query as indicators of a possible NoSQL injection or insider attack. Furthermore, in Sauvanaud *et al.* [41] an anomaly detection system is proposed for cloud services based on machine learning algorithms. It is designed to detect errors related to erroneous behaviour of the service and SLA violations in a cloud service, which is useful in identifying insider attacks.

Mitigation: Developers of a database can add extra functionality to validate input before it is being processed [27]. For example, if an input box expects seven digits, then it should restrict the input to data type integer and enforce a length of seven. This would help in preventing NoSQL injection attacks. Additionally, minimising privilege for the admin account is advocated so that even if an intruder attacks the system and compromises this account, data access is still restricted [34], which would assist in addressing insider attacks.

B. PATTERN 2 - ACCIDENTAL RE-IDENTIFICATION

Accidental Re-identification happens when an individual can be re-identified based on the output of a query, despite mechanisms implemented to ensure privacy of data. This pattern can happen through the execution of a sophisticated query which results in a small output set where re-identification of one or more individuals is possible. Naturally, this raises privacy concerns. MapReduce or Aggregation framework [42] (an expressive query mechanism provided by MongoDB) provide ways in which these types of sophisticated queries can be expressed.

Example: MapReduce is a popular data processing technique, which consists of two main functions: mappers, which split an input job into smaller manageable units, and reducers, which process the smaller jobs and write the results in an output file. The output of a reducer can directly leak sensitive information as it contains a global view of the final computation [4]. For example, one may wish to know the department, age, residential suburb, and medical details of all employees with a salary greater than $200,000 in an organisation. The output of the mapper would be the department, age, residential suburb, and medical details of all employees with a salary greater than $200,000. The reducer then aggregates the results. If the result consists of very few people only then people could be re-identified based on other details such as age and residential suburb.

Effect: Accidental Re-identification can result in disclosure of sensitive information stored in the database. Considering the number and variety of mappers and reducers and the fact that the programmer has no control over the intermediate output, the computed output from reducers may breach the privacy of user data. Composite attacks can also occur, where the adversary may link the output to some publicly available information on Facebook or Twitter [43].

Manifestation: Accidental Re-identification results from the usage of advanced queries or data processing abilities such as MapReduce queries and Aggregation Framework. MapReduce queries that run on NoSQL databases such as MongoDB, Cassandra, or used with Hadoop, can have this effect. Another similar querying capability is offered by MongoDB, known as the Aggregation framework, which can summarise huge amounts of data [44] and can result in disclosing sensitive data.

Detection: For this pattern, the output is dependent on the content of the data set. Detection relies on a high degree of familiarity with the data set as one can then envisage problematic output [4]. However, this may not be practical and also disregards the need for querying capabilities as offered by frameworks such as MapReduce. Another way of dealing with this pattern is to control access to the output and only make this more widely available if it is safe to do so.

Mitigation: Privacy-preserving techniques, which include generalisation, anonymisation, suppression, and encryption [2], can be deployed in the data set to ensure that even if the analysis returns specific values, they cannot be related to unique individuals. Furthermore, different privacy models have been proposed such as k-anonymity, l-Diversity, t-Closeness, and e-Differential privacy [29]. These can be applied to various attributes of the data set.

C. PATTERN 3 - WEAK AUTHENTICATION

Authentication is the process of verifying the identity given to users to access data and resources stored in a system, e.g., a NoSQL database [45]. It is an important process for data privacy as it prevents illegitimate access to data. NoSQL databases typically provide Weak Authentication to users because of (i) poor password storage mechanisms [46], and (ii) limited to no authentication capabilities [47].

Example: By default, authentication is disabled in Redis. When enabled in the configuration file, the database sends passwords in an unencrypted format [48], raising concerns regarding authentication capabilities of the system. On the contrary, MongoDB provides strong password storage mechanism, but does not enforce authentication in the sharded mode.

Effect: Weak Authentication provides an intruder access to the database. Once the access is gained the intruder can conduct intentional attacks such as masquerade attacks. In this case a node (intruder) appears to be like other nodes accessing the system and it uses this as an opportunity to learn more about the system [49]. Another possibility is that of a

hijacking attack, where the intruder gains access to the database and uses it to satisfy their own objectives [50].

Manifestation: Weak Authentication in NoSQL databases happens due to limited authentication mechanisms in standalone mode, limited or no authentication in sharded mode, and weak password storage mechanisms. Most NoSQL databases have limited default authentication features, rely on plaintext for communicating between the client and the server, and use limited external encryption tools [46], [51]. For instance, MongoDB and Redis do not provide authentication by default, allowing an intruder to get access to the system [1]. There is no data encryption performed during communication between the client and server of Redis, as well as other servers within the same or different cluster [25]. This means that an intruder can gain access to the username and password and hence the database, if the channel is sniffed. Furthermore, Cassandra saves password using plaintext or MD5 hash [52], which is a very weak password storage mechanism, enabling authentication attacks.

Detection: Weak Authentication can result in an individual with malicious intent gaining access to the system which may culminate into attacks such as masquerade and hijacking. For example, an attacker can get access to valid user credentials and not be an administrator to carry out an attack in MongoDB [53]. The presence of such attacks should indicate a bypass through the authentication mechanism of the database. Furthermore, the logs of NoSQL databases may be audited on a periodic basis to detect illegitimate access to data.

Mitigation: A careful examination of the documentation of a NoSQL database is required to understand the authentication features provided. For example, Cassandra by default has the setting *Allowallauthenticator* turned on, but this needs to be turned off to enable authentication services [45]. Being aware of default settings and available features, and configuring them depending on planned use can enable a minimal required level of authentication measures to be in place. Further, privacy-preserving mechanisms are suggested to ensure that sensitive information is not leaked even if an intruder gets access to data. For example, data anonymisation or generalisation options should be explored [28]. Additionally, Chang *et al.* [54] proposes a cloud computing adoption framework that provides guidelines to overcome technical challenges when using the cloud to deliver services, e.g., use of directory and lightweight directory protocol (LDAP) for access management.

D. PATTERN 4 - COARSE-GRAINED ACCESS CONTROL

Access control or authorisation is the process of controlling access to data and resources in a system [55]. It is usually performed by associating different types of users with certain sets of rules based on their roles and responsibilities [45]. NoSQL databases provide varying degrees of support for access control. Some types of support for access control are role-based, while others are data-based. They can also be combined (e.g., all users with manager privileges can access salary information).

Example: In MongoDB, authorisation is not enabled by default and there is no support for authorisation in sharded mode [3]. When authorisation is enabled, it provides two roles to the users: ready-only or read-and-write. The next option is to provide no access at all. A person with read-only privileges can access the entire data set, whereas a person with read-and-write privileges can make changes to the entire dataset. In either case, failure to have more granular access control places the privacy of data at risk. Additionally, specific role-based and data-based access is very weak in NoSQL databases. For instance, an organisation may want two specific data collections in a database to be accessed by an employee living in New York, working in the finance department, and having at least 10 years experience. This kind of fine-grained role-based access can only be achieved in MongoDB through extensions realised at the code level.

Effect: This pattern can provide illegitimate access to sensitive data, which can be used for malicious purposes. Access to confidential information represents a risk to the privacy of data and has a potential for perpetrator to conduct malicious crimes such as data theft, identity theft, monetary theft, and more [56].

Manifestation: Many NoSQL databases do not provide access control features by default and a user needs to enable them. In addition, there are limited role-based access control features [57]. Fine-grained role-based access control mechanisms are still in a very early stage for NoSQL databases and very few control frameworks have been proposed for document-oriented and column-family databases [58]. Furthermore, access to data and resources (data-based access) is at high levels of granularity [55]. For example, in MongoDB the lowest level of data-access is at the collection level. Providing access to particular documents in a collection requires programmatic changes to the database [59].

Detection: Unauthorised access to data can be detected through observing consequences of attacks by people with malicious intent, for example, monetary theft or identity theft [56]. Kholidy and Baiardi [60] proposed an intrusion detection system that can be deployed in clouds to deal with masquerade attacks where an intruder may impersonate a legitimate user. Gupta *et al.* [57] suggested the use of activity monitoring systems for NoSQL databases enabling monitoring of all activities of the system, creation of audit reports, keeping an eye on database access, and generating security alerts. Given the ubiquitous need of monitoring, there are many external applications available in the market that enable detection of inappropriate access, such as 'agentless monitor' by AppPerfect [61].

Mitigation: Authorisation features need to be enabled in a NoSQL database. Next, it is essential that fine-grained access control rules are defined in order to access data and resources. For example, while MongoDB does not support this directly, the use of a RESTful API behind a reverse proxy (a server that sits behind a firewall) can enable fine-grained permissions on the proxy server [3]. Furthermore, new techniques can be integrated into NoSQL databases. Moreno *et al.* [51]

proposed a technique for key-value databases in which data is labelled with authorisation rights, enabling appropriate access to data. Imam *et al.* [62] put forward a technique, which automatically suggests a schema in the initial phases of database development, which in turn improves the security of the system. This initial schema can be considered equivalent to the notion of views in relational databases. Ulusoy *et al.* [63] presented a framework to enforce fine-grained access control within Hadoop.

E. PATTERN 5 - VULNERABLE DATA IN MOTION

NoSQL databases store data in a distributed manner in their globally deployed shards, which is why computations take place in a distributed environment. Data in motion in such environment poses a risk to the privacy of that data because of poor security mechanisms employed by the cluster, which may compromise cluster operations (e.g., computations). While this is not under the control of the database, the secure transfer of data to the cluster and among the various shards is.

Example: Many companies such as Google, Yahoo, Facebook, and more, use MapReduce. The ANZ Bank is using MapReduce to better target customer needs [64]. MapReduce performs computations in a distributed manner. It is possible that a cluster becomes compromised because of poor security measures, resulting in the introduction of malicious mappers. In this case sensitive financial data of customers may leak thus potentially compromising their privacy.

Effect: Data in motion in distributed environments results in loss of administrative control [55]. Organisations that are storing data in public clouds assume that the cloud providers involved have a trustworthy system administrator and secure data management including maintenance controls. However, a breach of such controls may result in malicious attacks jeopardising the privacy of data [5], [64]. Data in motion also results in an opaque physical architecture with the user having no knowledge about the cloud's architecture [31]. When the nodes are malfunctioning, this can result in a number of attacks such as an impersonation attack, when an adversary pretends to be a legitimate user and performs computations that may result in data leakage [65] or a denial-of-service attack when the adversary becomes non-functional [66].

Manifestation: In a distributed environment, NoSQL databases run on several nodes. The higher the number of nodes the greater the opportunity for attacks as there are more entry points for intruders [34]. Therefore, computations performed in such a distributed environment using data processing techniques such as MapReduce, can be altered with the assistance of malfunctioning nodes (adversaries). Furthermore, rogue data nodes could be added to the cluster to corrupt the environment and gain access to sensitive data [31]. Additionally, the transfer of data from one shard to another needs to be secure. Currently NoSQL databases use frail encryption techniques for data in transit, which can result in data getting exposed if an intruder gains access to the network [47].

Detection: With the prominence of privacy issues in distributed environments, certain algorithms and techniques to detect compromised nodes in the MapReduce framework are proposed: a) semantic analysis of system calls and logs to detect misuse and attacks [65], and b) a computational provenance system that analyses provenance of computational tasks to track user data and computations and also detects anomalies that may have compromised these computations [30]. Additionally, a multistage separate query processing protocol that aggregates data from distributed databases using homomorphic encryption has been proposed to combat data privacy issues [67].

Mitigation: While distributed storage and computation is an advantage of NoSQL databases, certain strategies should be undertaken to mitigate the privacy risks associated with data in motion. Examples are: a) a model allowing client to read the data from any storage node(s), provided only those clients who have been granted access to the datum by access control policy hold the encryption key enabling them to decrypt the data [68], b) two secure and practical solutions (shuffle-in-the-middle and shuffle & balance) to prevent leakage of data from MapReduce operations observing the intermediate transfer between mappers and reducers [69]. Shuffle-in-the-middle securely shuffles all key-value pairs produced by mappers and consumed by reducers, whereas shuffle & balance prevents the adversary from observing the volume of data transferred between mappers and reducers [69], c) using network encryption and using better data encryption techniques when communicating with client and other clusters [47], and d) employing different encryption techniques such as DES, AES, Blowfish, RC5 and Idea for secure data transmission [70].

F. PATTERN 6 - VULNERABLE DATA AT REST

While data is mostly in motion in NoSQL databases because of the distributed nature of computations, there are times when data is at rest. NoSQL databases have been designed to provide high availability of and powerful processing capabilities for big data compared to relational databases [34]. However, this means compromising the security of data at rest as access needs to be quick and unimpeded. Balancing availability and processing power on the one hand with security of data at rest on the other is thus an issue for NoSQL databases [71] and may manifest as weak encryption of data at rest or use of data storage mechanisms susceptible to external attacks.

Example: Redis is a key-value database and is often used for session storage because of the speed it provides. To provide this feature, the database does not encrypt data at rest. This data is stored as plain text, which raises privacy concerns as all intruders need to do is to gain access in order to compromise the data. This may reveal sensitive information of users stored in the database. In this case, Redis prioritises processing power over security of data at rest.

Effect: The impact of this pattern is either on processing power of the database or the security and in turn the privacy

of data. If complex storage mechanisms and data encryption techniques are used, operational complexity would be increased negatively impacting the processing power of the database. However, if processing power is to be maintained the security of data is at risk. In Tang *et al.* [72] various issues resulting from poor storage of data at rest are identified, such as data loss, data breaches, and malicious insider attacks.

Manifestation: Balancing processing power and level of security of data at rest is a consequence of the need to encrypt data at rest and the usage of new data storage mechanisms to handle big data [73]. Traditionally, multi-tiered storage was used to handle data, with the database administrator having total control over the tiers. However, with the need to handle exponentially growing data, auto-tiering is adopted by NoSQL databases [74], [75]. Auto-tiering places data of frequent use in the upper tiers and less frequent data in lower tiers. The security of tiers decreases from top to bottom. However, the challenge here is that the database administrator has no control over where data is stored. Furthermore, there is dependence on external service providers as the cloud vendor uses their own security protocols for the cloud where the data is stored [76], who can also get clues based on data transmission among the tiers. This poses a risk to data confidentiality and integrity. Additionally, the lowest tier has the least number of security features enabled [77] even though it is possible that that layer contains sensitive information, thus risking data privacy. Moreover, most NoSQL databases store data at rest in plaintext format [28], [34]. This is because encryption and decryption of data typically requires the use of complex algorithms which in turn increases operational complexity. However, this raises privacy concerns regarding data at rest.

Detection: Detection of inappropriate access to data due to poor data storage services or weak data encryption can be achieved through the monitoring of issues such as data loss, data breaches, or other malicious attacks. A cloud data security model has been proposed [78] based on the business process model and notation (BPMN) where simulation results can reveal data security issues. There are four lanes: one for users, one for data request messages, one for cloud data centres, and one for intrusion detection. It is the fourth lane which provides warning signs and indicates the need for taking corrective actions.

Mitigation: Research suggests storing data at rest in an encrypted manner and using a strong encryption technique, e.g., AES encryption technique [32], [33]. A three layer framework has been proposed [47], which outlines techniques and mechanisms important for developing a protected environment. With the implementation of proper mechanisms for a protective environment along with strong encryption of data at rest, privacy of data at rest can be improved.

IV. SYSTEM EVALUATION

We chose eight widely used NoSQL databases, two from each category (i.e., key-value, document-oriented, graph, and column-family) to validate the identified privacy-breaching patterns. The databases used were: MongoDB 4.4.2, CouchDB 3.1.1, Redis 6.0.9, Aerospike 4.8.0.3, Cassandra 3.11.9, HBase 2.3.2, Neo4j 4.1.4, and OrientDB 3.1.0. Table 1 summarises our analysis across all privacy-breaching patterns, where '+' indicates that none of the manifestations of the pattern can be directly mitigated (other than e.g., through programmatic extensions) by the database and '+/−' indicates that some (but not all) manifestations of the pattern can be mitigated or that some manifestations cannot be fully mitigated.

A. MALICIOUS QUERY INTRODUCTION

This pattern is evidenced to occur as a NoSQL injection attack and insider attack in *all* selected databases (−).

B. ACCIDENTAL RE-IDENTIFICATION

This pattern can occur in all databases except Redis (+) and Aerospike (+). The use of aggregation framework in MongoDB and the use of MapReduce framework in MongoDB and other databases (−) may result in the manifestation of Accidental Re-identification.

C. WEAK AUTHENTICATION

In MongoDB, authentication features are disabled by default. Password is stored using salted challenge response authentication mechanism (SCRAM) [79], which provides SCRAM-SHA-1 and SCRAM-SHA-256 hashing features (+). CouchDB does not provide authentication by default, but supports password based or cookie based authentication, and uses the PBKDF2 hashing algorithm [80] to store passwords (+). When enabled, Redis supports password based authentication, however, the password is saved in plaintext (+/−). Aerospike uses password based authentication control and supports external authentication, such as LDAP [81]. Nonetheless, a random hashing technique is used to store passwords (+/−). Cassandra supports simple password based authentication (+/−). HBase supports Kerberos [82] authentication (+), when configured in the database. Neo4j does not provide authentication by default, but supports password based authentication. It uses SHA-256 [83] to store passwords, susceptible to dictionary and brute force attacks (+/−). OrientDB supports password based authentication and stores passwords using PBKDF2 (+).

D. COARSE-GRAINED ACCESS CONTROL

When enabled, MongoDB supports role-based access control, however, data-based access control is poor (+/−). CouchDB enables access control at the level of only two roles, admin and members. The granularity is at database level (−). Redis does not implement access control at any layer (−). Aerospike enables creation of roles with seven pre-specified privileges. The maximum data-based granularity is at namespace level (+/−). By default, access control is disabled in Cassandra. When enabled, it supports creation of roles and granting privileges at keyspace level (+/−). HBase allows configuring role-based and data-based access control.

TABLE 1. System evaluation results.

Pattern	MongoDB	CouchDB	Redis	Aerospike	Cassandra	HBase	Neo4j	OrientDB
Malicious Query Introduction	-	-	-	-	-	-	-	-
Accidental Re-identification	-	-	+	+	-	-	-	-
Weak Authentication	+	+/-	+/-	+/-	+/-	+	+/-	+
Coarse-grained Access Control	+/-	-	-	+/-	+/-	+	+	+
Vulnerable Data at Rest	+	+	+	+	+	+	+	+
Vulnerable Data in Motion	+	-	+/-	+	+	+	+	+

Cell-level access control is supported in HBase (+). When enabled, Neo4j (+) and OrientDB (+) supports creation of fine-grained role-based and data-based access controls (+).

E. VULNERABLE DATA IN MOTION

MongoDB supports transport layer security/secure socket layers (TLS/SSL) to encrypt all of MongoDB's network traffic. Further, though not enabled by default, MongoDB supports keyfiles or X.509 certificates for communication between clients and clusters (+). CouchDB supports X.509 certificates for TLS to encrypt data in motion (+). Redis does not support encryption of data in motion by default, but provides built in encryption for data in motion in its enterprise version, and supports TLS and X.509 certificates (+). Aerospike supports TLS (+), and when enabled, Cassandra (+) and HBase (+) support TLS/SSL for secure transmission of data. Neo4j (+) and OrientDB (+) support configuration of SSL/TLS to protect data in motion.

F. VULNERABLE DATA AT REST

MongoDB uses GridFS for storing large files and supports encryption of data at rest, but only in its enterprise version; using AES 256-CBC (+). CouchDB does not support encryption of data at rest (−). Redis provides built in encryption capability for data at rest, but only for replication groups, and not for data present in-memory (+/−). Aerospike uses a hybrid in-memory storage architecture and provides an option to encrypt data at rest using the AES-128 (+). Cassandra uses tiered storage and encrypts data at rest using AES-256 (+). HBase supports Clouders and HDFS data encryption. HBase stores data in regions, which is saved in different clusters (+). Neo4j stores data in layers but does not allow encryption at database level (+/−). OrientDB also supports encryption of data at rest at database level using AES or DES (+).

On analysis of Table 1, it is evident that graph databases (i.e., Neo4j and OrientDB) are resistant to most privacy-breaching patterns, which may be attributed to their architecture being quite similar to relational databases. Inspection of other databases conveys that MongoDB is more advanced in data privacy-related features. Our analysis also reveals that most features are enabled for enterprise (i.e., paid) versions. Furthermore, it is imperative that the documentation of each database is reviewed before it is used, as we found that some databases do not have complete documentation which may jeopardise the understanding of available features in a database. For example, NCache and Voldemort database do not provide detailed information regarding the privacy-breaching issues mentioned in this paper.

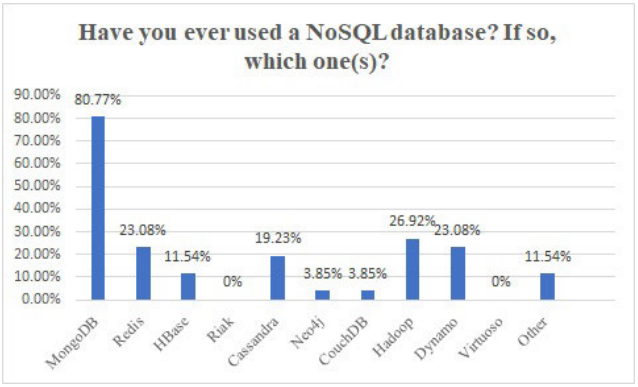

FIGURE 1. Databases that the participants had experience with.

V. EMPIRICAL EVALUATION

To empirically assess the privacy-breaching patterns, we focused on their pervasiveness and utility. Pervasiveness assists an understanding of how susceptible the databases are to the identified patterns and utility provides knowledge about how significant it is to address these patterns for data privacy.

We collected responses from participants who have experience in the area of NoSQL databases. A questionnaire was distributed through LinkedIn groups and also by email to known contacts. The questionnaire started with some demographic questions, such as NoSQL databases used, administered, and created by the participants, and was then followed by questions that aimed to evaluate the pervasiveness and utility of the 6 identified patterns. A 5 point Likert scale was used to obtain the responses of the participants.

We received 28 responses. The responses convey that the participants had experience with varied NoSQL databases and frameworks, which includes MongoDB, Redis, CouchDB, Azure, Hadoop, Cassandra, and DynamoDB. Figure 1 provides an overview of the databases the participants had experience with. Further, 100% of participants had created a NoSQL database, whereas, 71.4% participants had administered a NoSQL database.

78.5% respondents conveyed that privacy of data is a concern for NoSQL databases. Next, we retrieved and summarised the responses for all patterns. The responses communicate that at least 58% of the respondents have encountered the patterns either often, sometimes, or rarely (see Figure 2), validating the presence of the patterns across a wide range of NoSQL databases.

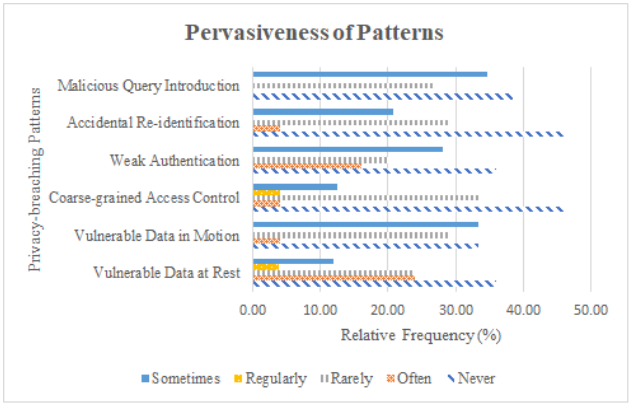

FIGURE 2. Pervasiveness of patterns.

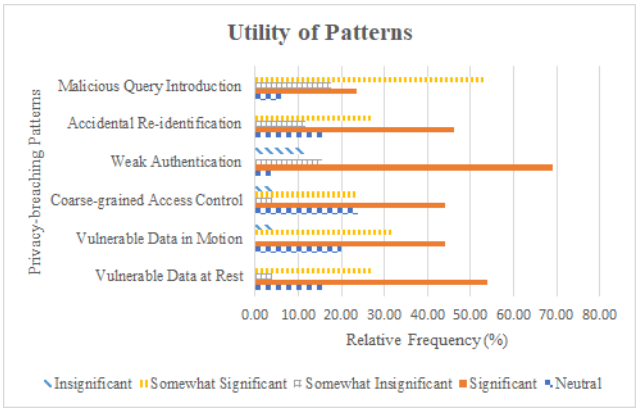

FIGURE 3. Utility of Patterns.

Additionally, we also observe from Figure 3 that overall 89% of the participants find the patterns significant to understand data privacy issues in NoSQL databases. The results convey the pervasiveness and utility of the patterns, thereby validating their value as a collection.

VI. CONCLUSION

Data privacy is of growing concern, which has resulted in an emerging stream of privacy-related research in NoSQL databases. However, this research is diverse and fragmented. This paper used patterns based approach to review and synthesise prior literature, and distil six privacy-breaching patterns, which explain the privacy related vulnerabilities in NoSQL databases, their manifestations, example detection techniques, and possible mitigation strategies. The survey results reveal the pervasiveness and utility of the patterns. The patterns presented in this paper have been derived after collating issues that have impacted multiple types of NoSQL databases. Most of these issues have not been documented before. They may be part of the folklore of the database community or are issues which have been addressed well by some of the existing NoSQL databases – neither of which is accessible to NoSQL database users. We capture these issues in a new and accessible way, a catalogue of privacy-breaching patterns with a consistent format. The patterns provide a common point of reference and terminology, which enables smooth communication about privacy breaching patterns in NoSQL databases. The patterns offer a repository of knowledge and contribute to two areas of growing significance: privacy and NoSQL databases. They provide guidance on how NoSQL databases can be extended to address the privacy-breaching issues they are susceptible to. The patterns can also be used for benchmarking purposes, an example of which is present in Section IV.

We also acknowledge certain limitations. First, the patterns are literature-based, however, we validate them through a survey and also test them across eight NoSQL databases. Second, the number of survey responses (28) may not be considered optimal, however, they were sufficient to comment on the pervasiveness and utility of patterns. Finally, we acknowledge that the patterns may not be complete, we present certain examples extracted from literature (e.g., for detection and mitigation). Nonetheless, the patterns provide a solid foundation to advance research in privacy and NoSQL databases.

The patterns also open various avenues for future research. First, the patterns can be refined and extended using other forms of validation, such as qualitative feedback. Second, each pattern can be studied in further detail to come up with detection and mitigation techniques that can be used by multiple NoSQL databases. Finally, the paper shows the utility of a patterns-based approach. This can be used to understand other significant but under-explored areas in NoSQL databases.

REFERENCES

[1] A. Zahid, R. Masood, and M. A. Shibli, "Security of sharded NoSQL databases: A comparative analysis," in *Proc. Conf. Inf. Assurance Cyber Secur. (CIACS)*, Jun. 2014, pp. 1–8.

[2] B. B. Mehta and U. P. Rao, "Privacy preserving unstructured big data analytics: Issues and challenges," *Procedia Comput. Sci.*, vol. 78, pp. 120–124, Dec. 2016.

[3] L. Okman, N. Gal-Oz, Y. Gonen, E. Gudes, and J. Abramov, "Security issues in NoSQL databases," in *Proc. IEEE 10th Int. Conf. Trust, Secur. Privacy Comput. Commun.*, Nov. 2011, pp. 541–547.

[4] K. Grolinger, M. Hayes, W. A. Higashino, A. L'Heureux, D. S. Allison, and M. A. M. Capretz, "Challenges for MapReduce in big data," in *Proc. IEEE World Congr. Services*, Jun./Jul. 2014, pp. 182–189.

[5] P. Derbeko, S. Dolev, E. Gudes, and S. Sharma, "Security and privacy aspects in MapReduce on clouds: A survey," *Comput. Sci. Rev.*, vol. 20, pp. 1–28, May 2016.

[6] H.-Y. Tran and J. Hu, "Privacy-preserving big data analytics a comprehensive survey," *J. Parallel Distrib. Comput.*, vol. 134, pp. 207–218, Dec. 2019.

[7] E. Sahafizadeh and M. A. Nematbakhsh, "A survey on security issues in big data and nosql," *Adv. Comput. Sci., Int. J.*, vol. 4, no. 4, pp. 68–72, 2015.

[8] T. Wang, Z. Zheng, M. H. Rehmani, S. Yao, and Z. Huo, "Privacy preservation in big data from the communication perspective—A survey," *IEEE Commun. Surveys Tuts.*, vol. 21, no. 1, pp. 753–778, 4th Quart., 2018.

[9] V. Teague, *Melbourne Professor Quits After Health Department Pressures her Over Data Breach*. London, U.K.: The Guardian, 2020. [Online]. Available: https://www.theguardian.com/australia-news/2020/mar/08/melbourne-professor-quits-after-health-department-pressures-her-over-data-breach

[10] M. Binjubeir, A. A. Ahmed, M. A. B. Ismail, A. S. Sadiq, and M. K. Khan, "Comprehensive survey on big data privacy protection," *IEEE Access*, vol. 8, pp. 20067–20079, 2020.

[11] D. Lea, "Christopher Alexander: An introduction for object-oriented designers," *ACM SIGSOFT Softw. Eng. Notes*, vol. 19, no. 1, pp. 39–46, 1994.

[12] E. Gamma, *Design Patterns: Elements of Reusable Object-Oriented Software*. London, U.K.: Pearson, 1995.
[13] S. Suriadi, R. Andrews, A. H. M. ter Hofstede, and M. T. Wynn, "Event log imperfection patterns for process mining: Towards a systematic approach to cleaning event logs," *Inf. Syst.*, vol. 64, pp. 132–150, Mar. 2017.
[14] W. M. P. van der Aalst, A. H. M. ter Hofstede, B. Kiepuszewski, and A. P. Barros, "Workflow patterns," *Distrib. Parallel Databases*, vol. 14, no. 1, pp. 5–51, 2003.
[15] S. Yu, "Big privacy: Challenges and opportunities of privacy study in the age of big data," *IEEE Access*, vol. 4, pp. 2751–2763, 2016.
[16] L. Xu, C. Jiang, J. Wang, J. Yuan, and Y. Ren, "Information security in big data: Privacy and data mining," *IEEE Access*, vol. 2, pp. 1149–1176, 2014.
[17] A. Katal, M. Wazid, and R. H. Goudar, "Big data: Issues, challenges, tools and good practices," in *Proc. 6th Int. Conf. Contemp. Comput. (IC)*, Aug. 2013, pp. 404–409.
[18] A. Mehmood, I. Natgunanathan, Y. Xiang, G. Hua, and S. Guo, "Protection of big data privacy," *IEEE Access*, vol. 4, pp. 1821–1834, 2016.
[19] Cisco. (2019). *Consumer Privacy Survey Cisco Cybersecurity Series*. [Online]. Available: https://www.cisco.com/c/dam/en/us/products/collateral/security/cybersecurityseries-2019-cps.pdf
[20] A. Dasgupta, A. Q. Gill, and F. Hussain, "A review of general data protection regulation for supply chain ecosystem," in *Proc. Int. Conf. Innov. Mobile Internet Services Ubiquitous Comput.* Cham, Switzerland: Springer, 2019, pp. 456–465.
[21] D. Maier, *The Theory of Relational Databases*, vol. 11. Rockville, MD, USA: Computer Science Press, 1983.
[22] A. Meier and M. Kaufmann, *SQL & NoSQL Databases : Models, Languages, Consistency Options and Architectures for Big Data Management*. Wiesbaden, Germany: Springer Vieweg, 2019.
[23] A. Bamrara, "Evaluating database security and cyber attacks: A relational approach," *J. Internet Banking Commerce*, vol. 20, no. 2, pp. 1–17, 2015.
[24] R. Cattell, "Scalable SQL and NoSQL data stores," *ACM SIGMOD Rec.*, vol. 39, no. 4, pp. 12–27, May 2011.
[25] P. Noiumkar and T. Chomsiri, "A comparison the level of security on top 5 open source NoSQL databases," in *Proc. 9th Int. Conf. Inf. Technol. Appl. (ICITA)*, 2014, pp. 1–7.
[26] J. Han, H. E. G. Le, and J. Du, "Survey on NoSQL database," in *Proc. 6th Int. Conf. Pervas. Comput. Appl.*, Oct. 2011, pp. 363–366.
[27] B. Hou, K. Qian, L. Li, Y. Shi, L. Tao, and J. Liu, "MongoDB NoSQL injection analysis and detection," in *Proc. IEEE 3rd Int. Conf. Cyber Secur. Cloud Comput. (CSCloud)*, Jun. 2016, pp. 75–78.
[28] P. Jain, M. Gyanchandani, and N. Khare, "Big data privacy: A technological perspective and review," *J. Big Data*, vol. 3, no. 1, p. 25, Dec. 2016.
[29] B. B. Mehta, U. P. Rao, N. Kumar, and S. K. Gadekula, "Towards privacy preserving big data analytics," in *Proc. 6th Int. Conf. Adv. Comput. Commun. Technol.*, Rohtak, India, 2016, pp. 28–35.
[30] C. Liao and A. Squicciarini, "Towards provenance-based anomaly detection in MapReduce," in *Proc. 15th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2015, pp. 647–656.
[31] S. N. Khezr and N. J. Navimipour, "MapReduce and its applications, challenges, and architecture: A comprehensive review and directions for future research," *J. Grid Comput.*, vol. 15, no. 3, pp. 295–321, Sep. 2017.
[32] V. R. Pancholi and B. P. Patel, "Enhancement of cloud computing security with secure data storage using AES," *Int. J. Innov. Res. Sci. Technol.*, vol. 2, no. 9, pp. 18–21, 2016.
[33] M. Derfouf, A. Mimouni, and M. Eleuldj, "Vulnerabilities and storage security in cloud computing," in *Proc. Int. Conf. Cloud Technol. Appl. (CloudTech)*, Jun. 2015, pp. 1–5.
[34] P. Raj and G. C. Deka, *A Deep Dive into NoSQL Databases: The Use Cases and Applications*, vol. 109. New York, NY, USA: Academic, 2018.
[35] G. Kul, S. Upadhyaya, and A. Hughes, "Complexity of insider attacks to databases," in *Proc. Int. Workshop Manag. Insider Secur. Threats*, Oct. 2017, pp. 25–32.
[36] H. Shahriar and H. M. Haddad, "Security vulnerabilities of NoSQL and SQL databases for MOOC applications," *Int. J. Digit. Soc.*, vol. 8, no. 1, pp. 1244–1250, Mar. 2017.
[37] A. Ron, A. Shulman-Peleg, and A. Puzanov, "Analysis and mitigation of NoSQL injections," *IEEE Security Privacy*, vol. 14, no. 2, pp. 30–39, Mar. 2016.
[38] M. R. U. Islam, M. S. Islam, Z. Ahmed, A. Iqbal, and R. Shahriyar, "Automatic detection of NoSQL injection using supervised learning," in *Proc. IEEE 43rd Annu. Comput. Softw. Appl. Conf. (COMPSAC)*, Jul. 2019, pp. 760–769.
[39] A. M. Eassa, M. Elhoseny, H. M. El-Bakry, and A. S. Salama, "NoSQL injection attack detection in Web applications using RESTful service," *Program. Comput. Softw.*, vol. 44, no. 6, pp. 435–444, Nov. 2018.
[40] S. Son, K. S. McKinley, and V. Shmatikov, "Diglossia: Detecting code injection attacks with precision and efficiency," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur. (CCS)*, 2013, pp. 1181–1192.
[41] C. Sauvanaud, M. Kaâniche, K. Kanoun, K. Lazri, and G. Da Silva Silvestre, "Anomaly detection and diagnosis for cloud services: Practical experiments and lessons learned," *J. Syst. Softw.*, vol. 139, pp. 84–106, May 2018.
[42] E. Plugge, D. Hows, P. Membrey, and T. Hawkins, *The Definitive Guide to MongoDB: A Complete Guide to Dealing With Big Data Using MongoDB*. New York, NY, USA: Apress, 2015.
[43] S. U. Bazai, J. Jang-Jaccard, and X. Zhang, "Scalable big data privacy with MapReduce," in *Encyclopedia Big Data Technol.*, vol. 1. Cham, Switzerland: Springer, pp. 1454–1462, Mar. 2019.
[44] D. Merriman, E. Horowitz, and C. T. Westin, "Aggregation framework system architecture and method," U.S. Patent 8 996 463, Mar. 31 2015.
[45] U. Saxena and S. Sachdeva, "An insightful view on security and performance of NoSQL databases," in *Proc. Int. Conf. Recent Develop. Sci., Eng. Technol.* Singapore: Springer, 2017, pp. 643–653.
[46] M. K. Srinivasan and P. Revathy, "State-of-the-art big data security taxonomies," in *Proc. 11th Innov. Softw. Eng. Conf.*, 2018, p. 16.
[47] K. Ahmad, M. S. Alam, and N. I. Udzir, "Security of NoSQL database against intruders," *Recent Patents Eng.*, vol. 13, no. 1, pp. 5–12, 2019.
[48] W. Zugaj and A. S. Beichler, "Analysis of standard security features for selected NoSQL systems," *Amer. J. Inf. Sci. Technol.*, vol. 3, no. 2, pp. 41–49, 2019.
[49] R. Datta, N. Marchang, S. Das, K. Kant, and N. Zhang, "Security for mobile ad hoc networks," *Handbook on Securing Cyber-Physical Critical Infrastructure*. Burlington, MA, USA: Morgan Kaufmann, 2012, pp. 147–190.
[50] F. Al-Turjman, "Intelligence and security in big 5G-oriented IoNT: An overview," *Future Gener. Comput. Syst.*, vol. 102, pp. 357–368, Jan. 2020.
[51] J. Moreno, E. B. Fernandez, E. Fernandez-Medina, and M. A. Serrano, "A security pattern for key-value NoSQL database authorization," in *Proc. 23rd Eur. Conf. Pattern Lang. Programs*, Jul. 2018, p. 12.
[52] R. Rivest, *The MD5 Message-Digest Algorithm*, document RFC 1321, Internet Activities Board, 1992.
[53] H. Kaur and K. J. Kaur, "A review: Study of document oriented databases and their security," *Int. J. Adv. Res. Comput. Sci.*, vol. 4, no. 8, pp. 227–228, 2013.
[54] V. Chang, Y.-H. Kuo, and M. Ramachandran, "Cloud computing adoption framework: A security framework for business clouds," *Future Gener. Comput. Syst.*, vol. 57, pp. 24–41, Apr. 2016.
[55] K. Grolinger, W. A. Higashino, A. Tiwari, and M. A. Capretz, "Data management in cloud environments: NoSQL and NewSQL data stores," *J. Cloud Computing, Adv., Syst. Appl.*, vol. 2, no. 1, p. 22, 2013.
[56] G. Smyth, "Using data virtualisation to detect an insider breach," *Comput. Fraud Secur.*, vol. 2017, no. 8, pp. 5–7, Aug. 2017.
[57] S. Gupta, N. K. Singh, and D. S. Tomar, "Analysis of NoSQL database vulnerabilities," in *Proc. 3rd Int. Conf. Internet Things Connected Technol. (ICIoTCT)*, 2018, pp. 26–27.
[58] P. Colombo and E. Ferrari, "Access control technologies for big data management systems: Literature review and future trends," *Cybersecurity*, vol. 2, no. 1, pp. 1–13, Dec. 2019.
[59] G. S. Kapadia, "Comparative study of role based access control in cloud databases and NoSQL databases," *Int. J. Adv. Res. Comput. Sci.*, vol. 8, no. 5, pp. 51–57, 2017.
[60] H. A. Kholidy and F. Baiardi, "CIDS: A framework for intrusion detection in cloud systems," in *Proc. 9th Int. Conf. Inf. Technol.-New Generat.*, Apr. 2012, pp. 379–385.
[61] AppPerfect. (2021). *Agentless Monitor*. [Online]. Available: http://www.appperfect.com/products/agentless-monitor.php#docs
[62] A. A. Imam, S. Basri, R. Ahmad, J. Watada, and M. T. González-Aparicio, "Automatic schema suggestion model for NoSQL document-stores databases," *J. Big Data*, vol. 5, no. 1, p. 46, Dec. 2018.
[63] H. Ulusoy, P. Colombo, E. Ferrari, M. Kantarcioglu, and E. Pattuk, "GuardMR: Fine-grained security policy enforcement for MapReduce systems," in *Proc. 10th ACM Symp. Inf., Comput. Commun. Secur.*, 2015, pp. 285–296.
[64] J. Dyer, "Secure computation in the cloud using MapReduce," Ph.D. dissertation, Dept. Comput. Sci., Univ. Manchester, Manchester, U.K., 2018.
[65] E. Yoon and A. Squicciarini, "Toward detecting compromised MapReduce workers through log analysis," in *Proc. 14th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput.*, May 2014, pp. 41–50.

[66] W. Wei, J. Du, T. Yu, and X. Gu, "SecureMR: A service integrity assurance framework for MapReduce," in *Proc. Annu. Comput. Secur. Appl. Conf.*, Dec. 2009, pp. 73–82.
[67] A. Kelarev, X. Yi, S. Badsha, X. Yang, L. Rylands, and J. Seberry, "A multistage protocol for aggregated queries in distributed cloud databases with privacy protection," *Future Gener. Comput. Syst.*, vol. 90, pp. 368–380, Jan. 2019.
[68] Y. Shalabi and E. Gudes, "Cryptographically enforced role-based access control for NoSQL distributed databases," in *Proc. IFIP Annu. Conf. Data Appl. Secur. Privacy*. Cham, Switzerland: Springer, 2017, pp. 3–19.
[69] O. Ohrimenko, M. Costa, C. Fournet, C. Gkantsidis, M. Kohlweiss, and D. Sharma, "Observing and preventing leakage in MapReduce," in *Proc. 22nd ACM SIGSAC Conf. Comput. Commun. Secur.*, Oct. 2015, pp. 1570–1581.
[70] R. Kannan and D. R. Mala, "Analysis of encryption techniques to enhance secure data transmission," *Int. J. Eng. Comput. Sci.*, vol. 7, no. 9, pp. 24311–24318, Sep. 2018.
[71] V. N. Gudivada, S. Jothilakshmi, and D. Rao, "Data management issues in big data applications," *ALLDATA*, vol. 15, pp. 16–21, Apr. 2015.
[72] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya, "Ensuring security and privacy preservation for cloud data services," *ACM Comput. Surveys*, vol. 49, no. 1, pp. 1–39, Jul. 2016.
[73] G. B. Tarekegn and Y. Y. Munaye, "Big data: Security issues, challenges and future scope," *Int. J. Comput. Eng. Technol.*, vol. 7, no. 4, pp. 12–24, 2016.
[74] E. R. Osawaru and R. A. A. Habeeb, "A highlight of security challenges in big data," *Int. J. Inf. Syst. Eng.*, vol. 2, no. 1, pp. 2265–2289, 2014.
[75] Z. Yang, M. Hoseinzadeh, A. Andrews, C. Mayers, D. T. Evans, R. T. Bolt, J. Bhimani, N. Mi, and S. Swanson, "AutoTiering: Automatic data placement manager in multi-tier all-flash datacenter," in *Proc. IEEE 36th Int. Perform. Comput. Commun. Conf. (IPCCC)*, Dec. 2017, pp. 1–8.
[76] J. Sen, "Security and privacy issues in cloud computing," in *Cloud Technology: Concepts, Methodologies, Tools, and Applications*. Harrisburg, PA, USA: IGI Global, 2015, pp. 1585–1630.
[77] M. Strohbach, J. Daubert, H. Ravkin, and M. Lischka, "Big data storage," in *New Horizons for a Data-Driven Economy*. Cham, Switzerland: Springer, 2016, pp. 119–141.
[78] M. Ramachandran and V. Chang, "Towards performance evaluation of cloud service providers for cloud data security," *Int. J. Inf. Manage.*, vol. 36, no. 4, pp. 618–625, Aug. 2016.
[79] C. Newman, A. Menon-Sen, A. Melnikov, and N. Williams, *Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms*, document RFC 5802, Internet Requests for Comments, 2010.
[80] B. Kaliski, "*Pkcs# 5: Password-Based Cryptography Specification Version 2.0*," document RFC 2898, Sep. 2000.
[81] V. Koutsonikola and A. Vakali, "LDAP: Framework, practices, and trends," *IEEE Internet Comput.*, vol. 8, no. 5, pp. 66–72, Sep. 2004.
[82] S. P. Miller, B. C. Neuman, J. I. Schiller, and J. H. Saltzer, "Kerberos authentication and authorization system," Project Athena Technical Plan, Massachusetts Inst. Technol., Cambridge, MA, USA, Tech. Rep. Section E.2.1, 1988.
[83] S. Gueron, S. Johnson, and J. Walker, "SHA-512/256," in *Proc. 8th Int. Conf. Inf. Technol., New Generat.*, Apr. 2011, pp. 354–358.

KANIKA GOEL received the degree in computer science engineering in 2009, the master's degree in information technology in 2012, and the Ph.D. degree in information systems from the Queensland University of Technology, Brisbane, Australia, in 2018. She was with Tata Consultancy Services for a year. She is currently an Associate Lecturer with the School of Information Systems, Queensland University of Technology. She is teaching business process automation, modern data management, and lean six sigma. She is interested in exploring the potential of data to transform business processes. Her research interests include data analytics, modern data management, process mining, data quality, process management.

ARTHUR H. M. TER HOFSTEDE received the Ph.D. degree from Radboud Universiteit Nijmegen, Nijmegen, The Netherlands, in 1993. He has been with the Queensland University of Technology (QUT), Brisbane, Australia, since 1997, where he is currently a Professor and the Head of the School of Information Systems. From 2010 to 2018, he was a part-time Professor with the Information Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands. From 2010 to 2011, he was a Senior Visiting Scholar with Tsinghua University, Beijing, China. In 2010, he was a Visiting Professor with the Sapienza University of Rome. He has managed the well-known YAWL initiative at QUT. He was involved in the well-known workflow patterns initiative. He has coauthored over 200 publications, including over 90 journal publications. His research interests include in the area of business process management, with emphasis on business process automation and process mining.