Searchable Security Scheme for Cloud NoSQL

Mohammad Ahmadian
ahmadian@knights.ucf.edu

Advisor: Professor Dan C. Marinescu
University of Central Florida

September 16, 2017
Research goal is to find an answer to:

Is it possible to delegate processing of a private data to third-party without getting revealed?
Outline I

1 Introduction And Motivation
   - Cloud Relational Database
   - Cloud Data Storage And Management Components
   - Cloud NoSQL
   - Data Models For NoSQL
   - Cryptosystems For Outsourced Data Store

2 RELATED WORK

3 RESEARCH OBJECTIVES AND APPROACH
   - Research Objectives
   - Threat Model
   - JSON And BSON

4 CURRENT WORK AND PRELIMINARY RESULTS
   - SecureNoSQL

5 Research Plan
   - Work In Progress And Tasks Time Table
Introduction And Motivation

Database as a Service (DBaaS)

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- Cloud Relational Database
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components

- Cloud NoSQL Databases
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases

- Crypto-systems For Outsourced Data Store
- Leakage Proof Data Processing In Public Cloud
Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- Cloud Relational Database
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components

- Cloud NoSQL Databases
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases

- Crypto-systems For Outsourced Data Store
- Leakage Proof Data Processing In Public Cloud
Database as a Service (DBaaS)

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- Cloud Relational Database
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components

- Cloud NoSQL Databases
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases

- Crypto-systems For Outsourced Data Store
- Leakage Proof Data Processing In Public Cloud
Database as a Service (DBaaS)

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- **Cloud Relational Database**
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components

- **Cloud NoSQL Databases**
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases

- **Crypto-systems For Outsourced Data Store**
- **Leakage Proof Data Processing In Public Cloud**
Introduction And Motivation

Database as a Service (DBaaS)

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- Cloud Relational Database
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components

- Cloud NoSQL Databases
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases

- Crypto-systems For Outsourced Data Store
- Leakage Proof Data Processing In Public Cloud
Database as a Service (DBaaS)

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- Cloud Relational Database
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components
- Cloud NoSQL Databases
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases
- Crypto-systems For Outsourced Data Store
- Leakage Proof Data Processing In Public Cloud
Database as a Service (DBaaS)

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- Cloud Relational Database
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components
- Cloud NoSQL Databases
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases
- Crypto-systems For Outsourced Data Store
  - Leakage Proof Data Processing In Public Cloud
Database as a Service (DBaaS)

Database as a Service (DBaaS) is a cloud-based approach to the storage and management of structured data. DBaaS delivers database functionality similar to what is found in on-premise database management systems such as relational and non-relational database systems.

- **Cloud Relational Database**
  - Searchable Security Scheme for RDBMS Databases
  - Cloud Data Storage And Management Components

- **Cloud NoSQL Databases**
  - Data Models For NoSQL Databases
  - Searchable Security Scheme For NoSQL Databases

- Crypto-systems For Outsourced Data Store
- Leakage Proof Data Processing In Public Cloud
Cloud storage is cost-effective, but it poses significant security and privacy risks.

The owner of the data has no longer control on where it is stored and how it is protected against unauthorized access.

For instance, AWS offers an array of flexible and affordable data management services including Simple Storage Service (S3), SimpleDB, RDS\textsuperscript{1}, Elastic Compute Cloud (EC2) and DynamoDB.

\textsuperscript{1}Amazon Relational Database Service
Introduction - Cloud Data Storage And Management Components

- **Amazon Simple Storage Service (AWS S3)**
  AWS S3 uses a simple data model:
  - Objects: like files, contain data and metadata but, objects are not organized in a hierarchy and every object exists at the same level.
  - Buckets: a logical unit of storage used to store objects
  - Only authenticated user have access to Amazon S3.
  - Access control does not provide protection for S3 data against malicious insider. Encryption can be applied for the stored data to protect from the cloud internal.

- **Amazon Elastic Compute Cloud (EC2)**
  EC2 uses the public key part of the key pair associated with the AWS account to secure login, so that only someone with the corresponding private key can access to the EC2 instance. In addition, by using concept of security group that are basically collections of rules the traffic of EC2 instance is manageable.
Introduction- Cloud Data Storage And Management Components

- Amazon Simple Storage Service (AWS S3)
  AWS S3 uses a simple data model:
  - Objects: like files, contain data and metadata but, objects are not organized in a hierarchy and every object exists at the same level.
  - Buckets: a logical unit of storage used to store objects
  - Only authenticated user have access to Amazon S3.
  - Access control does not provide protection for S3 data against malicious insider. Encryption can be applied for the stored data to protect from the cloud internal.

- Amazon Elastic Compute Cloud (EC2)
  EC2 uses the public key part of the key pair associated with the AWS account to secure login, so that only someone with the corresponding private key can access to the EC2 instance. In addition, by using concept of security group that are basically collections of rules the traffic of EC2 instance is manageable.
Introduction - Cloud Data Storage And Management

Components

- **Amazon Simple Storage Service (AWS S3)**
  - AWS S3 uses a simple data model:
    - **Objects**: like files, contain data and metadata but, objects are not organized in a hierarchy and every object exists at the same level.
    - **Buckets**: a logical unit of storage used to store objects
  - Only authenticated user have access to Amazon S3.
  - Access control does not provide protection for S3 data against malicious insider. Encryption can be applied for the stored data to protect from the cloud internal.

- **Amazon Elastic Compute Cloud (EC2)**
  - EC2 uses the public key part of the key pair associated with the AWS account to secure login, so that only someone with the corresponding private key can access to the EC2 instance. In addition, by using concept of security group that are basically collections of rules the traffic of EC2 instance is manageable.
Introduction - Cloud Data Storage And Management Components

- **Amazon Simple Storage Service (AWS S3)**
  AWS S3 uses a simple data model:
  - Objects: like files, contain data and metadata but, objects are not organized in a hierarchy and every object exists at the same level.
  - Buckets: a logical unit of storage used to store objects
  - Only authenticated user have access to Amazon S3.
  - Access control does not provide protection for S3 data against malicious insider. Encryption can be applied for the stored data to protect from the cloud internal.

- **Amazon Elastic Compute Cloud (EC2)**
  EC2 uses the public key part of the key pair associated with the AWS account to secure login, so that only someone with the corresponding private key can access to the EC2 instance. In addition, by using concept of security group that are basically collections of rules the traffic of EC2 instance is manageable.
Introduction - Cloud Data Storage And Management Components

- Amazon Simple Storage Service (AWS S3)
  AWS S3 uses a simple data model:
  - **Objects**: like files, contain data and metadata but, objects are not organized in a hierarchy and every object exists at the same level.
  - **Buckets**: a logical unit of storage used to store objects
  - Only authenticated user have access to Amazon S3.
  - Access control does not provide protection for S3 data against malicious insider. Encryption can be applied for the stored data to protect from the cloud internal.

- Amazon Elastic Compute Cloud (EC2)
  EC2 uses the public key part of the key pair associated with the AWS account to secure login, so that only someone with the corresponding private key can access to the EC2 instance. In addition, by using concept of security group that are basically collections of rules the traffic of EC2 instance is manageable.
Introduction - Cloud Data Storage And Management Components

- Amazon Simple Storage Service (AWS S3)
  AWS S3 uses a simple data model:
  - Objects: like files, contain data and metadata but, objects are not organized in a hierarchy and every object exists at the same level.
  - Buckets: a logical unit of storage used to store objects
  - Only authenticated user have access to Amazon S3.
  - Access control does not provide protection for S3 data against malicious insider. Encryption can be applied for the stored data to protect from the cloud internal.

- Amazon Elastic Compute Cloud (EC2)
  EC2 uses the public key part of the key pair associated with the AWS account to secure login, so that only someone with the corresponding private key can access to the EC2 instance. In addition, by using concept of security group that are basically collections of rules the traffic of EC2 instance is manageable.
Cloud NoSQL:
Cloud NoSQL is a fast and flexible database service for all applications that need consistent, single-digit millisecond latency at any scale. It is a fully managed cloud database and supports both document and key-value store models. Its flexible data model and reliable performance make it a great fit for mobile, web, gaming, ad tech, IoT, and many other applications.
Data Models For NoSQL Databases:

1. **Key-value stores:** A dictionary DS where a key uniquely identifies the value.

   ![Key-value diagram](image1)

2. **Column-family stores:** Data are stored in rows and each row has a unique key and set of columns.

   ![Column-family diagram](image2)
Data Models For NoSQL Databases:

1. **Key-value stores**: A dictionary DS where a key uniquely identifies the value.

   ![Key-value stores diagram]

   - Key 1 → Value 1
   - Key 2 → Value 2

2. **Column-family stores**: Data are stored in rows and each row has a unique key and set of columns.

   ![Column-family stores diagram]

   - ID = 1: Name John, Age 27, State California
   - ID = 2: Name Daniel, Age 32, State Montana
   - ID = 3: Name Mary, Age 31, State Washington
3 **Document stores:** Data are stored in internal structure (Document) to offer higher level of granularity. Each document has a unique key to identify.

![Document Store Example]

4 **Graph Databases:** This model is based on graph and can be used to represent complex structures and highly connected data.

![Graph Database Example]
Document stores: Data are stored in internal structure (Document) to offer higher level of granularity. Each document has a unique key to identify.

Graph Databases: This model is based on graph and can used to represent complex structures and highly connected data.
Data in the cloud can be in one of three states:

1. **Store**: Encryption of data before uploading to the Cloud.
2. **Transit**: Communication channels can be secured by using the standard HTTP over Secure Socket Layer (SSL). In addition, the endpoint authentication feature of the SSL protocol makes it possible to ensure clients are communicating with an authentic cloud server.
3. **Process**: Data owner should disclose decryption key to the server in order to decrypt the data before performing any required operation. The problem is when the decryption key is compromised, the data confidentiality would be affected. Therefore, in the cloud computing model, new set of cryptosystems is required.
Data in the cloud can be in one of three states:

1. **Store**: Encryption of data before uploading to the Cloud.
2. **Transit**: Communication channels can be secured by using the standard HTTP over Secure Socket Layer (SSL). In addition, the endpoint authentication feature of the SSL protocol makes it possible to ensure clients are communicating with an authentic cloud server.
3. **Process**: Data owner should disclose decryption key to the server in order to decrypt the data before performing any required operation. The problem is when the decryption key is compromised, the data confidentiality would be affected. Therefore, in the cloud computing model, new set of cryptosystems is required.
Data in the cloud can be in one of three states:

1. **Store**: Encryption of data before uploading to the Cloud.

2. **Transit**: Communication channels can be secured by using the standard HTTP over Secure Socket Layer (SSL). In addition, the endpoint authentication feature of the SSL protocol makes it possible to ensure clients are communicating with an authentic cloud server.

3. **Process**: Data owner should disclose decryption key to the server in order to decrypt the data before performing any required operation. The problem is when the decryption key is compromised, the data confidentiality would be affected. Therefore, in the cloud computing model, new set of cryptosystems is required.
DET scheme always produces the same ciphertext for an identical pair of given plaintext and key.\(^2\) DET leaks information about ciphertext of same plaintext. DET enables server to process pipeline aggregation stages such as group, count, retrieving distinct values and equality match\(^3\) on the fields within an embedded document. The embedded document can maintain the link with the primary document through application of DET encryption. See Equation 1.

**Deterministic Encryption**

\[
\text{for } j = 1 \ldots n; \quad C_j = E_k(P_j); \quad P_j = D_k(C_j)
\]  

\(^2\)Block ciphers in Electronic Code Book (ECB) mode with a constant IV are DET.

\(^3\)Equality matches over common fields in an embedded document will select documents in the collection containing fields with specified values.
RND scheme (probabilistic) encryption, the same message with the same key yields different ciphertext. This randomness provides the highest level of security and different encryption algorithms provide RND property. RND type schemes are semantically secure against chosen plaintext attacks and hides all kind of information about ciphertext. RND scheme does not allow any efficient computation on the ciphertext.

Random Encryption

\[
\begin{align*}
C_1 &= E_k(P_1 \oplus IV), & P_1 &= IV \oplus D_k(C_1) \\
\text{for } j = 2 \ldots n; \quad C_j &= E_k(P_j \oplus C_{j-1}), & P_j &= C_{j-1} \oplus D_k(C_j) 
\end{align*}
\] (2)

4 AES in Cipher Block Chaining (CBC) mode is used for RND. AES with a key size of 128, 192 or 256 bits and with a block size of 128 bits.

5 Where: \( E_k \) is the Enc., \( D_k \) is the Dec., \( k \) is secret key \( P \) is plaintext and \( C \) is ciphertext.
Order-Preserving Encryption (OPE) projects the order relation between plaintext data elements to their ciphertext values. OPE leaks the order of ciphertext, so it supports a lower degree of security.

Order-Preserving Encryption

\[ \forall x, y \mid x, y \in Data\ Domain \]
\[ x < y \implies OPE_k(x) < OPE_k(y) \]

An efficient inequality comparisons on the encrypted data elements can be performed by applying OPE which supports range queries, comparison, Min(), Max() on the ciphertext.
AHOM allows the server to conduct computations on ciphertext with the final result that get decrypted at the proxy. In spite of sustained research efforts of the Fully Homomorphic Encryption (FHE), there is no efficient FHE, except for limited operations. We applied Paillier [1] scheme that supports additive operations. It should be noted that $m_1, m_2$ are messages to be encrypted where $m_1, m_2 \in \mathbb{Z}_n$. $r_1, r_2 \in \mathbb{Z}_n^*$ are randomly selected.

\begin{equation}
D_k\left( E_k(m_1, r_1) \times E_k(m_2, r_2) \mod n^2 \right) = m_1 + m_2 \mod n \tag{4}
\end{equation}

In other words, the product of two ciphertexts decrypt to the sum of their corresponding plaintexts.
The first SQL-aware query processing over encrypted database was *CryptDB* [2]. CryptDB satisfies data confidentiality for the relational database. However, CryptDB cannot perform queries over data encrypted with different keys. Other problem that CryptDB has is information leakage from encrypted data.

A practical searchable security scheme known as Oblivious Cross Tags (OXT) is introduced by Cash et al. [3] which can search on encrypted data sets in sub-linear time complexity by using different types of indices, however it is not practical on NoSQL data sets which are designed to scale to millions of users doing updates simultaneously.

Extended OXT introduced by Faber et al. adds a set of new features such as multi-keyword, substring, wild-cards and substring searching to the basic OXT approach. The main downsides of this

SecureNoSQL is a system which acts as a proxy to secure the communication between the NoSQL database server, and the applications server. Advantages: Using original expressive query language. Benefits from secondary indexes of database system.
The first SQL-aware query processing over encrypted database was CryptDB [2]. CryptDB satisfies data confidentiality for the relational database. However, CryptDB cannot perform queries over data encrypted with different keys. Other problem that CryptDB has is information leakage from encrypted data.

A practical searchable security scheme known as Oblivious Cross Tags (OXT) is introduced by Cash et al. [3] which can search on encrypted data sets in sub-linear time complexity by using different types of indices, however it is not practical on NoSQL data sets which are designed to scale to millions of users doing updates simultaneously.

Extended OXT introduced by Faber et al. adds a set of new features such as multi-keyword, substring, wild-cards and substring searching to the basic OXT approach. The main downsides of this

SecureNoSQL is a system which acts as a proxy to secure the communication between the NoSQL database server, and the applications server. Advantages: Using original expressive query language. Benefits from secondary indexes of database system.
The first SQL-aware query processing over encrypted database was CryptDB [2]. CryptDB satisfies data confidentiality for the relational database. However, CryptDB cannot perform queries over data encrypted with different keys. Other problem that CryptDB has is information leakage from encrypted data.

A practical searchable security scheme known as Oblivious Cross Tags (OXT) is introduced by Cash et al. [3] which can search on encrypted data sets in sub-linear time complexity by using different types of indices, however it is not practical on NoSQL data sets which are designed to scale to millions of users doing updates simultaneously.

Extended OXT introduced by Faber et al. adds a set of new features such as multi-keyword, substring, wild-cards and substring searching to the basic OXT approach. The main downsides of this

SecureNoSQL is a system which acts as a proxy to secure the communication between the NoSQL database server, and the applications server. Advantages: Using original expressive query language. Benefits from secondary indexes of database system.
The first SQL-aware query processing over encrypted database was CryptDB [2]. CryptDB satisfies data confidentiality for the relational database. However, CryptDB cannot perform queries over data encrypted with different keys. Other problem that CryptDB has is information leakage from encrypted data.

A practical searchable security scheme known as Oblivious Cross Tags (OXT) is introduced by Cash et al. [3] which can search on encrypted data sets in sub-linear time complexity by using different types of indices, however it is not practical on NoSQL data sets which are designed to scale to millions of users doing updates simultaneously.

Extended OXT introduced by Faber et al. adds a set of new features such as multi-keyword, substring, wild-cards and substring searching to the basic OXT approach. The main downsides of this

SecureNoSQL is a system which acts as a proxy to secure the communication between the NoSQL database server, and the applications server. Advantages: Using original expressive query language. Benefits from secondary indexes of database system.
## Related Work - Summary

### Comparison with related work

Table 1: Information leakage management methods comparison

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
<th>Context</th>
<th>Advantage</th>
<th>Downside</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>Oblivious Cross-</td>
<td>Searchable symmetric</td>
<td>Searches for a set of keywords</td>
<td>Practical</td>
<td>(1) Multiple interactions;</td>
<td>Cash et al. [4]</td>
</tr>
<tr>
<td>Tags (OXT)</td>
<td>encryption</td>
<td></td>
<td></td>
<td>(2) Pre-processing</td>
<td></td>
</tr>
<tr>
<td>Extended-OXT</td>
<td>Searchable symmetric</td>
<td>Searches for a set of keywords</td>
<td>Extends OXT to: (1) Substring;</td>
<td>(1) Multiple interactions;</td>
<td>Faber et al. [5]</td>
</tr>
<tr>
<td></td>
<td>encryption</td>
<td></td>
<td>(2) Wildcards, Phrase &amp;</td>
<td>(2) Preprocessing</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Substring</td>
<td></td>
<td></td>
</tr>
<tr>
<td>CryptDB</td>
<td>Secure query</td>
<td>SQL database</td>
<td>Efficient</td>
<td>Leakage from encrypted data</td>
<td>Popa et al. [2]</td>
</tr>
<tr>
<td></td>
<td>processing</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SecureNoSQL</td>
<td>Leakage resilient</td>
<td>NoSQL database</td>
<td>Covers: (1) search over</td>
<td>Requires extra hardware resources</td>
<td>Current work *</td>
</tr>
<tr>
<td></td>
<td>query processing</td>
<td></td>
<td>encrypted NoSQL databases;</td>
<td>for Proxy</td>
<td></td>
</tr>
<tr>
<td></td>
<td>over encrypted</td>
<td></td>
<td>(2) Leakage prevention</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>database</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* The paper related to this work is currently under review.
Drawbacks of CryptDb

A team of Microsoft researchers led by Seny Kamara claims to have been successful at recovering a substantial amount of data from health records stored in CryptDB (PDF), a database technology that uses layers of encryption to allow users to search through encrypted data without exposing its contents.
**Research Objectives**: The goal of this research is to design security schemes that enable cloud users to securely receive the productivity and computational benefits of the cloud DBaaS without compromising security and privacy.

**Motivation**: A 70% annual growth rate in DBaaS, and considering the cloud threat model an efficient security scheme is required for high volume data stored and processed in the cloud.

**Threat Model**: A threat model describes the threats against cloud DBaaS.

**JSON And BSON**: JSON is an open standard format used to transmit data objects consisting of key-value pairs using self-describing text (BSON is binary extension).
**Research Objectives**: The goal of this research is to design security schemes that enable cloud users to securely receive the productivity and computational benefits of the cloud DBaaS without compromising security and privacy.

**Motivation**: A 70% annual growth rate in DBaaS, and considering the cloud threat model an efficient security scheme is required for high volume data stored and processed in the cloud.

**Threat Model**: A threat model describes the threats against cloud DBaaS.

**JSON And BSON**: JSON is an open standard format used to transmit data objects consisting of key-value pairs using self describing text (BSON is binary extension).
Research Objectives: The goal of this research is to design security schemes that enable cloud users to securely receive the productivity and computational benefits of the cloud DBaaS without compromising security and privacy.

Motivation: A 70% annual growth rate in DBaaS, and considering the cloud threat model an efficient security scheme is required for high volume data stored and processed in the cloud.

Threat Model: A threat model describes the threats against cloud DBaaS.

JSON And BSON: JSON is an open standard format used to transmit data objects consisting of key-value pairs using self describing text (BSON is binary extension).
**Research Objectives:** The goal of this research is to design security schemes that enable cloud users to securely receive the productivity and computational benefits of the cloud DBaaS without compromising security and privacy.

**Motivation:** A 70% annual growth rate in DBaaS, and considering the cloud threat model an efficient security scheme is required for high volume data stored and processed in the cloud.

**Threat Model:** A threat model describes the threats against cloud DBaaS.

**JSON And BSON:** JSON is an open standard format used to transmit data objects consisting of key-value pairs using self describing text (BSON is binary extension).
Threat Model: We investigate cloud threat model from the adversarial prospective which is a holistic process based on end-to-end security. The model identifies two classes of threats.

- **External attacker**: An attacker from the outside of cloud environment might obtain unauthorized access to the data.

- **Cloud malicious insiders**: Unauthorized access to data by the cloud internals.
Threat Model: We investigate cloud threat model from the adversarial prospective which is a holistic process based on end-to-end security. The model identifies two classes of threats.

- **External attacker**: An attacker from the outside of cloud environment might obtain unauthorized access to the data.
- **Cloud malicious insiders**: Unauthorized access to data by the cloud internals
RESEARCH OBJECTIVES AND APPROACH: JSON And BSON

**JSON And BSON:**
- Open standard format
- Self describing format
- BSON is a binary extension for JSON
- BSON supports more data types

In this work we use JSON to create a new concept called **security plan.**
**RESEARCH OBJECTIVES AND APPROACH-JSON And BSON**

**JSON And BSON:**
- Open standard format
- Self describing format
- BSON is a binary extension for JSON
- BSON supports more data types

In this work we use JSON to create a new concept called **security plan**.
RESEARCH OBJECTIVES AND APPROACH-JSON And BSON

JSON And BSON:
- Open standard format
- Self describing format
- BSON is a binary extension for JSON
- BSON supports more data types

In this work we use JSON to create a new concept called security plan.
RESEARCH OBJECTIVES AND APPROACH-JSON And BSON

JSON And BSON:
- Open standard format
- Self describing format
- BSON is a binary extension for JSON
- BSON supports more data types

In this work we use JSON to create a new concept called security plan.
We deign SecureNoSQL which is a system that provides practical and provable confidentiality in presence of these attacks for applications backed by NoSQL databases. The key part of SecureNoSQL is evaluation a set of operations on the encrypted databases. Moreover, the designed novel algorithms for information leakage prevention from data or query are added to SecureNoSQL. We also introduced a novel descriptive language based on the JSON notations which enables the users to generate a security plan. The security plan is useful tools for data owners for regulating security parameters management without getting involved in the details.
Figure 1: High-level architecture of SecureNoSQL as a secure proxy between users applications and cloud NoSQL database server.
Some features of SecureNoSQL

1. Descriptive language based on JSON notations to create a security plan.
2. A multi-key, multi-level mechanism.
3. The effective validation procedure against security plan in SecureNoSQL helps to avoid unnecessarily increase of workload and response time of remote cloud server.
4. Support for a comprehensive, flexible protection. The solution is open-ended, users can add new customized cryptographic modules simply by using designed descriptive language.
5. A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.
6. SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud. 

---

a. The malicious insider could pool all databases and extract sensitive information from correlation with various hosted databases.
Some features of SecureNoSQL

1. Descriptive language based on JSON notations to create a security plan.
2. A multi-key, multi-level mechanism.
3. The effective validation procedure against security plan in SecureNoSQL helps to avoid unnecessarily increase of workload and response time of remote cloud server.
4. Support for a comprehensive, flexible protection. The solution is open-ended, users can add new customized cryptographic modules simply by using designed descriptive language.
5. A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.
6. SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud. *The malicious insider could pool all databases and extract sensitive information from correlation with various hosted databases.*
Some features of SecureNoSQL

1. Descriptive language based on JSON notations to create a security plan.

2. A multi-key, multi-level mechanism.

3. The effective validation procedure against security plan in SecureNoSQL helps to avoid unnecessarily increase of workload and response time of remote cloud server.

4. Support for a comprehensive, flexible protection. The solution is open-ended, users can add new customized cryptographic modules simply by using designed descriptive language.

5. A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.

6. SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud. The malicious insider could pool all databases and extract sensitive information from correlation with various hosted databases.
Some features of SecureNoSQL

1. Descriptive language based on JSON notations to create a security plan.

2. A multi-key, multi-level mechanism.

3. The effective validation procedure against security plan in SecureNoSQL helps to avoid unnecessarily increase of workload and response time of remote cloud server.

4. Support for a comprehensive, flexible protection. The solution is open-ended, users can add new customized cryptographic

5. A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.

6. SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud. \(^a\)

\(^a\)The malicious insider could pool all databases and extract sensitive information from correlation with various hosted databases.
Some features of SecureNoSQL

1. Descriptive language based on JSON notations to create a security plan.

2. A multi-key, multi-level mechanism.

3. The effective validation procedure against security plan in SecureNoSQL helps to avoid unnecessarily increase of workload and response time of remote cloud server.

4. Support for a comprehensive, flexible protection. The solution is open-ended, users can add new customized cryptographic

5. A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.

SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud.  

---

"A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.

SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud.  

---

"A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.

SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud.  

---

"The malicious insider could pool all databases and extract sensitive information from correlation with various hosted databases."
Some features of SecureNoSQL

1. Descriptive language based on JSON notations to create a security plan.
2. A multi-key, multi-level mechanism.
3. The effective validation procedure against security plan in SecureNoSQL helps to avoid unnecessarily increase of workload and response time of remote cloud server.
4. Support for a comprehensive, flexible protection. The solution is open-ended, users can add new customized cryptographic
5. A balanced system with a security level-proportional overhead. The overhead of scheme is proportional to the desired level of security.
6. SecureNoSQL addresses the information leakage from fully or partially encrypted databases in the cloud.  

a. The malicious insider could pool all databases and extract sensitive information from correlation with various hosted databases.
Super Document:

\[ d_1 = \langle \langle k_1, v_1 \rangle, \langle k_2, v_2 \rangle, \ldots, \langle k_i, v_i \rangle \rangle \]
\[ d_2 = \langle \langle k_1, v_1 \rangle, \langle k_2, v_2 \rangle, \ldots, \langle k_j, v_j \rangle \rangle \]

\[ \ldots \]
\[ d_n = \langle \langle k_1, v_1 \rangle, \langle k_2, v_2 \rangle, \ldots, \langle k_l, v_l \rangle \rangle \]  \hspace{1cm} (5)

Super Document \( \mathcal{D} = \bigcup_{i=1}^{n} d_i \)

**Match function** : \( \mathcal{M}(d_i, d_j) \) determines whether any two given documents \( d_i, d_j \) can be merged or not.\(^6\)

---

\(^6\)Two documents can be merged provided that they share the same attribute from an *identifying* class or group of attributes from *semi-identity* class.
**Security plan** is a document that contains a hierarchical collection of key-value pairs that describes data elements, parameters of cryptosystems and mapping between these two. Every security plan document includes four top-level sections represented in key-value pairs.

![Diagram of Security Plan]

**Figure 2:** The high level structure of the security plan.
Figure 3: Collection (metadata) encryption: (a) The chart outlines the structure of collection containing the name of collection and name of all fields which are considered as meta-data thus should be protected with proper cryptographic module. The pointer to a cryptomodule, the encryption key, and the initialization vector used for the encryption of the items. (b) The description of a collection and security parameters in designed JSON based language.
Cryptographic modules introduces all cryptosystems and their parameters such as key, key-size, initialization vector and output-size.
Figure 5: *Data elements* containing attributes of data elements such as name, type and value for each collection and name. Then introduces security parameters for each data element. (b) The data element section of a sample database which are represented in designed notation. A data item has 7 fields: id, name, salary, balance, ccn, ssn, and email. The id, name, email and salary are required fields.
Figure 6: Structure and description of Mapping cryptographic modules to the Data element: (a) Security plan with the fourth section expanded. This section establishes a correspondence between the data fields and the cryptographic modules used to encrypt and decrypt it. (b) The mapping section of the schema for a sample database with 7 fields. For example, the id and the name will be encrypted with OPE 128 bit and AES-DET, respectively.
Figure 7: *SecureNoSQL* applied to: (a) The key-value data model; $Key_1, \ldots, Key_n$ are all encrypted using the cryptographic module $z$ while the corresponding values, $Value_1, \ldots, Value_n$ are encrypted with cryptographic modules $1, 2, \ldots, n$, respectively. (b) The document store data model; the meta-data such as collection name encrypted as well as attributes with assigned cryptographic modules.
**Figure 8:** The validation process of input data against security plan in the client side.
Figure 9: Security plan designed for sample input: (a) Data element section of sample security plan. (b) Output of JSON Data validation for sample database.
Figure 10: The query `db.customers.find({salary:{$gt:5000}, balance:{$lt:2000}})` received from an application. (a) The parsing tree of the query (b) The cryptographic modules applied to the data elements according to schema definition.
Table 2: Sample queries and their corresponding encrypted version

<table>
<thead>
<tr>
<th>Query</th>
<th>Encrypted query</th>
</tr>
</thead>
</table>
| 1     | db.customers.find({ssn:936136916})
|       | db["k/levnbanDMQHNkb9cRgUg=="].find({"5pgAxn6BF08WtM7zyuYaKg==":74172405478441908041711118833862143778}) |
| 2     | db.customers.find({balance:{gte:5084610},balance:{lte:9911843}})
|       | db["k/levnbanDMQHNkb9cRgUg=="].find({"3iXpo2l8xZpW7J7TezFdeA==":{gte:402982988013604629517872370128473753},"3iXpo218xZpW7J7TezFdeA==":{lte:785596355698717592780268633369454231}}) |
| 3     | db.customers.aggregate([{$group:{_id:null,minBalance:{min:"$balance"}}}] )
|       | db["k/levnbanDMQHNkb9cRgUg=="].aggregate([{$group:{_id:null,EncMinBalance:{min:"$3iXpo2l8xZpW7J7TezFdeA=="}}}] ) |
| 4     | db.customers.aggregate([{$group:{_id:null,maxBalance:{max:"$balance"}}}] )
|       | db["k/levnbanDMQHNkb9cRgUg=="].aggregate([{$group:{_id:null,EncmaxBalance:{max:"$3iXpo2l8xZpW7J7TezFdeA=="}}}] ) |
| 5     | db.customers.find({$or:[{Salary:{$gt:516046}},{balance:{$lt:285462}}]})
|       | db["k/levnbanDMQHNkb9cRgUg=="].find({ $or: [ { "9mnGu8Q2VDstE+T9jFw2wQ==": 40994186216785746613193244129885849 },{"3iXpo2l8xZpW7J7TezFdeA==":{lt:22657430453144634679791167652174833}]}) |
### Table 3: Overhead of encryption upon security level

<table>
<thead>
<tr>
<th>Database</th>
<th>Plain</th>
<th>OPE64</th>
<th>OPE128</th>
<th>OPE256</th>
<th>OPE512</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size(MB)</td>
<td>170</td>
<td>430</td>
<td>508</td>
<td>662</td>
<td>1000</td>
</tr>
</tbody>
</table>

### Table 4: Overhead of RND and DET encryption

<table>
<thead>
<tr>
<th>Database</th>
<th>Plain</th>
<th>RND</th>
<th>DET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size(MB)</td>
<td>170</td>
<td>170</td>
<td>170</td>
</tr>
</tbody>
</table>

### Table 5: Overhead of AHOM encryption

<table>
<thead>
<tr>
<th>Database</th>
<th>Plain</th>
<th>AHOM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Size(MB)</td>
<td>170</td>
<td>10880</td>
</tr>
</tbody>
</table>
Figure 11: Query processing time in milliseconds (ms) for the unencrypted database and for the encrypted databases when the 32-bit keys are encrypted as 64, 128, 256 and 512-bit integers.
Response time: shortest for comparison and longest for aggregated queries.
The query processing time: for a given type of query increases, but only slightly, less than 5% when the key length increases from 64, to 128, 256, and 512 bit.
As expected, the OPE encryption time increases significantly with the size of the encryption space; it increases almost tenfold when the size of the encrypted output increases from 64-bit to 1024-bit and it is about 10 ms for 256-bit.
The decryption time is considerably smaller, it increases only slightly from 0.11 ms to 0.17 when the size of the encrypted key increases from 64-bit to 1024 bit.
Research Plan (Past-Current-Future)
Figure 12: Estimate work plan and timeline
### Table 6: List of publications

<table>
<thead>
<tr>
<th>Paper No</th>
<th>Paper Title</th>
<th>Authorship</th>
<th>Journal or Conference</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper 6</td>
<td>On information leakage in cloud database services</td>
<td>M.Ahmadian, D.Marinescu</td>
<td>Transaction of sustainable computation</td>
<td>Under review</td>
</tr>
</tbody>
</table>
The current research will be continued by the following suggestions:

- Multiple proxies in order to deal with a huge number of clients,
- Developing an efficient, fully homomorphic encryption for unlimited operations over the encrypted data,
- Encryption key management mechanism development for periodically assigning new key for cryptosystems in order to obtain higher levels of security.
Information Leakage Prevention
Information Leakage

Information leakage can be defined as using combination of data, meta-data and query that are classified at lower level $L_1$ to extract information that are at higher level $L_2$. 
Introduction - Information Leakage

Information Leakage

This work is under progress ...


The End