Background

Recently, cloud database storage has become an inexpensive and convenient option to store information; however, this relatively new area of service can be vulnerable to security breaches [1]. Storing data in a foreign location requires the owner to relinquish control of their information. This opens the possibility for internal, malicious attacks that can involve the manipulation, omission, or addition of data [2].

Our research tests a potential solution for retaining data as it was intended to be stored in these cloud databases: by converting the original databases to Integrity Coded Databases (ICDB) [3]. ICDBs utilize Integrity Codes (IC): cryptographic codes created for the data by a private key that only the data owner has access to. When the database is queried, an integrity code is returned along with the queried information. The owner is able to verify that the information is correct and fresh [3]. Consequently, ICDBs also incur performance and memory penalties. In our research, we explore, test, and benchmark ICDBs to determine the costs and benefits of maintaining an ICDB versus a standard database.

Objectives

- Implement an Integrity Coded Database (ICDB)
- Verify that the data owner is able to detect malicious changes
- Test the performance of an ICDB
- Compare the performance to a standard database

Procedure

**Implementation**

MySQL was used to set up and configure several databases [4].

**Conversion**

Java conversion modules generated all integrity codes.

**Testing**

MySQLSlap and Workbench tested a variety of queries [5].

**Analysis**

Results were analyzed and compared for relative memory and performance.

Benchmarks

![Database Sizes (MiB)](image)

Figure 1. Database size relationships between 3 databases converted with AES, Hashing, and RSA. This chart uses a logarithmic base 2 scale, measured in MiB.

<table>
<thead>
<tr>
<th>Database</th>
<th>AES</th>
<th>Hashing</th>
<th>RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>World</td>
<td>1.5s</td>
<td>9.6s</td>
<td>17.2s</td>
</tr>
<tr>
<td>Sakila</td>
<td>1.5s</td>
<td>24.8s</td>
<td>8.5s</td>
</tr>
<tr>
<td>Employees</td>
<td>1.5s</td>
<td>13.3s</td>
<td>8.6s</td>
</tr>
</tbody>
</table>

**Conversion Time (s)**

![Database Conversion Time (s)](image)

Figure 2. Database avg. conversion time relationships between 3 databases converted with AES, Hashing, and RSA. This chart uses a logarithmic base 2 scale, measured in seconds.

<table>
<thead>
<tr>
<th>Database</th>
<th>AES</th>
<th>Hashing</th>
<th>RSA</th>
</tr>
</thead>
<tbody>
<tr>
<td>World</td>
<td>1.5s</td>
<td>9.6s</td>
<td>17.2s</td>
</tr>
<tr>
<td>Sakila</td>
<td>1.5s</td>
<td>24.8s</td>
<td>8.5s</td>
</tr>
<tr>
<td>Employees</td>
<td>1.5s</td>
<td>13.3s</td>
<td>8.6s</td>
</tr>
</tbody>
</table>

**SELECT * Efficiency**

![SELECT * Efficiency](image)

Figure 3. Database avg. query efficiency is measured by dividing the ICDB execution/retrieval time by the standard database execution/retrieval time. Each data point is a multiple of the query execution on a standard database.

<table>
<thead>
<tr>
<th>emp_no</th>
<th>birth_date</th>
<th>first_name</th>
<th>last_name</th>
<th>gender</th>
</tr>
</thead>
<tbody>
<tr>
<td>10001</td>
<td>1953-09-02</td>
<td>Georgi</td>
<td>Pacello</td>
<td>M</td>
</tr>
<tr>
<td>10002</td>
<td>1954-06-02</td>
<td>Beazel</td>
<td>Simmel</td>
<td>F</td>
</tr>
<tr>
<td>10003</td>
<td>1955-12-03</td>
<td>Porto</td>
<td>Romford</td>
<td>M</td>
</tr>
<tr>
<td>10004</td>
<td>1954-05-01</td>
<td>Christian</td>
<td>Kovick</td>
<td>M</td>
</tr>
</tbody>
</table>

ICDBs can contain integrity codes along each data entry.

Results

- ICDBs are much larger than their standard database counterparts, by a factor of at least 2
- Different implementations (AES, Hashing, RSA) offer unique approaches for an ICDB solution
- AES used the least memory, while RSA used the most
- AES converted in seconds, while RSA can take hours
- AES queried the fastest, while RSA queried the slowest
- Queries can take 1.2 - 5.0 times as long to execute, depending on the complexity of the query and the size of each integrity code
- ICDBs are able to verify against data forgery, data substitution, and old data attacks

References


Acknowledgements

We would like to thank our mentors Dr. Yeh and Dr. Xu for organizing our research project. This work is supported by US National Science Foundation (NSF) under grant CNS 1461133.