DataScience@SMU

Encryption Options for MySQL and MongoDB Data at Rest

MSDS 7330 Term Project (08/20/2018)

Shanqing Gu

MSDS 7330 Term Project Outline

Introduction

What is Encryption at Rest?
Why do we need data encrypted?

MySQL

1.1 MySQL Versions
1.2 General Data Encryption Methods in MySQL
1.3 Data Encryption Methods in MySQL Workbench
1.4 Simulation 1: Turn on SSL encryption and test connection between client and server
1.5 Simulation 2: Encrypt individual fields with AES
1.6 Simulation 3: Encrypt individual fields with MD5

MongoDB

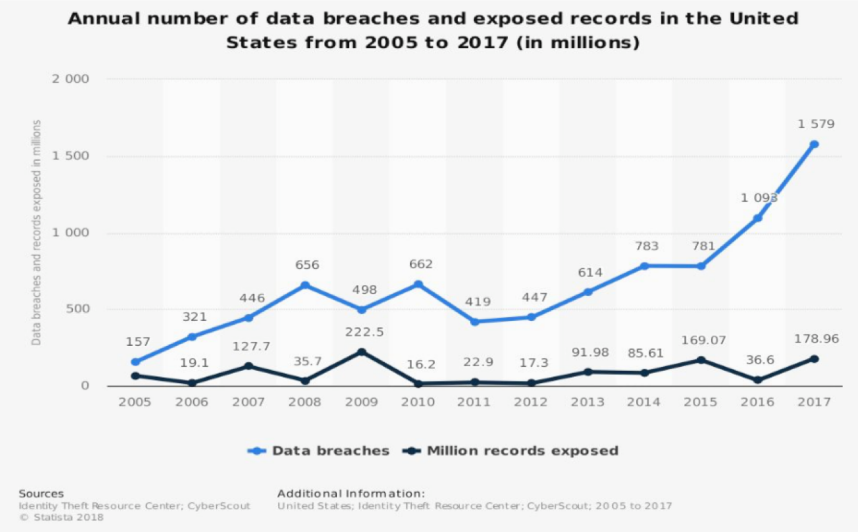
2.1 MongoDB Versions
2.2 Data Encryption Methods in MongoDB
2.3 Encryption at Rest in MongoDB Atlas
2.4 Encryption at Rest in MongoDB Community Edition
2.5 Methods for MongoDB individual field encryption
2.6 Simulation for MongoDB individual field encryption

Summary

Introduction

1. What is Encryption at Rest?

- ❖ Data at rest generally refers to the data that is stored in persistent storage (e. g. disk, tape, database, etc.)
- ❖ Encryption at rest can be achieved at 4 levels (hardware, filesystem, database, or application)
- ❖ Encryption data on the network is known as “Data-in-Motion Encryption” (e.g. TLS/SSL)



2. Why do we need data encrypted?

- ➢ Regulatory compliance for handling and storing regulated or sensitive data (e.g. HIPPA, etc.)
- ➢ Internal compliance policy to protect sensitive data (e. g. SSNs, credit card numbers etc.)
- ➢ Secure private data (Encryption is hard and often ignored?)
- ➢ Cyber defense

MySQL

1.1 MySQL Versions

Oracle MySQL Cloud Service (commercial) is built on MySQL Enterprise Edition and powered by Oracle Cloud, providing an enterprise-grade MySQL database service.

MySQL Enterprise Edition (commercial) includes the most comprehensive set of advanced features and management tools for MySQL.

MySQL Cluster CGE (commercial) is a real-time open source transactional database designed for fast, always-on access to data under high throughput conditions.

MySQL Community Edition (GPL) includes **Community Server**, Cluster, Router, Shell, **Workbench**, Connectors, on Windows (installer & Tools), Yum Repository, APT Repository, SUE Repository)

1.2 General Data Encryption Methods in MySQL

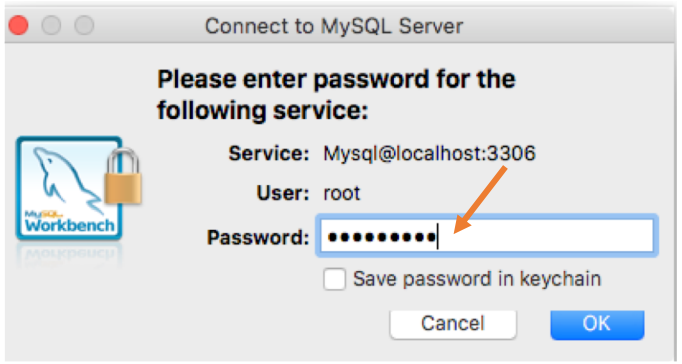
- Full-disk encryption method (weakest)
- Application level encryption method (best but not always possible to change the application code)
- Database-level (table) encryption (valuable alternative and commonly used for MySQL Community Edition

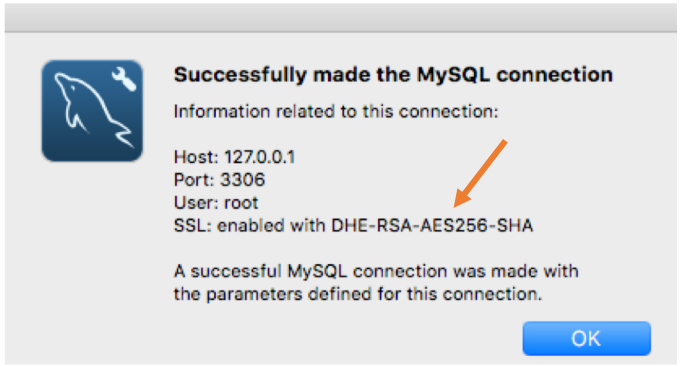
1.3 Data Encryption Methods in MySQL Workbench

- Turn on SSL encryption and test connection between client and server. This SSL encryption needs to set up the paths to Client key file, Client Certificate file, and Certificate Authority file to SSL.

- Use the previously installed EcommerceDB in MySQL Workbench, encrypt specifically the CardNumber fields with table space encryption functions (i.e. AES_DECRYPT, MD5, etc.)

1.4 Simulation 1: Turn on SSL encryption and test connection between client and server






```
mysql> SHOW GLOBAL VARIABLES LIKE 'tls_version';
+---------------+-----------------------+
| Variable_name | Value                 |
+---------------+-----------------------+
| tls_version   | TLSv1,TLSv1.1,TLSv1.2 |
+---------------+-----------------------+
1 row in set (0.00 sec)
```

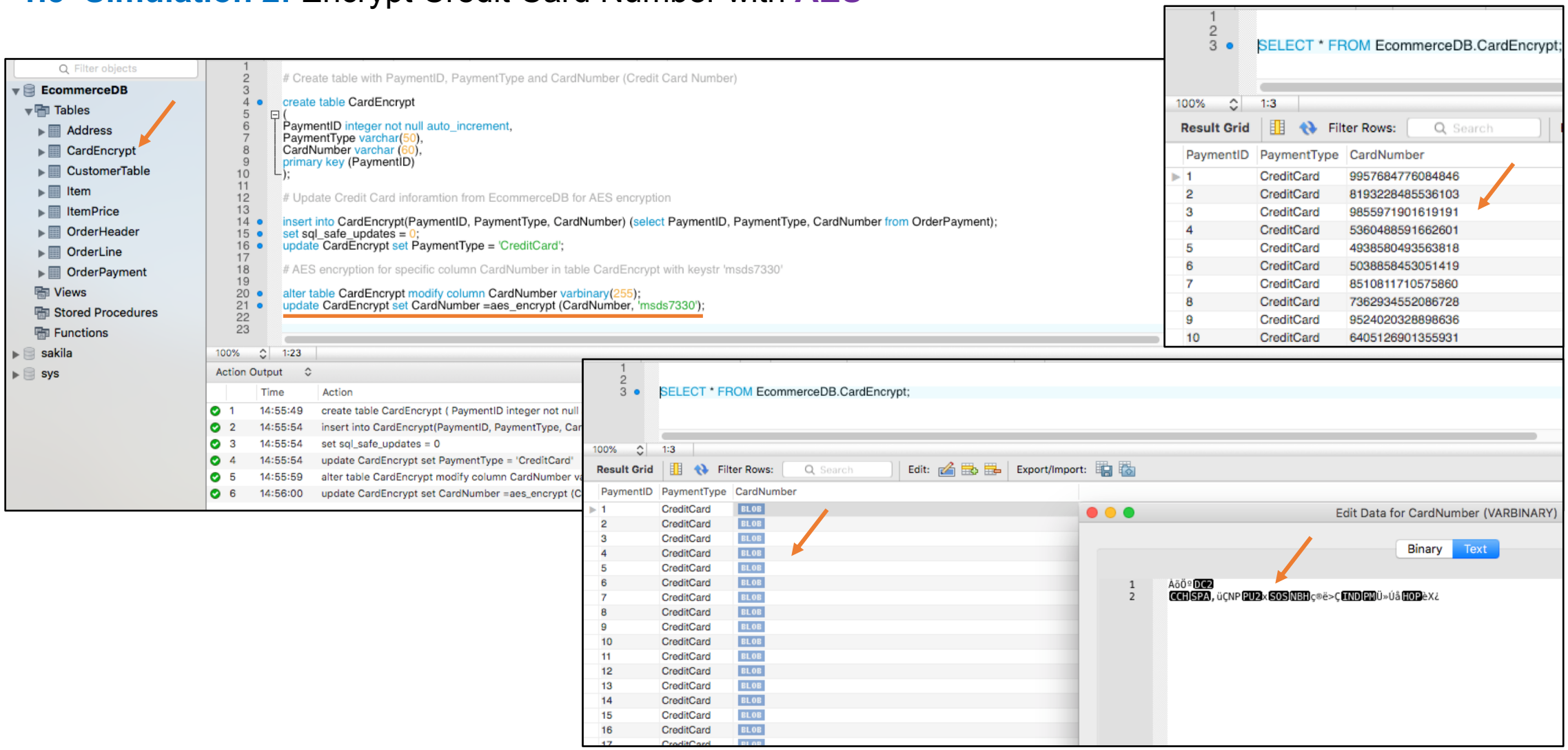
```
mysql> show global variables like 'have_%ssl';
+---------------+-------+
| Variable_name | Value |
+---------------+-------+
| have_openssl  | YES   |
| have_ssl      | YES   |
+---------------+-------+
2 rows in set (0.00 sec)

mysql> SHOW STATUS LIKE 'Ssl_server_not%';
+----------------------+------------------------+
| Variable_name        | Value                  |
+----------------------+------------------------+
| Ssl_server_not_after | Aug  4 13:56:18 2028 GMT |
| Ssl_server_not_before | Aug  7 13:56:18 2018 GMT |
+----------------------+------------------------+
2 rows in set (0.00 sec)
```

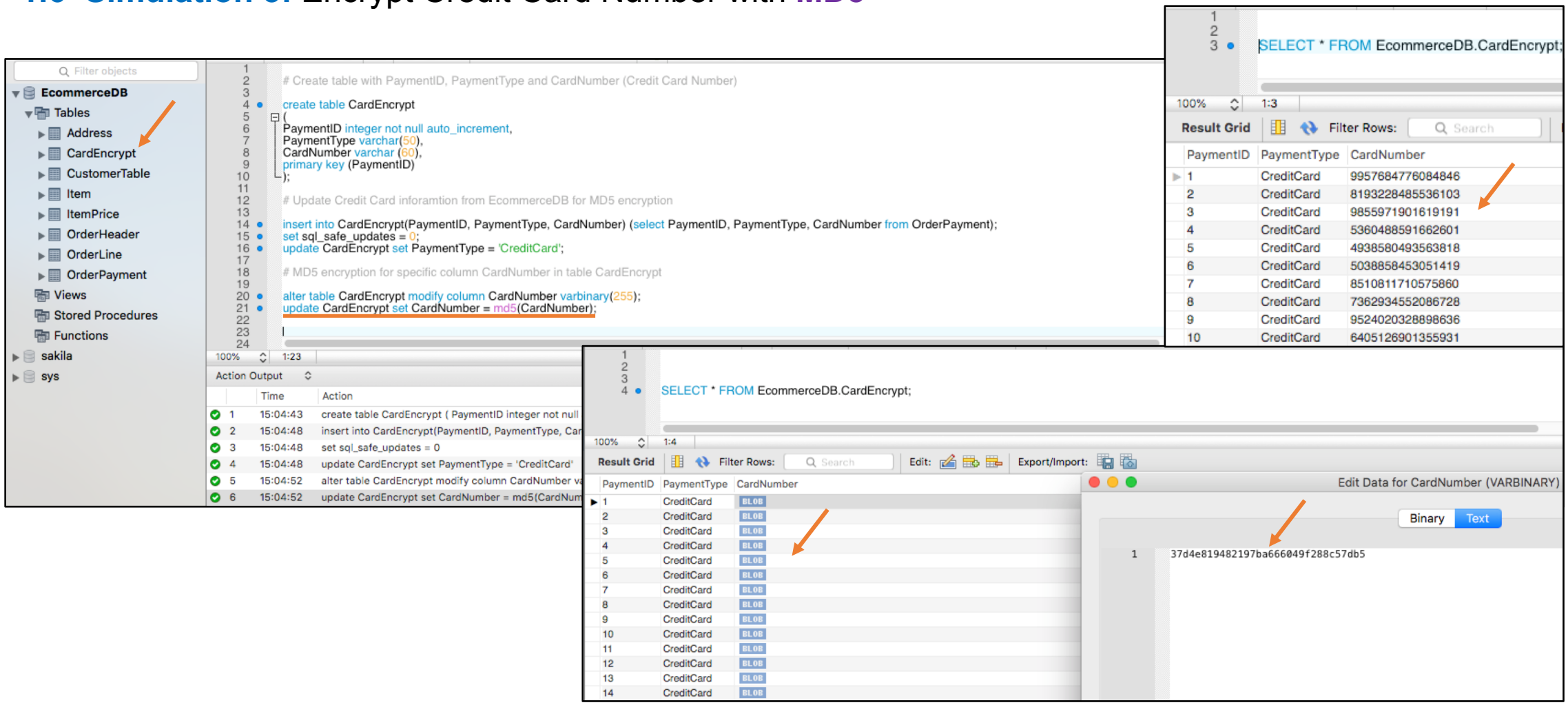
```
mysql> show global variables like '%ssl%';
+--------------------+-----------------+
| Variable_name      | Value           |
+--------------------+-----------------+
| have_openssl       | YES             |
| have_ssl           | YES             |
| mysqlx_ssl_ca      |                 |
| mysqlx_ssl_capath  |                 |
| mysqlx_ssl_cert    |                 |
| mysqlx_ssl_cipher  |                 |
| mysqlx_ssl_crl     |                 |
| mysqlx_ssl_crlpath |                 |
| mysqlx_ssl_key     |                 |
| ssl_ca             | ca.pem          |
| ssl_capath         |                 |
| ssl_cert           | server-cert.pem |
| ssl_cipher         |                 |
| ssl_crl            |                 |
| ssl_crlpath        |                 |
| ssl_fips_mode      | OFF             |
| ssl_key            | server-key.pem  |
+--------------------+-----------------+
17 rows in set (0.00 sec)
```

ssl_ca: Path to Certificate Authority file for SSL.
ssl_cert: Path to Client Certificate file for SSL.
ssl_key: Path to Client Key file for SSL

1.5 Simulation 2: Encrypt Credit Card Number with AES



1.6 Simulation 3: Encrypt Credit Card Number with MD5



MongoDB

2.1 MongoDB Versions

- MongoDB Atlas is to deploy, operate and scale a MongoDB in the cloud
- MongoDB Community Server (current stable release 4.0.1)
- MongoDB Enterprise Server includes in-memory and encrypted storage engine and advance security
- MongoDB Ops Manager is to monitor, automate, backup and query optimization in data center
- MongoDB Compass as the GUI for MongoDB allows to querying, indexing, document validation and so on
- MongoDB Connector for BI allows to use BI tool of choice to visualize, discover and report
- MongoDB Charts creates visualization of MongoDB data

2.2 Data Encryption Methods in MongoDB

- **MongoDB Enterprise Server** secures data with LDAP and Kerberos access controls.
- **MongoDB Community Edition** has two main authentication methods (Scram-SHA-1 and X.509 Certificate Authentication) and encrypt transaction between client and server using TLS/SSL.
- **MongoDB Compass** has authentication (Username/Password, Kerberos, LDAP or X.509) and SSL encryption.
- **MongoDB Atlas** provides the AWS KMS encryption key details with the WiredTiger™ Encrypted Storage Engine

Two levels of Encryption at Rest:

Storage engine encryption:

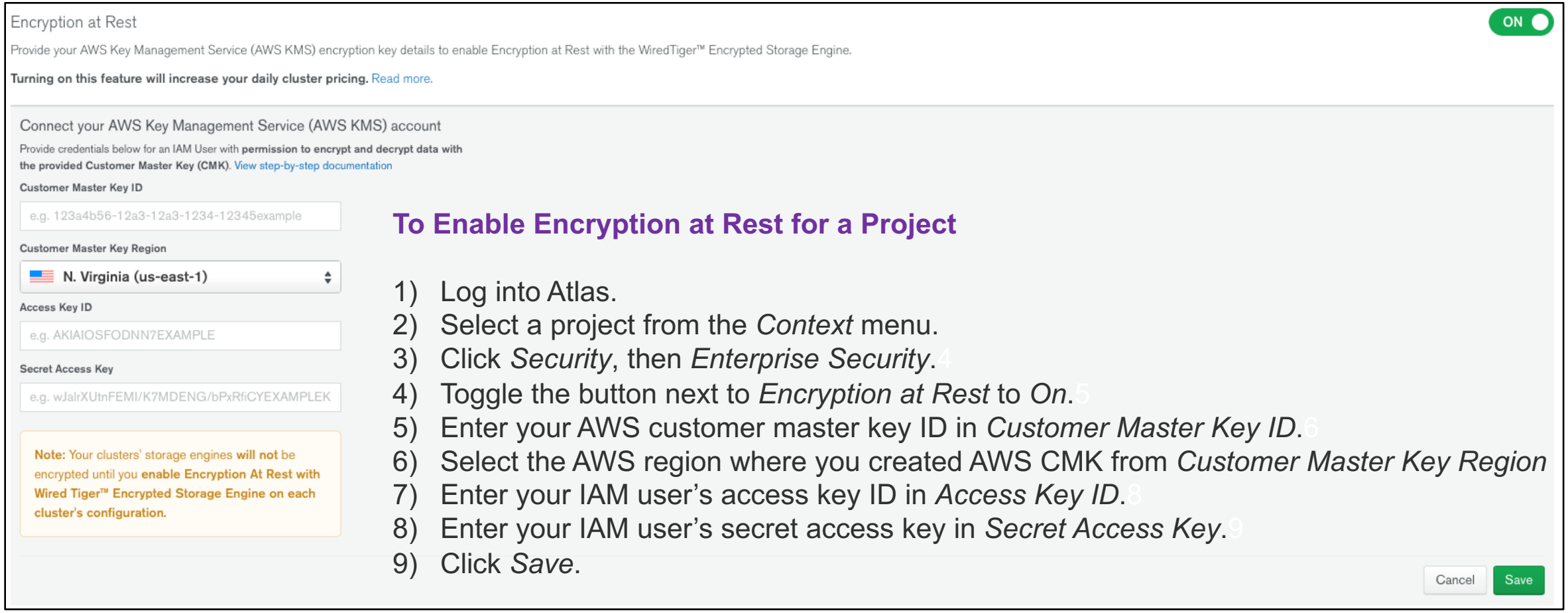
- Native encryption option for the WiredTiger storage engine only
- AES256-CBC via OpenSSL (https://www.openssl.org)
- AES-256 uses a symmetric key; i.e. the same key to encrypt and decrypt text

Application level encryption:

- Selectively encrypt only required fields in application
- Advantages: only readable by the application, low resource cost, and offload encryption overhead from database

2.3 Enable Encryption at Rest in MongoDB Atlas

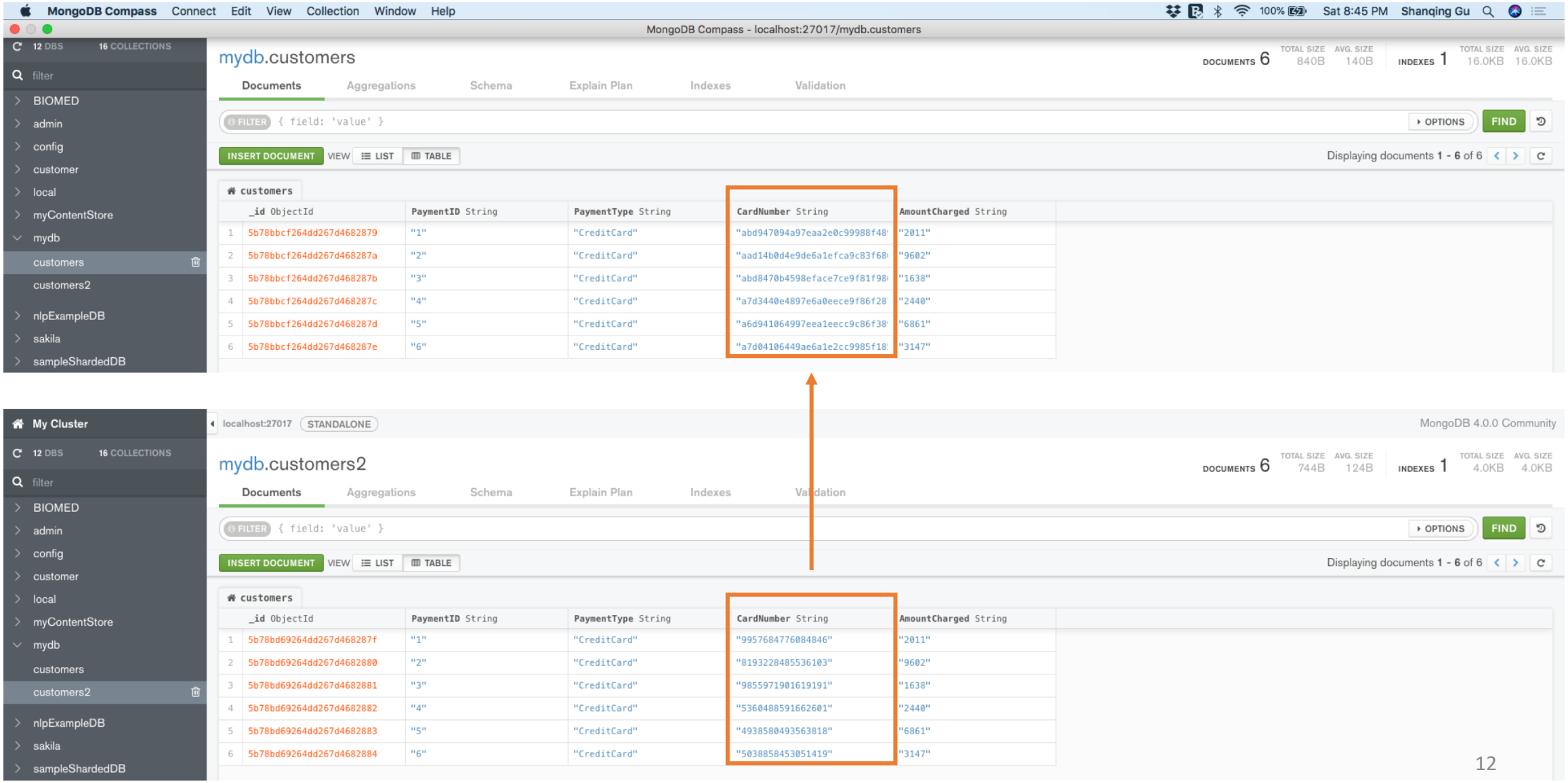
MongoDB Atlas provides the **AWS KMS** encryption key details with the **WiredTiger**™ Encrypted Storage Engine, supports encryption via customer **KMS** and **LDAP** User Authentication and Authorization in Atlas projects and clusters.



To Enable Encryption at Rest for a Project

1) Log into Atlas.
2) Select a project from the *Context* menu.
3) Click *Security*, then *Enterprise Security*.
4) Toggle the button next to *Encryption at Rest* to *On*.
5) Enter your AWS customer master key ID in *Customer Master Key ID*.
6) Select the AWS region where you created AWS CMK from *Customer Master Key Region*
7) Enter your IAM user's access key ID in *Access Key ID*.
8) Enter your IAM user's secret access key in *Secret Access Key*.
9) Click *Save*.

2.4 Encryption at Rest in MongoDB Community Edition



2.5 Methods for MongoDB individual field encryption

- eCrptFS (http://ecryptfs.org) is used by Percona to encrypt MongoDB Data at Rest (https://www.percona.com/blog/2018/04/02/mongodb-data-at-rest-encryption-using-ecryptfs/)

- Mongoose.js (https://www.npmjs.com/package/mongoose) uses AES-256-CTR to encrypt individual fields (https://www.npmjs.com/package/mongoose-field-encryption)

- Use Node.js Crypto Module and MongoDB Node.js Driver to encrypt individual fields
  - Node.js Crypto is one built-in module in Node.js (https://nodejs.org) to handle OpenSSL cryptographic functions
  - NPM (https://www.npmjs.com) is the pre-installed package manager for the Node.js server platform
  - Use MongoDB with Node.js (http://mongodb.github.io/node-mongodb-native/3.1/)

2.6 Simulation for MongoDB individual field encryption

```
SELECT * FROM EcommerceDB.CardEncrypt;
```

100% 1:3

Result Grid | Filter Rows: Search

PaymentID	PaymentType	CardNumber
1	CreditCard	9957684776084846
2	CreditCard	8193228485536103
3	CreditCard	9855971901619191
4	CreditCard	5360488591662601
5	CreditCard	4938580493563818
6	CreditCard	5038858453051419
7	CreditCard	8510811710575860
8	CreditCard	7362934552086728
9	CreditCard	9524020328898636
10	CreditCard	6405126901355931

Json Data

CreditCardInfo.json

```
[
    {
        "PaymentId" : 1,
        "PaymentType" : "CreditCard",
        "CardNumber" : "9957684776084846",
        "AmountCharged" : 2011,
        "OrderHeader_OrderNumber" : 1
    },
    {
        "PaymentId" : 2,
        "PaymentType" : "CreditCard",
        "CardNumber" : "8193228485536103",
        "AmountCharged" : 9602,
        "OrderHeader_OrderNumber" : 2
    },
    {
        "PaymentId" : 3,
        "PaymentType" : "CreditCard",
        "CardNumber" : "9855971901619191",
        "AmountCharged" : 1638,
        "OrderHeader_OrderNumber" : 3
    },
```

MongoDB Node.js Driver

```
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://localhost:27017/mydb";
undefined
>
> MongoClient.connect(url, function(err, db) {
...   if (err) throw err;
...   var myobj = [
...     {PaymentID: '1', PaymentType: 'CreditCard', CardNumber:'9957684776084846', AmountCharged: '2011'},
... {PaymentID: '2', PaymentType: 'CreditCard', CardNumber:'8193228485536103', AmountCharged: '9602'},
... {PaymentID: '3', PaymentType: 'CreditCard', CardNumber:'9855971901619191', AmountCharged: '1638'},
... {PaymentID: '4', PaymentType: 'CreditCard', CardNumber:'5360488591662601', AmountCharged: '2440'},
... {PaymentID: '5', PaymentType: 'CreditCard', CardNumber:'4938580493563818', AmountCharged: '6861'},
... {PaymentID: '6', PaymentType: 'CreditCard', CardNumber:'5038858453051419', AmountCharged: '3147'},
...
...   ];
...   db.collection("customers2").insertMany(myobj, function(err, res) {
...     if (err) throw err;
...     console.log("Number of records inserted: " + res.insertedCount);
...     db.close();
...   });
... });Table created!

undefined
> Number of records inserted: 6
```

My Cluster — localhost:27017 STANDALONE — MongoDB 4.0.0 Community

12 DBS 16 COLLECTIONS

filter

- BIOMED
- admin
- config
- customer
- local
- myContentStore
- mydb
  - customers
  - customers2
- nlpExampleDB
- sakila
- sampleShardedDB

mydb.customers2

DOCUMENTS 6 TOTAL SIZE 744B AVG. SIZE 124B | INDEXES 1 TOTAL SIZE 4.0KB AVG. SIZE 4.0KB

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

FILTER { field: 'value' } OPTIONS FIND

INSERT DOCUMENT VIEW LIST TABLE — Displaying documents 1 - 6 of 6

customers

	_id ObjectId	PaymentID String	PaymentType String	CardNumber String	AmountCharged String
1	5b78bd69264dd267d468287f	"1"	"CreditCard"	"9957684776084846"	"2011"
2	5b78bd69264dd267d4682880	"2"	"CreditCard"	"8193228485536103"	"9602"
3	5b78bd69264dd267d4682881	"3"	"CreditCard"	"9855971901619191"	"1638"
4	5b78bd69264dd267d4682882	"4"	"CreditCard"	"5360488591662601"	"2440"
5	5b78bd69264dd267d4682883	"5"	"CreditCard"	"4938580493563818"	"6861"
6	5b78bd69264dd267d4682884	"6"	"CreditCard"	"5038858453051419"	"3147"

2.6 Simulation for MongoDB individual field encryption

```
> var crypto = require('crypto'),algorithm = 'aes-256-ctr',password = 'RJ23edrf';
undefined
> function encrypt(text){
...      var cipher = crypto.createCipher(algorithm,password)
...      var crypted = cipher.update(text,'utf8','hex')
...      crypted += cipher.final('hex');
...      return crypted;
... }
undefined
> var text = "9957684776084846";
undefined
>
undefined
> var e = encrypt(text);
undefined
> console.log(e);
abd947094a97eaa2e0c99988f489f548
```

aes-256-ctr

```
> var crypto = require('crypto'),algorithm = 'aes-256-cbc',passw
undefined
> function encrypt(text){
...      var cipher = crypto.createCipher(algorithm,password)
...      var crypted = cipher.update(text,'utf8','hex')
...      crypted += cipher.final('hex');
...      return crypted;
... }
undefined
> var text = "9957684776084846";
undefined
>
undefined
> var e = encrypt(text);
undefined
> console.log(e);
d0d9350e0f4125b7781501a371a0a2b7b53f64373fc74d14d5aa186728ad4482
```

aes-256-cbc

MongoDB Node.js Driver

```
> var mongo = require('mongodb');
undefined
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://localhost:27017/mydb";
undefined
> MongoClient.connect(url, function(err, db) {
...   if (err) throw err;
...   console.log("Database created!");
...   db.close();
... });
undefined
> Database created!
var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://localhost:27017/mydb";
undefined
> MongoClient.connect(url, function(err, db) {
...   if (err) throw err;
...   db.createCollection("customers", function(err, res) {
.....     if (err) throw err;
.....     console.log("Table created!");
.....     db.close();
.....   });
... });
undefined
> Table created!

> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://localhost:27017/mydb";
undefined
> MongoClient.connect(url, function(err, db) {
...   if (err) throw err;
...   var myobj = [
...     {PaymentID: '1', PaymentType: 'CreditCard', CardNumber:'abd947094a97eaa2e0c99988f489f548', AmountCharged: '2011'},
... {PaymentID: '2', PaymentType: 'CreditCard', CardNumber:'aad14b0d4e9de6a1efca9c83f680f14d', AmountCharged: '9602'},
... {PaymentID: '3', PaymentType: 'CreditCard', CardNumber:'abd8470b4598eface7ce9f81f980f84f', AmountCharged: '1638'},
... {PaymentID: '4', PaymentType: 'CreditCard', CardNumber:'a7d3440e4897e6a0eece9f86f287f14f', AmountCharged: '2440'},
... {PaymentID: '5', PaymentType: 'CreditCard', CardNumber:'a6d941064997eea1eecc9c86f389f046', AmountCharged: '6861'},
... {PaymentID: '6', PaymentType: 'CreditCard', CardNumber:'a7d04106449ae6a1e2cc9985f185f047', AmountCharged: '3147'},
...
...   ];
...   db.collection("customers").insertMany(myobj, function(err, res) {
...     if (err) throw err;
...     console.log("Number of records inserted: " + res.insertedCount);
...     db.close();
...   });
... });
undefined
> Number of records inserted: 6
```

mydb.customers

Documents | Aggregations | Schema | Explain Plan | Indexes | Validation

FILTER { field: 'value' }

INSERT DOCUMENT | VIEW | LIST | TABLE

customers

	_id ObjectId	PaymentID String	PaymentType String	CardNumber String	AmountCharged String
1	5b78bbcf264dd267d4682879	"1"	"CreditCard"	"abd947094a97eaa2e0c99988f48	"2011"
2	5b78bbcf264dd267d468287a	"2"	"CreditCard"	"aad14b0d4e9de6a1efca9c83f68	"9602"
3	5b78bbcf264dd267d468287b	"3"	"CreditCard"	"abd8470b4598eface7ce9f81f98	"1638"
4	5b78bbcf264dd267d468287c	"4"	"CreditCard"	"a7d3440e4897e6a0eece9f86f28	"2440"
5	5b78bbcf264dd267d468287d	"5"	"CreditCard"	"a6d941064997eea1eecc9c86f38	"6861"
6	5b78bbcf264dd267d468287e	"6"	"CreditCard"	"a7d04106449ae6a1e2cc9985f18	"3147"

Summary

- MySQL and MongoDB use different strategies for data encryption at rest
- Enterprise MySQL and MongoDB provide encryption resource services (but with cost)
- Individual field encryption in MySQL GPL Workbench is easier than in MongoDB Community Edition

Acknowledgements

Raghuram Srinivas and all classmates in MSDS 7330 File Organization and Database Management 401