CLIENT SIDE

Field Level Encryption in Multi-Cloud environments

KMIP, Key rotation, Key migration

Pierre Petersson
Advisory Solutions Architect EMEA

Is Multi Cloud adopted by enterprises ?

60%

Small business
(<100 Employees)

76%

Mid size companies
(101-5000 Employees)

90%

Large enterprises
(>5001 Employees)

HashiCorp State of Cloud Strategy Survey:

Agenda

1. Encryption of today: *strengths and gaps*

2. Client Side Field Level Encryption with KMIP: *what it solves for and its implementation*

3. Key management: *available options*

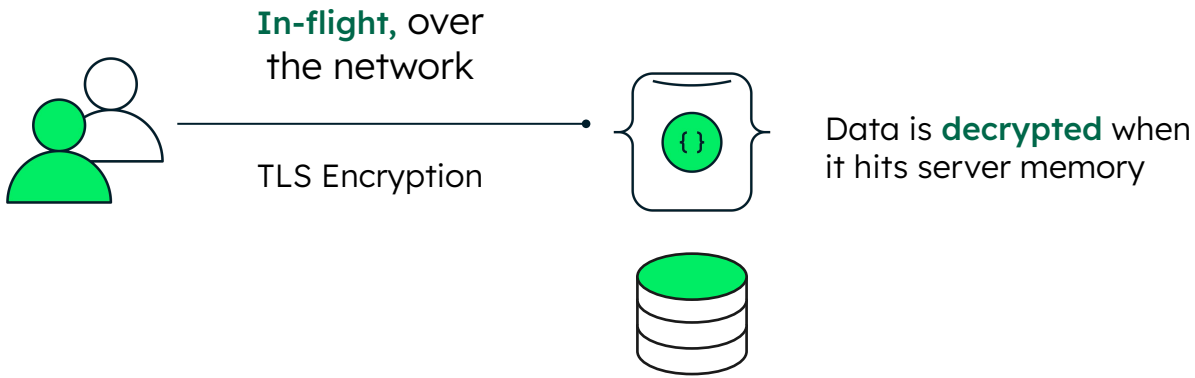
4. Demo

5. Q & A

Encryption is proven and trusted but there are gaps

Moving and storing data:

Most databases have it covered



Moving and storing data:

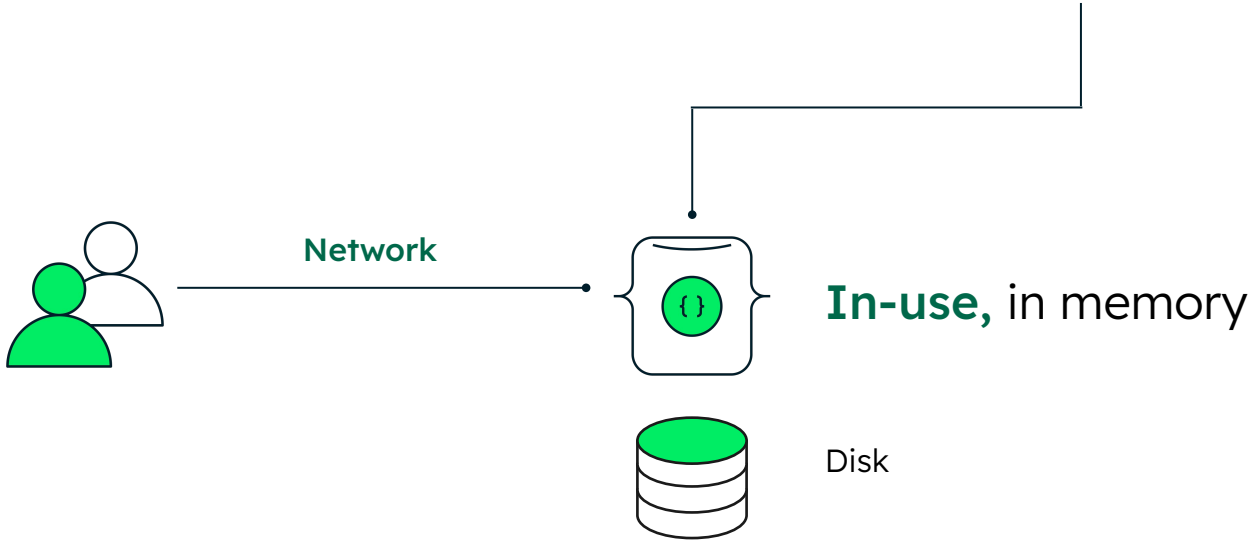
Most databases have it covered

Data is **decrypted** when it is loaded back into server memory

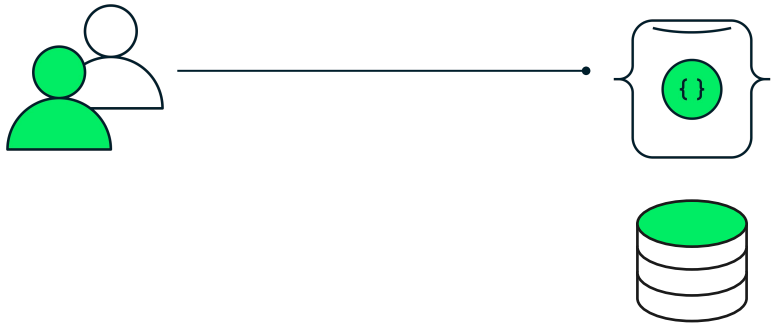
At-rest, on disk

Volume Encryption
BYO Key, Storage Engine Encryption

But what about data **here?**



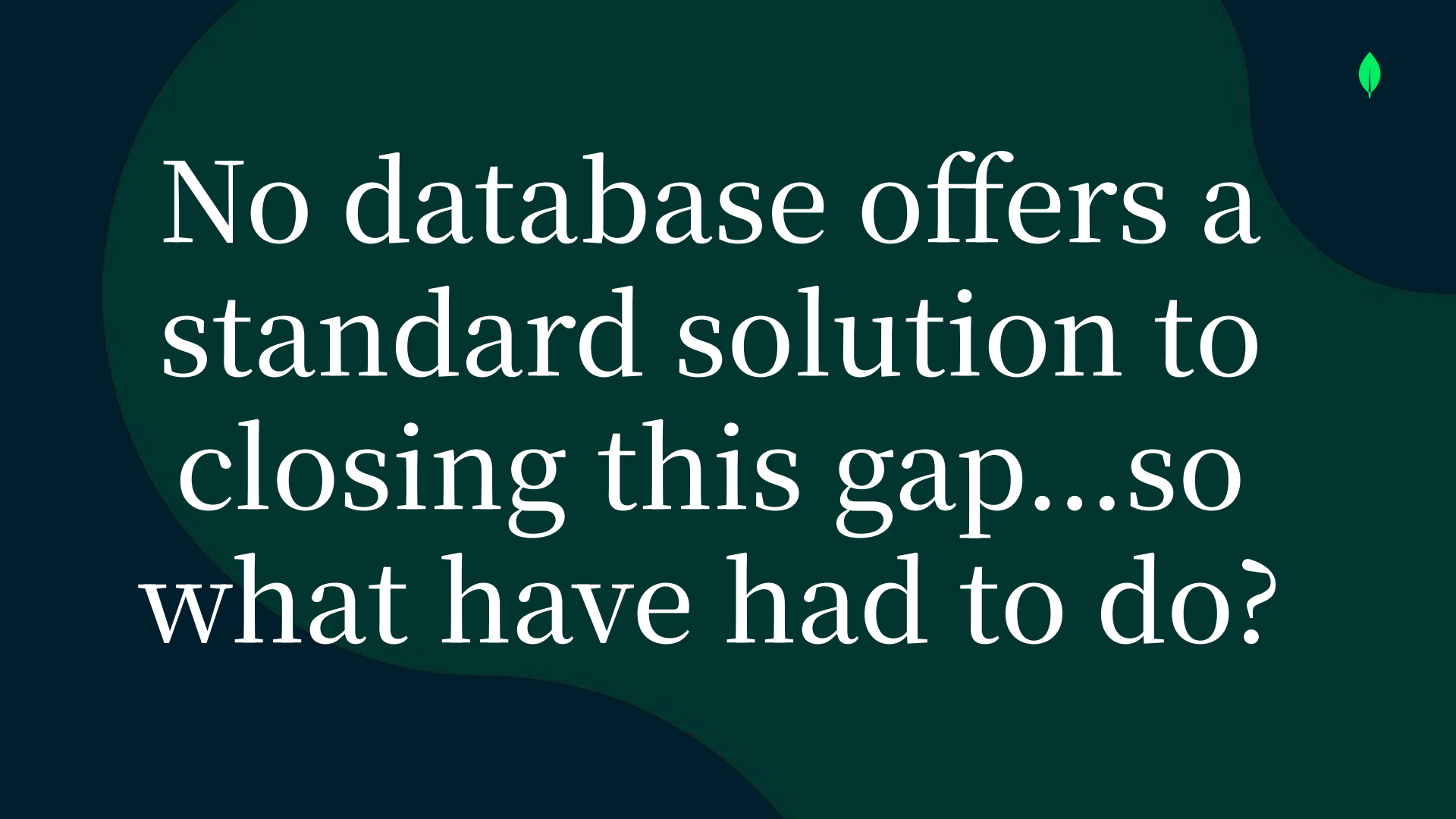
Data is in plaintext while it's being processed by the database



In-use, in memory

Data is vulnerable to insider access and active database breaches:

- Authorized and compromised administrators & users
- RAM scraping
- Process inspection


No database offers a standard solution to closing this gap...so what have had to do?

Application Level Encryption

Encrypt sensitive fields in the application, before it's sent to the database

Application Level Encryption

Encrypt sensitive fields in the application, before it's sent to the database

```
{
  firstName: "Pat",
  lastName:  "Lee",
  ssn:       "901-01-0001",
  email:     "lee@example.com",
  mobile:    "+1-212-555-1234",
  medRecNum: 235498
}
```

3 PII Fields

Go from this in the application

Application Level Encryption

Encrypt sensitive fields in the application, before it's sent to the database

```
firstName:  "Pat",
lastName:   "Lee",
ssn:        "901-01-0001",
email:      "lee@example.com",
mobile:     "+1-212-555-1234",
medRecNum:  235498
```

3 PII Fields

```
{
  firstName:  "Pat",
  lastName:   "Lee",
  ssn:        "r6EaUcgZ41Gw ... ,
  email:      "K4bSU3TlclXh ... ",
  mobile:     "oR72CW4Wf SEj ... ",
  medRecNum:  235498
}
```

To this in the database

CHALLENGES

Application Level Encryption

Highly complex: slows down application development

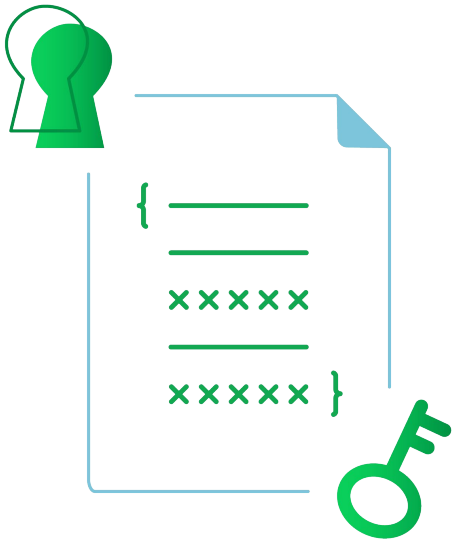
Limits application functionality, entire records returned as blobs

Compromises user experience requires additional client code

We’re stuck between a rock and a hard place

1. Do nothing and take the risk of unencrypted data in-use, in in a live database **being exploited** and the company suffering a data breach

2. Implementing application level encryption which adds **huge complexity** and friction to development processes



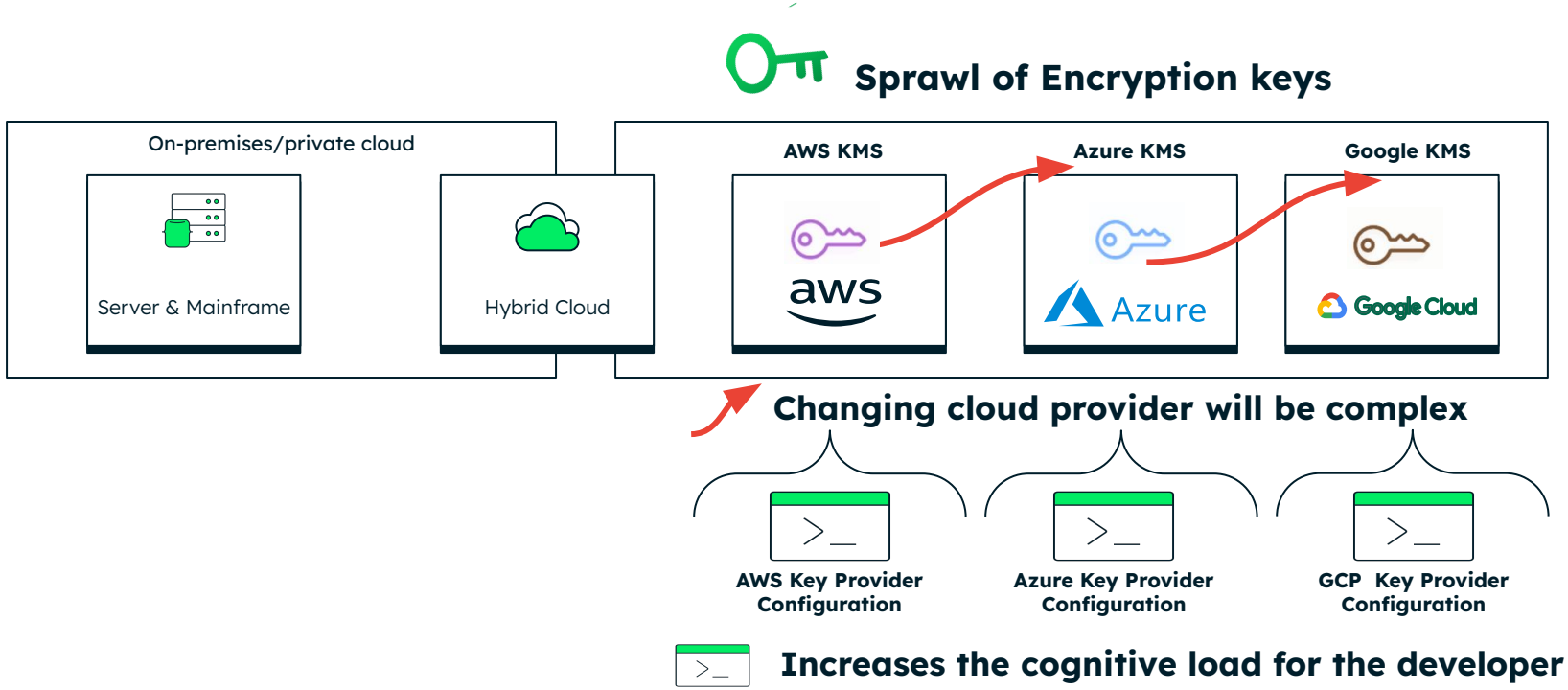
MongoDB Client Side FLE gives you a **much safer and simpler approach**

Select an encryption key, configure the fields to be encrypted in the MongoDB driver...and GO

- Sensitive data **never leaves the application** without first being encrypted. Data **remains encrypted** server-side
- No need to **modify applications**
- Encrypted data is **still queryable**
- **Minimal performance impact** to the database and the application
- **Reduce Cognitive load,** intuitive and easy for developers to configure and setup

Challenges supporting Client Field Level Encryption in Multi Cloud Environments

Challenges using CSFLE in a Multi Cloud Environment



Key Provider Options

Key Provider Options

Local

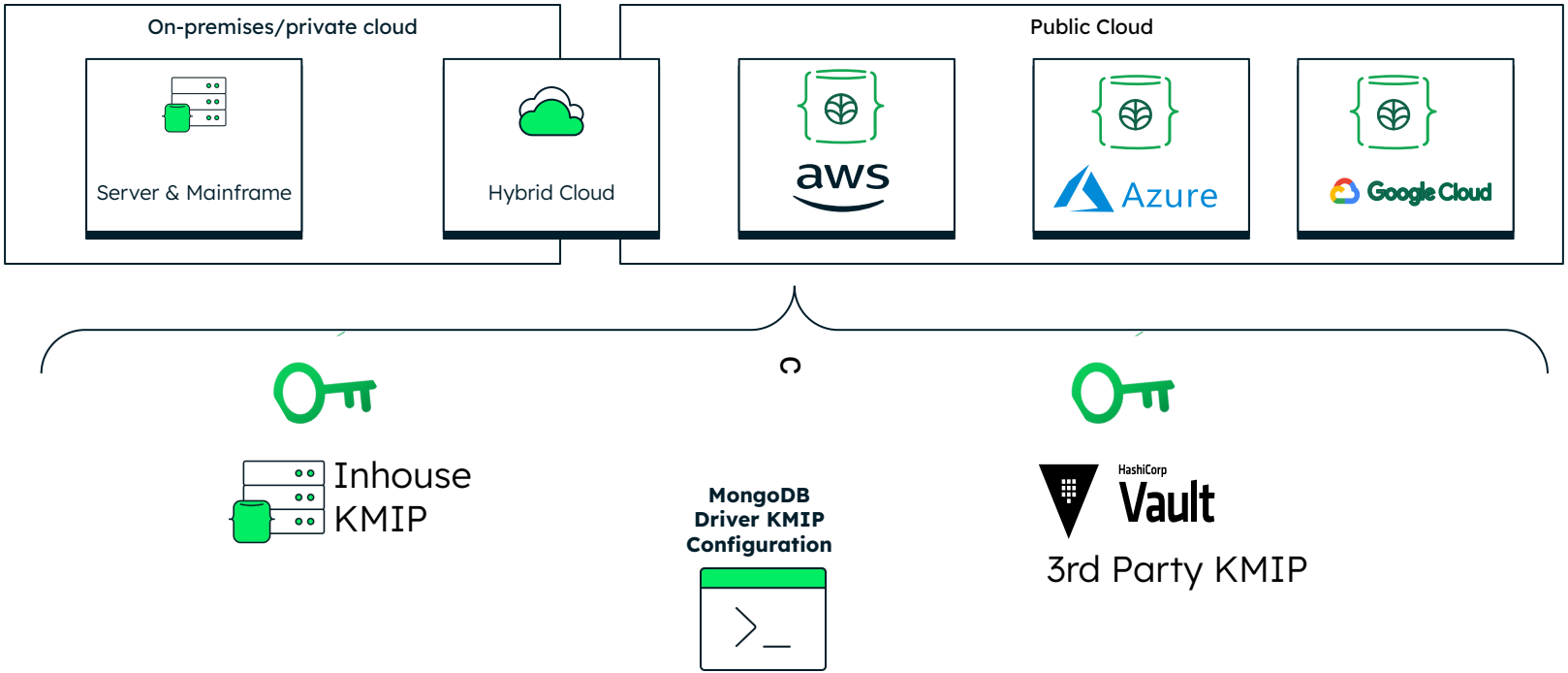
Cloud specific **K**ey **M**anagement **S**ystem

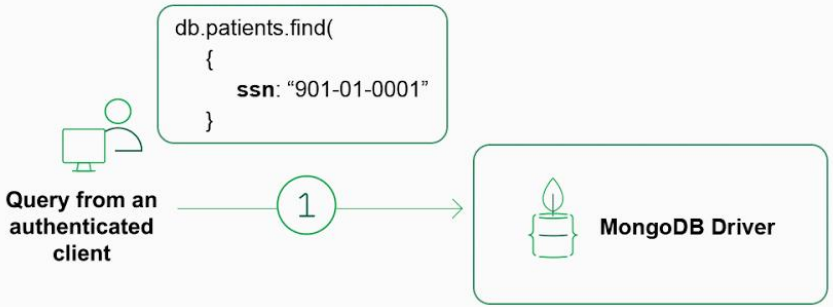
Key Management **I**nteroperability **P**rotocol - Enabled provider

KMIP and what it solves and its implementation

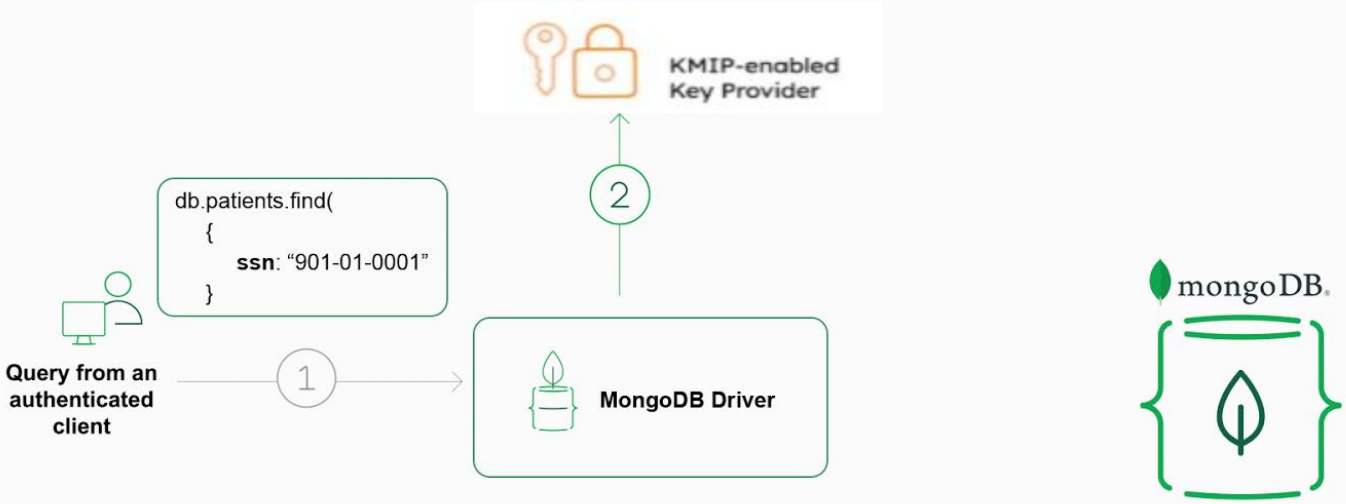
MongoDB Client Side FLE with KMIP

Allows Centralized, Cloud agnostic key management with KMIP

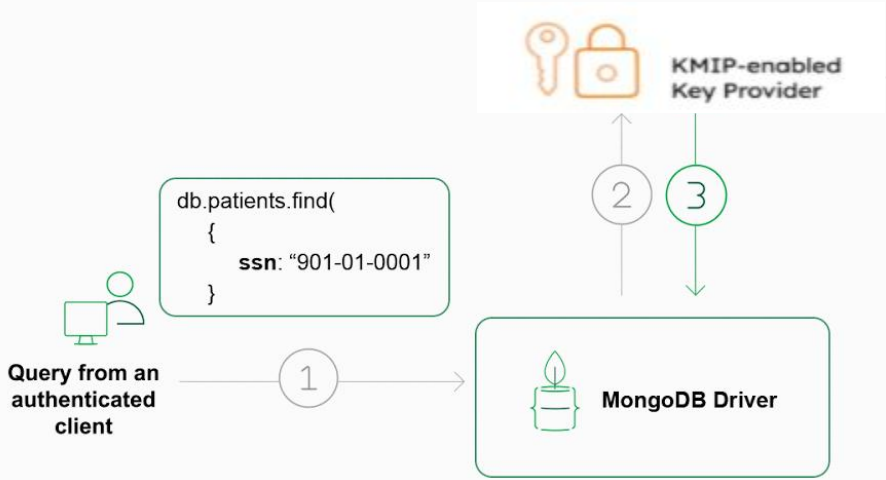




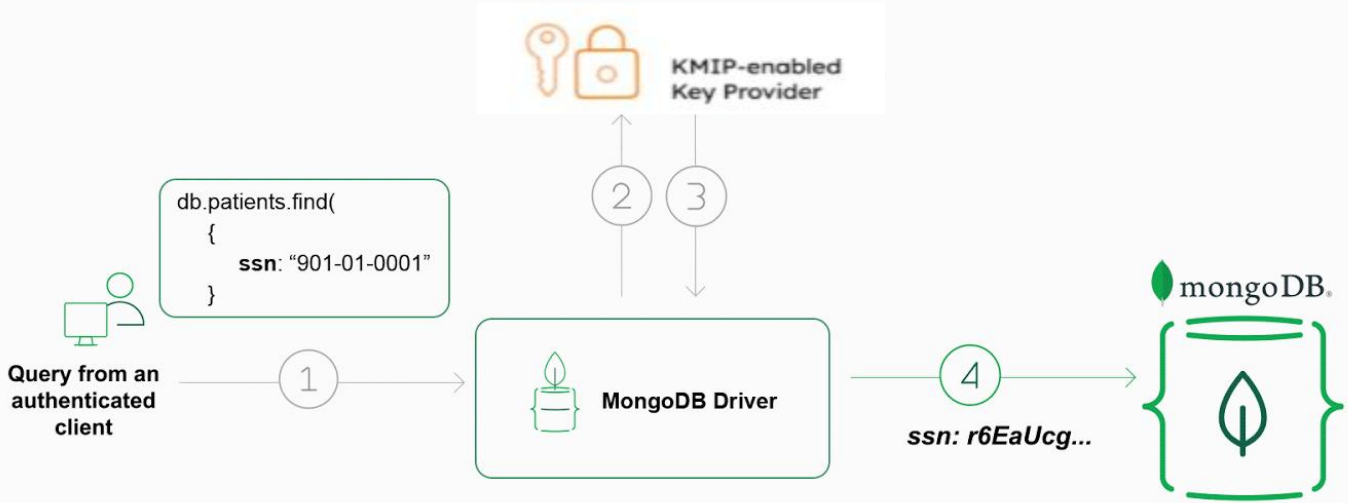
Upon receiving a query, the MongoDB driver checks to see if any encrypted fields are involved



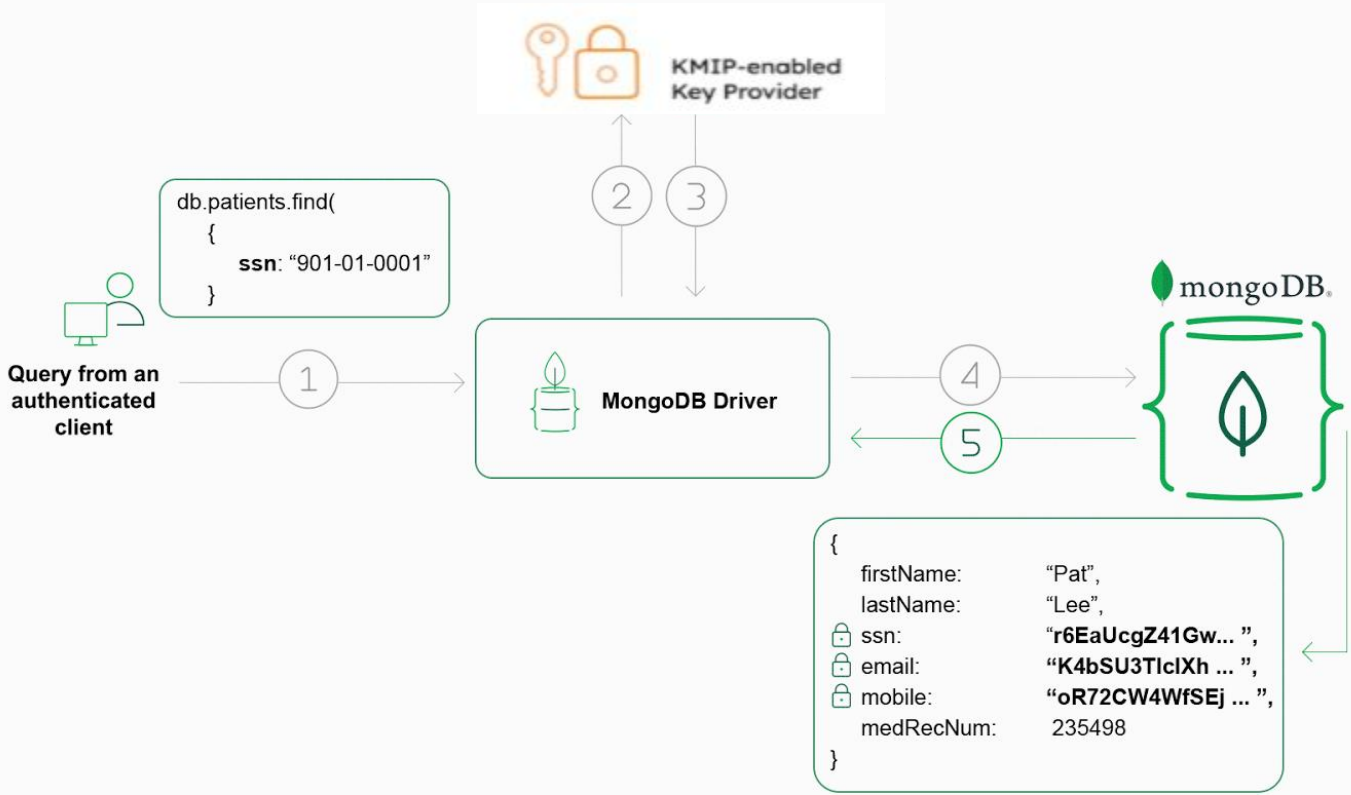
The driver requests the envelope encryption key from the KMIP key provider



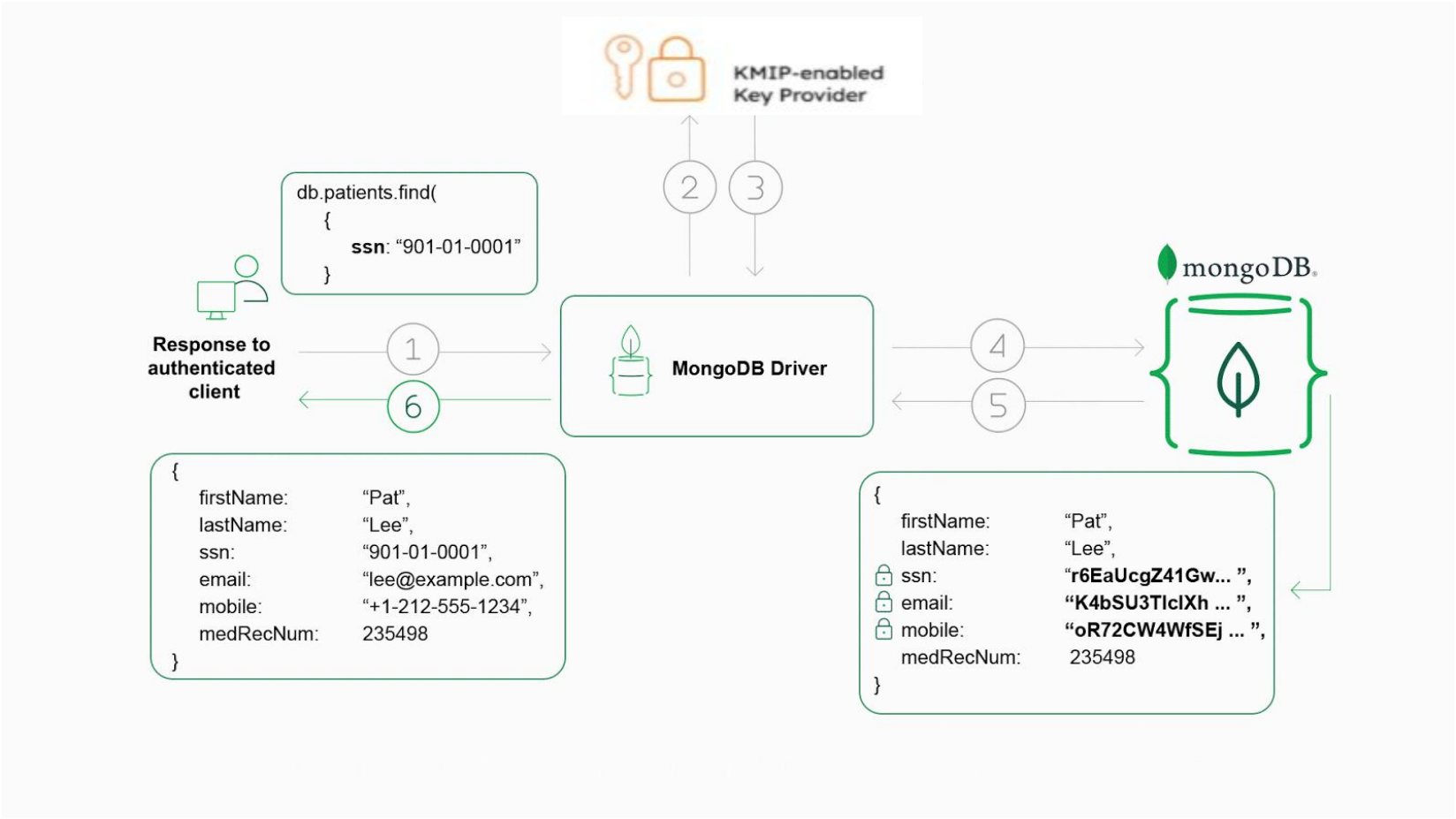
The driver decrypts the data encryptions keys using the envelope key, which then encrypts the sensitive fields



The driver submits the query to the MongoDB server with the encrypted fields rendered as ciphertext



MongoDB returns the encrypted results of the query to the driver


KMIP-enabled
Key Provider
db.patients.find(
{
ssn: “901-01-0001”
}
2
3
Response to
authenticated
client
1
6
MongoDB Driver
4
5
mongoDB.
{
firstName: “Pat”,
lastName: “Lee”,
ssn: “901-01-0001”,
email: “lee@example.com”,
mobile: “+1-212-555-1234”,
medRecNum: 235498
}
{
firstName: “Pat”,
lastName: “Lee”,
ssn: “r6EaUcgZ41Gw... ”,
email: “K4bSU3TlclXh ... ”,
mobile: “oR72CW4WfSEj ... ”,
medRecNum: 235498
}

SETUP

How to configure CSFLE with KMIP

Step 1 Configure KMIP Provider

```
kms_providers = {
   "kmip": {
       "endpoint": "localhost:5696"
   }
}
```

Step 2 Specify your Certificates

```
kms_tls_options = {
   "kmip": {
       "tlsCAFile": "./certs/vault-ca.pem",
       "tlsCertificateKeyFile": "./certs/vault-client.pem"
   }
}
```

Step 3: Create Data Encryption Key

```
data_key_id_1 = client_encryption.create_data_key(
    'kmip', key_alt_names=['pymongo_encryption_example_1'])

data_key_id_2 = client_encryption.create_data_key(
    'kmip', key_alt_names=['pymongo_encryption_example_2'])
```

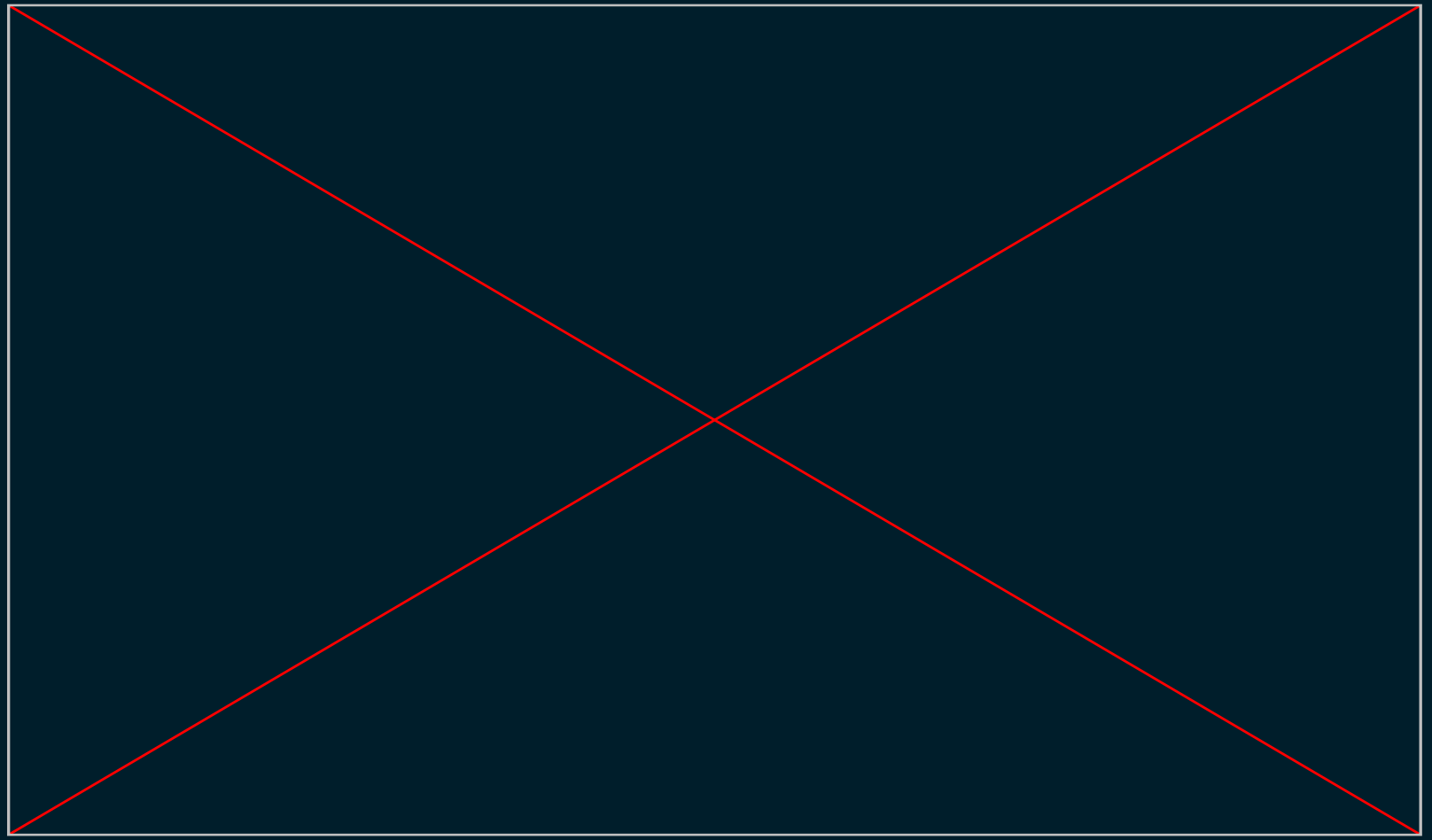
Step 4: Specify JSON Schema

```
schema = {
"bsonType": "object",
"properties": {
    "firstName": { "bsonType": "string" },
    "lastName":  { "bsonType": "string" },
    "ssn": {
        "encrypt": {
            "bsonType": "string",
            "algorithm": "AEAD_AES_256_CBC_HMAC_SHA_512-Deterministic",
            "keyId": [ data_key_id_1 ]
        }
    },
    ….
  ….
  …
```

Step 5: Run queries

```
mongo_client_csfle = MongoClient(configuration.connection_uri,auto_encryption_opts=csfle_options)
db_name, coll_name = configuration.encrypted_namespace.split(".", 1)
coll = mongo_client_csfle[db_name][coll_name]
coll.insert_one({
    "firstName": 'Alan',
    "lastName":  'Turing',
    "ssn":       '901-01-0001',
    "address": {
        "street": '123 Main',
        "city": 'Omaha',
        "zip": '90210'
    },
    "contact": {
        "mobile": '202-555-1212',
        "email":  'alan@example.com'
    }
})
```

Demo CSFLE with KMIP



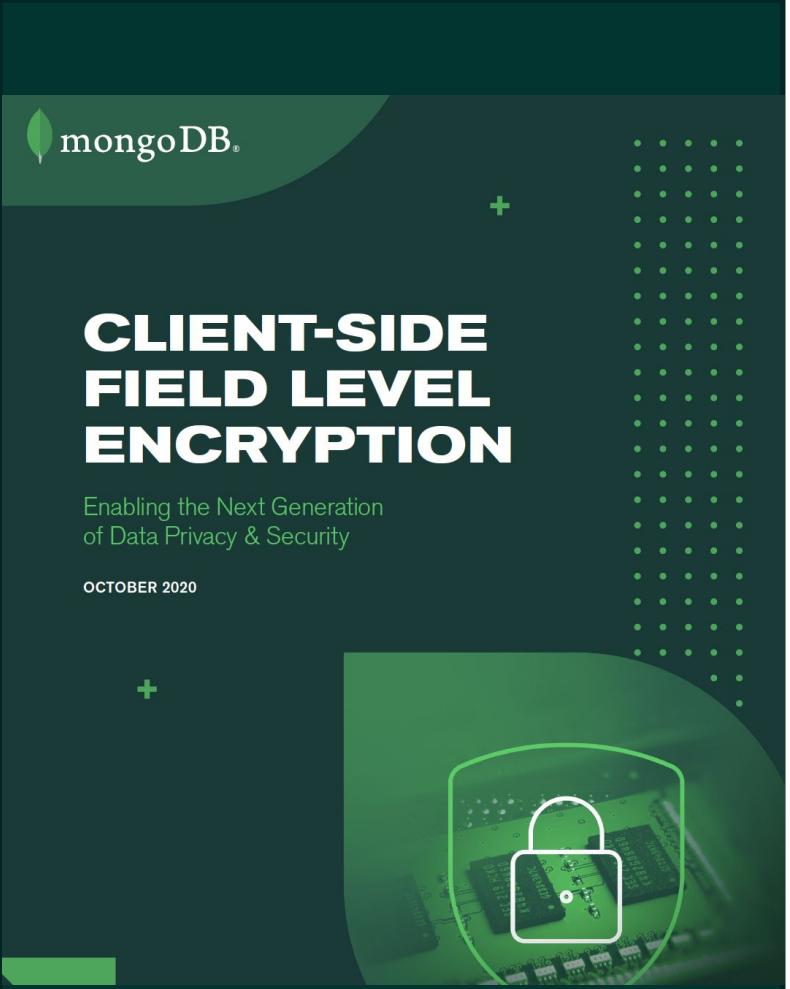
Github Repo

Key Rotation

- Now supporting key vault rotation
  - Previously, only new data encryption keys were protected by a new CMK
  - Key Vault rotation replaces all former versions of CMK seamlessly, via a single API call

Key Migration

- Changing from one key provider to another used to require decrypting and re-encrypting all of your data
- A single API call now seamlessly migrates your keys from any supported key provider to another one
  - AWS - GCP
  - Local - Azure
  - GCP - KMIP
- With no impact to your application or data



Resources to Learn More

- **Whitepaper: Enabling the Next Generation of Security & Privacy with Client-Side FLE**

  https://www.mongodb.com/collateral/field-level-encryption

- **Getting started**
  - Documentation: implementation & reference
  - Guide: worked tutorial in multiple languages for full stack developers

Your feedback matters! Please rate the session.

Client-Side Field Level Encryption in Multi-Cloud Environments

Pierre Petersson, MongoDB | .local London

Q&A

Pierre Petersson
Sr Solutions Architect MongoDB