MongoDB World 2022

CSFLE and Queryable Encryption

Pierre Petersson
Advisory Solutions Architect EMEA

Queryable Encryption and Client-Side FLE

Design Goal

Provide some of the strongest levels of data privacy and security for regulated workloads

- End to end protection of your most sensitive data
- Increase your confidence in moving to managed services in the cloud
- **Enable querying on encrypted data, in a secure, fast and scalable data (range, prefix)**
- Simplify compliance with modern privacy regs: “Right to Erasure

Agenda

Encryption of Today

Encryption in Multi Cloud Environments

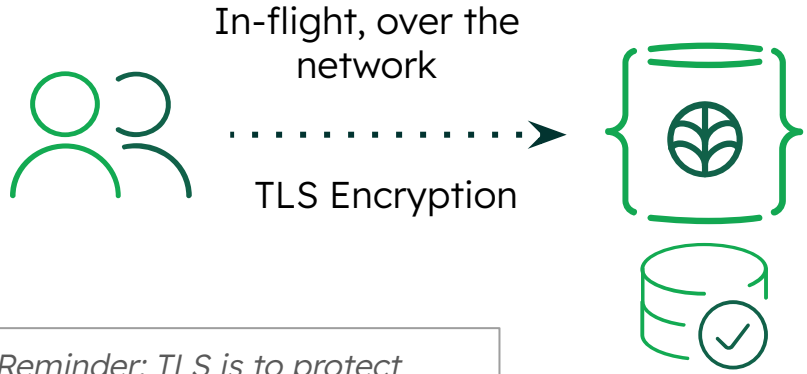
CSFLE vs Queryable Encryption

Key Management Enhancements

Q & A

Encryption is proven and trusted, but there are gaps

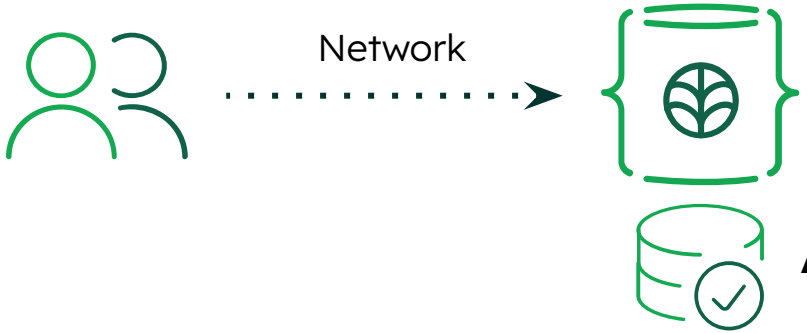
Moving and storing data: most databases have it covered



Data is **decrypted** when it's received on the DB server

Reminder: TLS is to protect against network eavesdropping

Moving and **storing** data: most databases have it covered



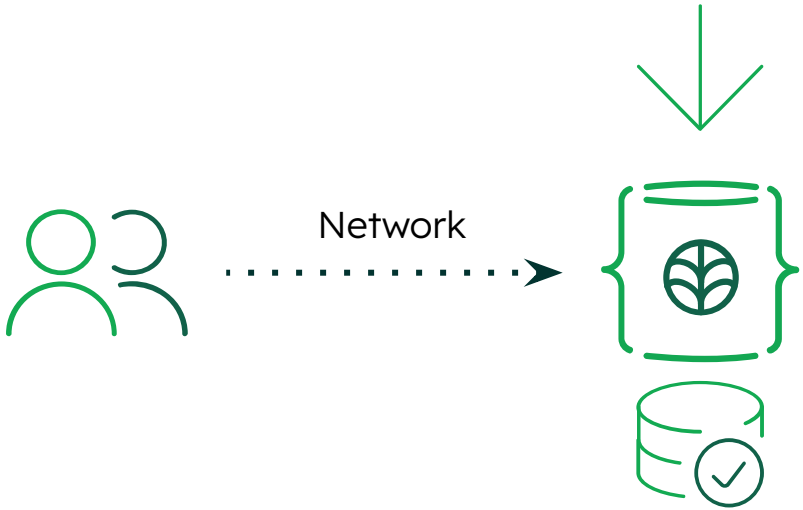
Data is **decrypted** when the DB starts up

At-rest, on disk

Volume Encryption
Storage Engine Encryption

Reminder: At-rest encryption is (mostly) to protect non-running databases & backups

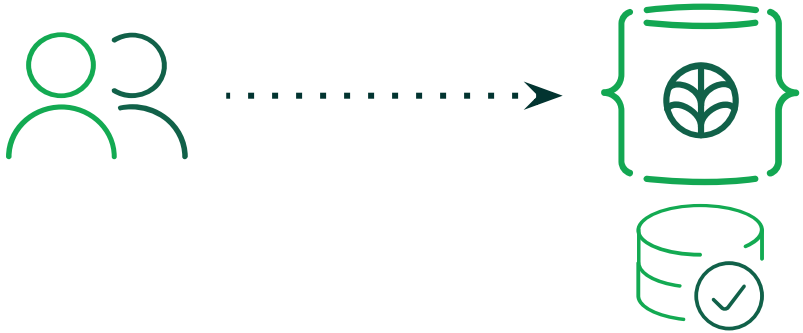
But what about data here?



In-use, in memory

Very few practical solutions exist to protect data in-use

Data is in plain text while its being processed by the database

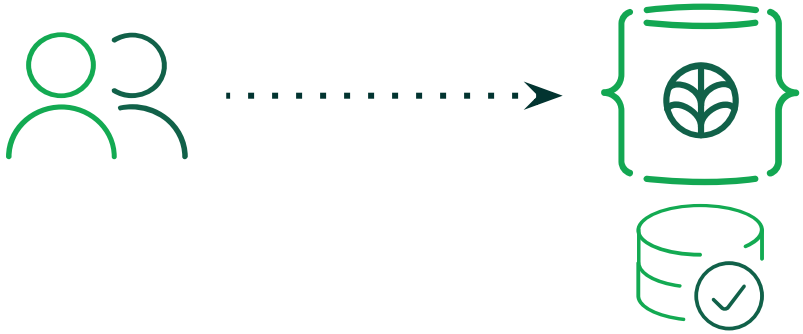


In-use, in memory

Data is vulnerable to insider access and active database breaches:

- Authorized and compromised administrators, DBAs & privileged users
- RAM scraping
- Process inspection
- Cloud providers

Data is in plain text while its being processed by the database



In-use, in memory

Data is vulnerable to insider access and active database breaches:

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

This is why we built Client-Side Field Level Encryption!

How much of a risk is this...really?

55%

Of breaches committed by organized criminal groups, including state actors

37%

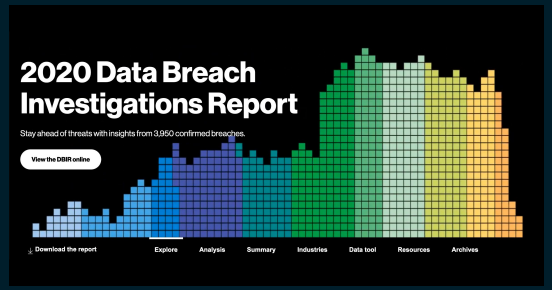
Of breaches used stolen credentials to access secure systems

30%

Of breaches were committed by internal users

verizon✓

Verizon Data Breach Investigations Report, 2020



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

Application Encryption

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

Code with no encryption

```
db.patients.insert({
   firstName: 'Jane',
   lastName:  'Doe',
   ssn: "901-01-0002",
   dob: new Date('1989-06-23'),
})
```

Code with explicit encryption

```
db.patients.insert({
firstName: 'Jane',
lastName:  'Doe',
ssn: db.getMongo().encrypt( key1 ,
"901-01-0002" , ENC_DETERM ),
dob: db.getMongo().encrypt( key1 ,
new Date('1989-06-23'), ENC_RANDOM)
})
```

CHALLENGES

Application/Explicit Level Encryption

Highly complex: slows down application development

Limits application functionality, entire records returned as blobs, **not possible to search**

Compromises user experience requires additional client code

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

Automatic Encryption

Code with no encryption

```
db.patients.insert({
   firstName: 'Jane',
   lastName:  'Doe',
   ssn: "901-01-0002",
   dob: new Date('1989-06-23'),
})
```

Code with automatic encryption

```
db.patients.insert({
   firstName: 'Jane',
   lastName:  'Doe',
   ssn: "901-01-0002",
   dob: new Date('1989-06-23'),
})
```

No difference

Automatic Encryption

Code with no encryption

```
db.patients.insert({
   firstName: 'Jane',
   lastName:  'Doe',
   ssn: "901-01-0002",
   dob: new Date('1989-06-23'),
})
```

Code with explicit encryption

```
db.patients.insert({
   firstName: 'Jane',
   lastName:  'Doe',
   ssn: "901-01-0002",
   dob: new Date('1989-06-23'),
})
```

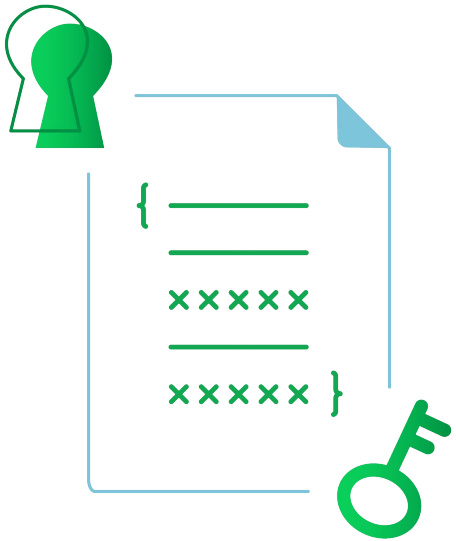
How?

Automatic encryption, available in MongoDB Atlas and Enterprise, uses JSON Schema

- Centrally defines which fields to encrypt and the algorithm to use

Benefits of Automatic Encryption

- **Higher productivity:** No additional code for developers to write, reduces chance of human error
- **Improved data governance:** Centrally enforce the encryption of sensitive fields



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, equality match**
- **Minimal performance impact** to the database and the application
- **Reduce Cognitive load,** intuitive and easy for developers to configure and setup

What about CSFLE in a **Multi Cloud** Environments

Is Multi Cloud adopted by enterprises ?

60%

Small business
(<100 Employees)

76%

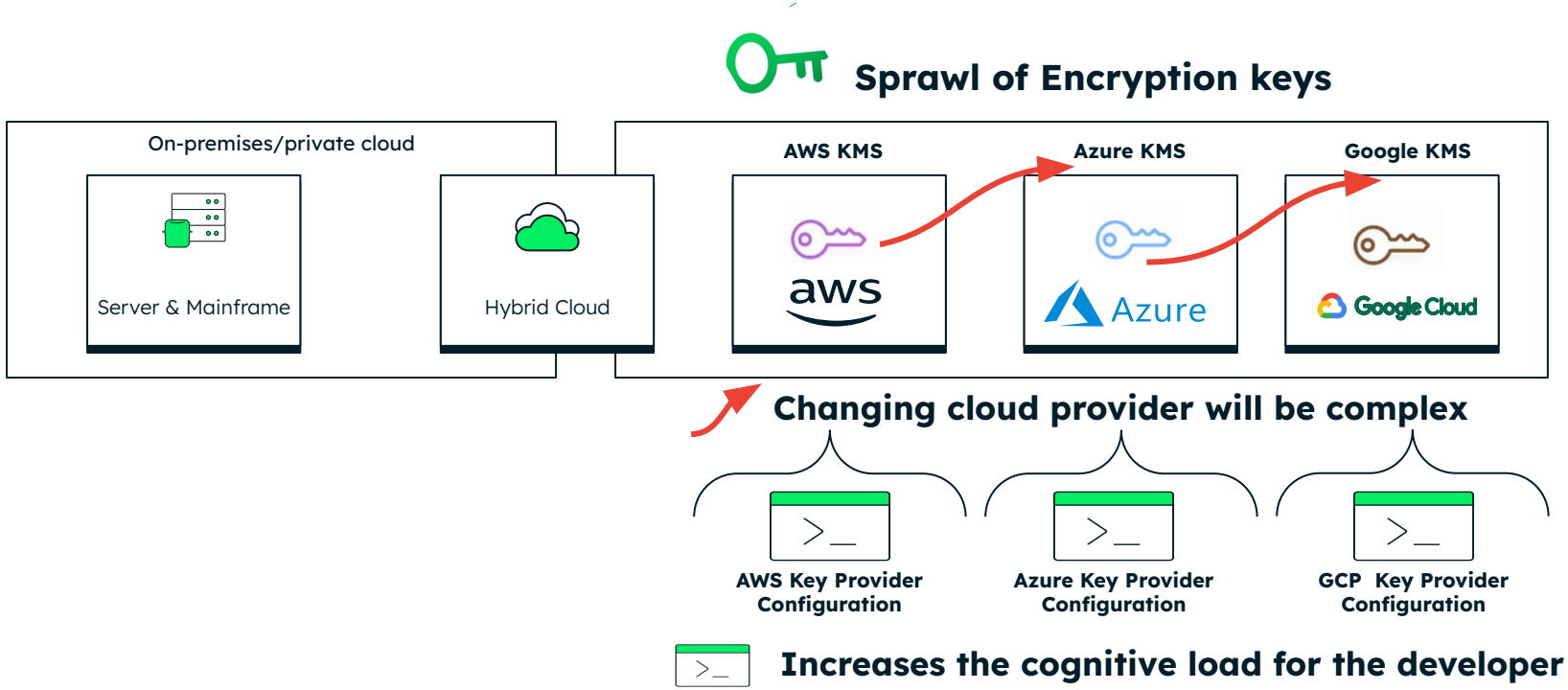
Mid size companies
(101-5000 Employees)

90%

Large enterprises
(>5001 Employees)

HashiCorp State of Cloud Strategy Survey:

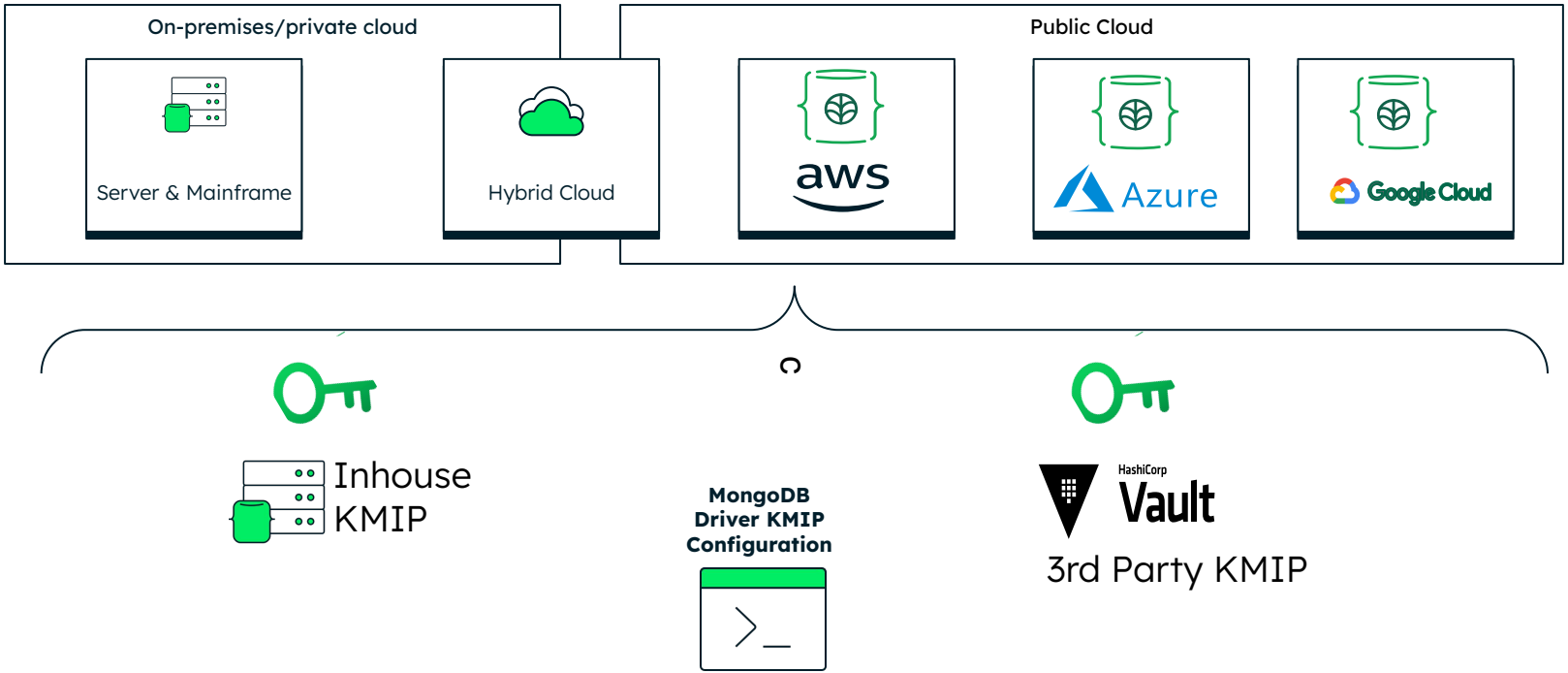
Challenges using CSFLE in a Multi Cloud Environment



CSFLE in Multi Cloud Environments

MongoDB Client Side FLE with KMIP

Allows Centralized, Cloud agnostic key management with KMIP

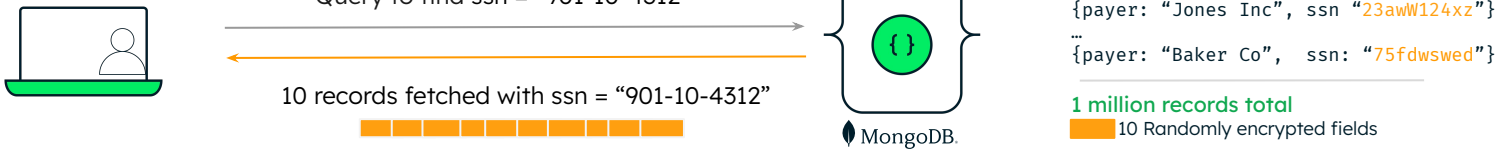


Introducing Queryable Encryption

Queryable Encryption

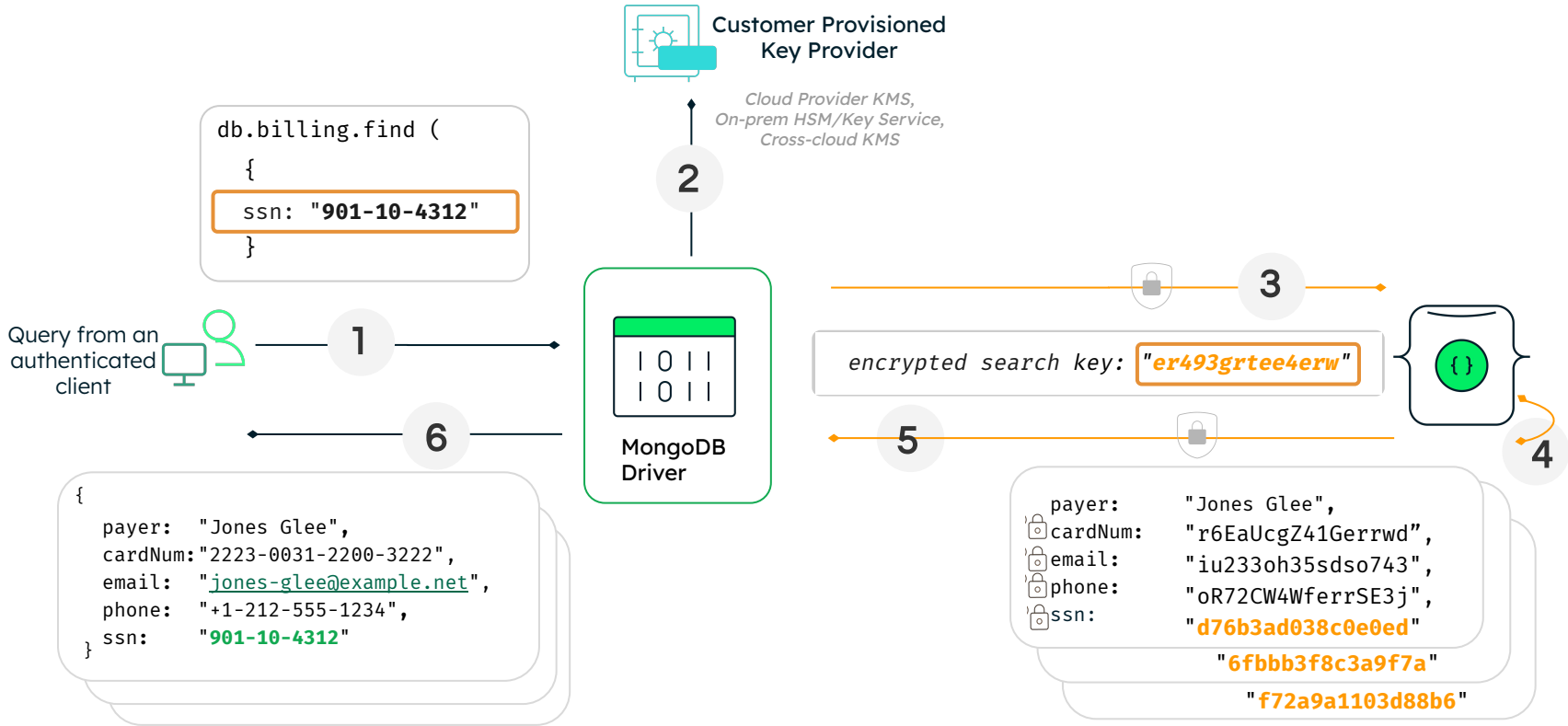
- Encrypt the sensitive data (fields)
- Easy development cycle
- No crypto experience required
- Encrypted throughout the data lifecycle
- **Rich expressive queries**

- MongoDB is the only platform to implement fast searchable encryption scheme
- Server-side processing of encrypted data
- Server does not know anything about the data



MongoDB’s Approach

Let's look closer



Encrypted fields are always stored, transmitted, processed, and retrieved as ciphertext, including queries

Use Cases

Industry: Financial Services
Bank application needs to find transactions using a **range of dates** or **dollar amounts** for fraud detection

Industry: Human Resources
HR system allows searching for employees by the **last 4 digits** of their social security number

Industry: Health Care
Customer support agents needs to find patient records by searching for the **first few characters** of their name

Queryable Encryption – Key Benefits

Reduce institutional risk

Confident in storing and processing your sensitive workloads in MongoDB Atlas (Cloud)

Faster app development

No crypto experience required
Intuitive and easy for developers to set up and use

Rich querying on encrypted data

Run expressive queries like range, equality, prefix, suffix, substring, and more on encrypted data

Strong technical controls for critical data privacy use cases

Meet the strictest data privacy requirements for confidentiality on security critical workloads

End-to-end fully randomized encryption

Data never exists in the clear outside of the client
Dramatically reduces attack surface

Ground-breaking query technology, standards-based cryptography

Based on strong, standards-based cryptographic primitives

Our journey & roadmap

Our Journey & Roadmap


Seny Kamara


Tarik Moataz

Aroki Systems acquisition

Pioneers in Encrypted Search

2019

Client-Side Field Level Encryption (CSFLE)

Equality search on Deterministic encryption

2021

Aroki Systems acquisition

Pioneers in Encrypted Search

May 2022

Formation of Advanced Cryptography Research group

Seny Kamara, Tarik Moataz, and a team of PhD cryptography researchers

June 2022

Queryable Encryption Preview

Structured Encryption core functionality; *Equality search* on randomized encryption

Post 6.0

Queryable Encryption v1.1

Addition of *Range* query capabilities

Post 6.0

Queryable Encryption v1.2

Addition of *prefix,suffix, substring* query capabilities

Future

New privacy-enhancing cryptography capabilities

What is in 6.0 Public Preview?

Crypto framework

Foundational work that enables equality and future query types

Equality

Equality comparisons on randomly encrypted data

Compass

Queryable Encryption Configuration

Decryption

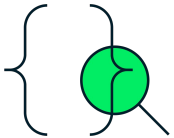
What is a Public Preview?

- Available with 6.0 RC release
  - Evaluation only
  - May be breaking changes
  - Not recommended for production workloads

Post 6.0 - Additional Query Types

Range

Prefix/Suffix



Substring

CSFLE vs Queryable Encryption

Both Are Supported

CSFLE is NOT being deprecated

CSFLE to Queryable Encryption migration not (yet) supported

CSFLE workloads should stay on CSFLE

Queryable Encryption net new only

Must be specified at collection creation

CSFLE

- Client-side encryption
- Server is (largely) unaware
- Queryability
  - Equality only - Deterministic
    - Data leakage on low entropy fields
- Flexible key usage
  - unique key per field
  - 1 key for all fields
  - per-document keys
- No additional data elements

Queryable Encryption

- Client-side encryption
- Server is integral
- Queryability
  - New functional search index
  - Equality - Fully random
    - No snapshot leakage, even on low entropy fields
  - Range, prefix, suffix and substring
- Requires a unique key per field
- Additional data
  - 1 new field per document
    - `__safeContent__`
  - 3 new system collections: `enxcol_.*`
  - **Do not modify any of these!**

CSFLE vs Queryable Encryption Trade-offs

	Inserts	Find (equality)	Find (range, prefix, suffix, substring)	Storage Overhead	Frequency Leakage
FLE	Fast	Fast	No	Minimal	Possibly
Queryable Encryption	Slower	Fast	Yes	Yes	None

Key Management Enhancements

(Coming Soon)

Key Rotation

- Rotate the entire MongoDB key vault
  - 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

Additional Resources

Resources CSFLE

Client-Side Field Level Encryption The Next Generation of Privacy & Security MDBW22 Video

Whitepaper CSFLE

CSFLE Multi Cloud Environments MDBW22 Video

Queryable Encryption Resources

Docs (very much a WIP): https://www.mongodb.com/docs/upcoming/core/queryable-encryption/
Blog post: https://www.mongodb.com/blog/post/mongodb-releases-queryable-encryption-preview
Product page & FAQ: https://mongodb.com/products/queryable-encryption

Thank you!

When to use Queryable Encryption

When you need to do rich queries and not just equality match

If requirements are:

Range queries

E.g Date range

Substring Queries

E.g query part of string

Prefix Queries

E.g query on starts with

Suffix Queries

E.g query on ends with

No frequency leakage

Randomized ciphertext more secure

When **to** use CSFLE Encryption

If requirements are:

Fast inserts

Only one collection needs to updated in contrast to 3 with Queryably Encryption

Flexible key usage

One Key per field, one key per collection or combination.

Only Equality match

E.g query on exact match using deterministic algorithm

Frequency leakage

If some leakage is acceptable

Right to be forgotten

Define key name as customer id and utilize json pointers