PROJETO FINAL DE GRADUAÇÃO

Avaliação de solução de Searchable Encryption

Isabela Lobato Braga

Brasília, Setembro de 2022

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

UNIVERSIDADE DE BRASÍLIA
Faculdade de Tecnologia

PROJETO FINAL DE GRADUAÇÃO

Avaliação de solução de Searchable Encryption

Isabela Lobato Braga

Relatório submetido ao Departamento de Engenharia Elétrica como requisito parcial para obtenção do grau de Engenheira de Redes de Comunicação

Banca Examinadora

Prof. Rafael Timoteo de Sousa Júnior, ENE/UnB _____
Orientador

Prof. Robson de Oliveira Albuquerque, ENE/UnB _____
Co-orientador

Prof. Fábio Lúcio Lopes de Mendonça, ENE/UnB _____
Examinador Interno

Prof. Georges Daniel Amvame Nze, ENE/UnB _____
Examinador Interno

Prof. Daniel Alves da Silva _____
Examinador Externo

Agradecimentos

Primeiramente, agradeço aos meus pais, Rosynubia Gonçalves Lobato e Jackson Braga Rodrigues, pela ajuda e apoio que prestaram durante toda a minha vida. O incentivo deles, associado ao privilégio de estudar, foram fundamentais para a conclusão deste ciclo.

Gostaria de agradecer à minha irmã, Letícia Lobato Braga, por todos os conselhos e amizade incondicional, por me encorajar nos momentos mais difíceis e por acreditar tanto em mim.

Ao meu namorado e melhor amigo, Pedro Silveira Rosa, agradeço por dividir essa trajetória comigo, em fases boas e ruins, sendo o meu companheiro e compartilhando grandes desafios.

Às minhas amigas, Paula de Medeiros Daniel e Gabrielle Carneiro Soares, por sempre me lembrarem da minha essência e por toda a força que me transmitiram ao longo da minha graduação.

Aos meus amigos de universidade, em especial ao Bruno Brandão Vieira do Norte, por todo apoio e troca de experiências que me permitiram crescer como pessoa.

Aos meus professores Rafael Timoteo de Sousa Júnior e Robson de Oliveira Albuquerque, pela ajuda, os ensinamentos e a paciência que contribuíram de forma significativa para a realização deste trabalho.

Agradeço o apoio técnico e computacional do Laboratório de Tecnologias para Tomada de Decisão - LATITUDE, da Universidade de Brasília, que conta com apoio do CNPq - Conselho Nacional de Pesquisa (Outorgas 312180/2019-5 PQ-2 e 465741/2014-2 INCT em Cibersegurança), do Ministério da Economia (Outorgas 005/2016 DIPLA e 083/2016 ENAP), do Conselho Administrativo de Defesa Econômica (Outorga CADE 08700.000047/2019-14), da Advocacia Geral da União (Outorga AGU 697.935/2019), do Departamento Nacional de Auditoria do SUS (Outorga DENASUS 23106.118410/2020-85), da Procuradoria Geral da Fazenda Nacional (Outorga PGFN 23106.148934/2019-67) e dos Decanatos de Pesquisa e Inovação e de Pós-Graduação da Universidade de Brasília (Outorga 7129 FUB/EMENDA/DPI/COPEI/AMORIS).

Isabela Lobato Braga

Resumo

O crescimento do volume de dados trafegados pelos meios de comunicação, assim como a preocupação relacionada à privacidade da informação, motivaram a adequação de novas tecnologias. O armazenamento de dados e o poder de processamento atrelado aos recursos, precisam estar em conformidade com a grande volumetria de informação tratada. Assim, a oferta de serviços na computação em nuvem obteve espaço por proporcionar, de forma prática, o acesso a recursos com escalabilidade, alto desempenho e interoperabilidade.

Nesse sentido, os detentores dos dados passaram a terceirizar o armazenamento das informações para os ambientes *cloud*. O nível de segurança e privacidade nesses ambientes se tornaram relevantes objetos de pesquisa. Isso ocorre, pois se faz necessário garantir que os dados armazenados nos servidores de terceiros não estejam expostos a possíveis violações de segurança.

A criptografia é uma solução comum para proteger a confidencialidade dos dados e lidar com possíveis ameaças de segurança. Além disso, a criptografia em nível do lado do cliente é um processo no qual os dados são criptografados antes de serem enviados ao banco de dados. No entanto, este processo modifica os dados e dificulta a execução de consultas. Diversos estudos foram realizados para encontrar maneiras de viabilizar um banco de dados criptografado pesquisável, por exemplo o Searchable Encryption.

O esquema de Searchable Encryption fornece um novo método de criptografia e consulta em dados criptografados para diferentes tipos de informação. O presente trabalho consiste na avaliação de uma solução baseada em Searchable Encryption aplicada ao serviço *multi-cloud* do MongoDB. Será analisado o desempenho e a eficiência dos bancos de dados a partir do uso desta tecnologia.

Palavras-chave: Searchable Encryption, Queryable Encryption, MongoDB, Computação em nuvem, Segurança

Abstract

The increase in the volume of data transmitted by the media, as well as the concern related to information privacy, caused the appropriateness of new technologies. The data storage and the processing power tied to resources, must comply with the large volume of information processed. Thus, the service offering in cloud computing gained space for providing, in a practical way, access to resources with scalability, high performance and interoperability.

In this sense, the data-holders began to outsource the storage of information to cloud environments. The level of security and privacy in these environments have become relevant research objects. This is because it is necessary to ensure that the data stored on third-party servers are not exposed to possible security breaches.

Cryptography is a common solution to protect data confidentiality and deal with potential security threats. Also, client-side encryption is a process in which data is encrypted before being sent to the database. However, this process modifies the data and makes it difficult to execute queries. Several studies have been carried out to find ways to make a searchable encrypted database feasible, for example Searchable Encryption.

The Searchable Encryption scheme provides a new method of cryptography and querying on encrypted data for different types of information. The present work consists of the evaluation of a solution based on Searchable Encryption applied to MongoDB's multi-cloud service. It will be determined the performance and efficiency of databases using this technology.

Keywords: Searchable Encryption, Queryable Encryption, MongoDB, Cloud computing, Security

LISTA DE FIGURAS

LISTA DE SIGLAS

Acrônimos

AES	Advanced Encryption Standard
DES	Data Encryption Standard
RSA	Rivest-Shamir-Adleman
PHE	Partially Homomorphic Encryption
SE	Searchable Encryption
SSE	Searchable Symmetric Encryption
PEKS	Public Key Encryption with Keyword Search
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
AWS	Amazon Web Services
SQL	Structured Query Language
NoSQL	Not Only SQL
JSON	JavaScript Object Notation
BSON	Binary JSON
KMS	Key Management Service
CSFLE	Client-Side Field Level Encryption
AEAD	Authenticated Encryption with Associated Data
CBC	Cipher Block Chaining
MAC	Message Authentication Code
SSN	Social Security Number
TSL	Transport Layer Security
SSL	Secure Sockets Layer
DNS	Domain Name System
IP	Internet Protocol
RAM	Random Access Memory
CPU	Central Processing Unit
vCPU	Virtual CPU
LGPD	Lei Geral da Proteção de Dados
ID	Identificador Único

SUMÁRIO

Capítulo 1

Introdução

O crescente avanço das tecnologias atrelado às transformações sociais e econômicas da sociedade contemporânea, demarcaram um novo cenário em relação ao uso da informação e aos grandes volumes de dados disponíveis (HEATH; BIZER, 2011). Instituições governamentais, empresas e várias organizações espalhadas pelo mundo coletam diariamente informações e dados que sejam relevantes ao seu contexto. Estes dados podem auxiliar nas tomadas de decisões baseadas na realidade de um negócio, como também, podem ser utilizadas para identificar ou prever falhas em qual seja a instituição (BRITO; MACHADO, 2017).

Com o maior fluxo de informações através dos meios de comunicação, os dados coletados estão expostos e sujeitos ao vazamento. Desse modo, possibilita que indivíduos possam ser facilmente identificados, e assim, terem a sua privacidade violada (BRITO; MACHADO, 2017), visto que, esses dados fornecem informações sensíveis, tais como nome, CPF, endereço, telefone, orientação sexual, entre outras. Com isso, é necessário proteger os dados dos indivíduos e buscar esquemas e ferramentas para preservar essas informações e não permitir a violação da sua privacidade.

Atualmente, existem técnicas de proteção de dados que visam ocultar informações sigilosas de pessoas que não possuem autorização de acesso. A criptografia é uma das formas de obter a segurança em um sistema computacional a partir de conceitos de confidencialidade, integridade, autenticação e irretratabilidade. Um sistema criptográfico utiliza um processo para cifrar ou codificar informações legíveis através de um algoritmo de criptografia referenciado por uma chave. Dessa forma, é possível converter um texto aberto (texto simples) em um texto ilegível (texto cifrado), ou também realizar o processo oposto e recuperar a informação original (ORDONEZ; PEREIRA; CHIARAMONTE, 2005).

Em um cenário que se torna necessário processar as informações e mantê-las protegidas em relação à privacidade e segurança, a criptografia passa por novos desafios que precisam ser assegurados. Antigamente, realizar operações em dados criptografados era considerado inviável, pois gerava um alto custo no processamento dos dados e exigia um nível elevado de memória computacional para as aplicações. No entanto, a partir dos avanços da tecnologia somados a preocupação com a segurança da informação, novas discussões relacionadas à criptografia homomórfica foram motivadas (GARIBALDI JUNIOR, 2018).

Assim, a criptografia homomórfica surgiu para possibilitar procedimentos computacionais sobre um texto cifrado sem que seja necessário decifrá-lo. Dessa forma, é possível manter a privacidade e a segurança dos dados durante todo o processo computacional, preservando o sigilo das informações (GARIBALDI JUNIOR, 2018).

Diante desse contexto, surge também um novo paradigma relacionado ao armazenamento do grande volume dessas informações. A terceirização do aprovisionamento de dados para ambientes híbridos de *multi-cloud*, apresenta-se como uma solução eficaz devido ao modelo de pagamento por uso, além dos recursos de escalabilidade, desempenho e interoperabilidade (RAI; SAHOO; MEHFUZ, 2015). Porém, por não ser totalmente transparente o nível de segurança e a privacidade desses ambientes, manifesta-se uma preocupação a possíveis violações de dados.

O Searchable Encryption (SE) é uma abordagem coerente nesse cenário *multi-cloud*. Esta técnica criptográfica proporciona a proteção dos dados confidenciais e sensíveis de forma a preservar a capacidade de pesquisa no lado do servidor. A partir do SE, é possível que o servidor realize operações de pesquisa em dados criptografados na nuvem sem extrair nenhuma informação dos documentos (WANG; WANG; CHEN, 2016).

1.1 Objetivos

O trabalho visa propor e implementar a solução de um sistema seguro de criptografia aplicada à banco de dados. O sistema será implementado a partir de um esquema de Searchable Encryption denominado Queryable Encryption. O provedor de banco de dados será o MongoDB Atlas, um banco NoSQL, hospedado na nuvem. O objetivo geral do trabalho é criptografar os dados do lado do cliente, armazená-los no banco de dados e realizar consultas simples e complexas do lado do servidor sem que esses dados sejam descriptografados.

1.1.1 Objetivos Específicos

1.1.1.1 Implementar um sistema com aplicação do esquema Queryable Encryption

Inicialmente, será configurado o ambiente de desenvolvimento a partir dos seguintes requisitos:

- Baixar a biblioteca compartilhada de criptografia automática do MongoDB;
- Instalar o driver compatível à Queryable Encryption, para o presente trabalho, utiliza-se o PyMongo na versão 4.2.0;
- Instanciar um cluster no MongoDB Atlas.

1.1.1.2 Implementar três cenários diferentes de criptografia em banco de dados

Afim de estabelecer cenários comparativos para posterior análise, um dataset único será inserido em três bancos de dados distintos:

- Banco de dados não criptografado;
- Banco de dados parcialmente criptografado;
- Banco de dados totalmente criptografado.

1.1.1.3 Comparar os cenários implementados

Será realizado um estudo comparativo a partir das métricas geradas pelos três cenários de banco de dados, com o objetivo de avaliar a aplicabilidade prática do esquema Queryable Encryption desenvolvido nesse trabalho.

1.1.1.4 Apresentar os resultados da implementação

Por fim, serão expostos os principais resultados da implementação do esquema de Queryable Encryption no MongoDB, bem como o estudo comparativo baseado nos três cenários de criptografia em banco de dados.

1.2 Metodologia de pesquisa

Para cumprir os objetivos deste trabalho, a metodologia de pesquisa desenvolvida ocorre em três fases.

A primeira fase tem como objetivo uma revisão bibliográfica no estado da arte para compreender de forma ampla os casos de uso de sistemas com criptografia homomórfica e a aplicabilidade da criptografia consultável. Portanto, realiza-se um estudo elaborado em materiais publicados em bases eletrônicas, bem como artigos, trabalhos de monografias, repositórios do Github, documentações das bibliotecas utilizadas, ou qualquer material relevante com acesso público.

A segunda fase consiste em definir, a partir de todo o escopo avaliado na revisão bibliográfica, quais tecnologias serão utilizadas nesse trabalho. Essa definição considera tópicos importantes como a aplicabilidade, desempenho e aderência das tecnologias ao contexto pretendido do trabalho.

Finalmente, na terceira fase implementa-se o sistema seguro de criptografia aplicada à banco de dados com a abordagem de Searchable Encryption, utilizando as tecnologias definidas anteriormente. Além disso, realiza-se testes em diferentes cenários de banco de dados para garantir um levantamento comparativo entre sistemas não criptografados, parcialmente criptografados e totalmente criptografados.

Capítulo 2

Estado da Arte e Trabalho Relacionado

O presente capítulo aborda os principais conceitos que concernem à segurança de dados a partir da criptografia, bem como conceitos teóricos de confiabilidade e integridade. Além disso, apresenta-se os principais aspectos da criptografia homomórfica e sua aplicabilidade, no contexto de um sistema com dados sensíveis. Com isso, serão expostos as definições da abordagem Searchable Encryption e do esquema Queryable Encryption. Por fim, apresenta-se uma breve descrição sobre computação em nuvem e banco de dados não relacionais com foco no MongoDB.

2.1 Criptografia

A criptografia é um processo fundamental no contexto de proteção de dados. Quando pensamos em um sistema que possui informações pessoais de um indivíduo como, por exemplo, cpf, prontuário médico e números de cartões de crédito, percebemos que em diversos aspectos temos segredos que precisam ser mantidos, seja por um quesito de privacidade ou até mesmo para garantir a nossa segurança (BURNETT, 2002).

A partir da criptografia é possível transformar textos legíveis em ilegíveis, garantindo que a informação contida no texto se mantenha secreta para quem não tenha autorização para acessá-la. Assim, para realizar o processo de criptografia é necessário recorrer aos chamados algoritmos. Eles são compostos por um conjunto de métodos e técnicas responsáveis por encriptar o texto simples em texto cifrado a partir de uma chave secreta (TERADA, 2008).

A chave secreta é o que garante que o algoritmo execute o processo de criptografia e descriptografia. Quando a mesma chave é utilizada nos dois lados do processo, ou seja, tanto para codificar a mensagem, quanto para decodificá-la, caracteriza-se uma criptografia de chave simétrica. Nesse contexto, a chave deve sempre persistir em segredo (privada) (OLIVEIRA, R. R., 2012).

Inicialmente, a criptografia moderna surgiu a partir dos ideais da chave simétrica. Na criptografia de chave simétrica, em um contexto que seja necessário compartilhar as informações secretas, será preciso compartilhar também as chaves desse processo. Assim, para evitar ruídos de segurança, um modelo de criptografia criado na década de 1970, possibilitou uma nova maneira de

realizar processos criptográficos, chamado criptografia assimétrica (OLIVEIRA, R. R., 2012).

Na criptografia assimétrica, existem duas chaves distintas: a chave privada e a chave pública. Nesse sentido, a chave pública está prontamente disponível e não precisa estar protegida, ela garante o processo de codificação da mensagem. Já a chave privada é única e exclusiva de cada agente envolvido no processo, ela garante a decodificação da mensagem (SIMMONS, 1979).

Atualmente, a criptografia é adotada em diversos tipos de sistemas. Com o uso de serviços para processamento de dados, em especial os serviços *multi-cloud*, os dados criptografados são processados e utilizados por esses sistemas. Assim, quando pretende-se realizar uma operação em um dado criptografado é necessário que o sistema detenha a chave (simétrica ou assimétrica) para descriptografar o dado. Nesse sentido, discussões acerca da possibilidade de realizar operações sobre os dados sem a divulgação da chave para um serviço externo são impulsionadas. O algoritmo criptográfico homomórfico surge como uma das principais técnicas para sanar este e diversos outros problemas da criptografia moderna (OGBURN; TURNER; DAHAL, 2013).

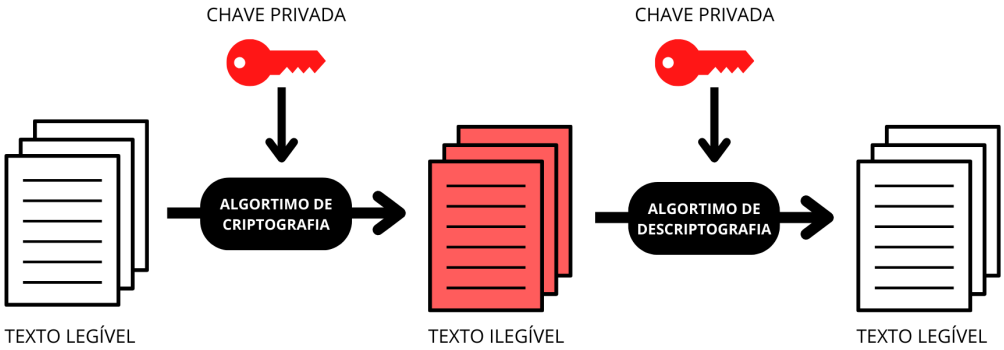
Assim, o principal objetivo da criptografia homomórfica é garantir que operações possam ser realizadas diretamente na mensagem cifrada, sem a necessidade de se descriptografar essas informações (GARIBALDI JUNIOR, 2018). A Criptografia Pesquisável (Searchable Encryption) também surge no mesmo contexto, ela permite que operações de pesquisa em dados criptografados sejam realizadas pelo servidor sem que ele extraia nenhuma informação dos dados (CURTMOLA et al., 2011).

2.1.1 Criptografia simétrica

O modelo de criptografia simétrica é o mais antigo e nele apenas uma chave é utilizada para criptografar e descriptografar a mensagem trocada entre duas partes no canal de comunicação. Esse modelo se torna promissor em um cenário com uma volumetria expressiva de dados pois, os algoritmos são simples e, por esse motivo, proporcionam melhor velocidade de processamento e implementação (AGNALDO et al., 2022).

Na figura 2.1 está representado um esquema de criptografia simétrica:

Figura 2.1: Esquema de criptografia simétrica



Fonte: elaborado pelo autor (2022).

Um desafio atrelado aos criptossistemas simétricos é o gerenciamento de chave compartilhada. Inicialmente, em um contexto com baixa quantidade de envolvidos no esquema, a sobrecarga do gerenciamento não se torna expressiva. Porém, levando em consideração que cada par incluído no sistema requer uma chave para estabelecer uma comunicação segura, a quantidade de chaves necessárias aumentaria de forma considerável e, por consequência, a administração para garantir a segurança e confiabilidade delas também (BURNETT, 2002).

Atualmente, a criptografia de chave simétrica pode ser realizada a partir de duas técnicas principais: cifragem de bloco e cifragem de fluxo (KUROSE; ROSS, 2012). Para o presente estudo, teremos como foco a cifra de bloco *Advanced Encryption Standard* (AES).

2.1.1.1 Algoritmo AES

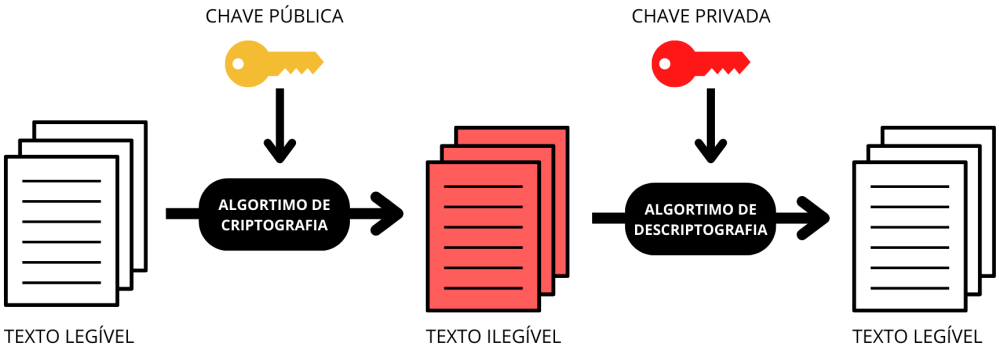
O *Advanced Encryption Standard* (AES) foi desenvolvido a partir da necessidade de se estabelecer um algoritmo mais seguro que o *Data Encryption Standard* (DES). Logo, o DES foi o algoritmo antecessor do AES (WEBER, 1995).

O algoritmo AES é uma cifra de bloco, portanto, o texto a ser criptografado é dividido em blocos, e cada bloco é cifrado de forma independente. A diferença presente nesse algoritmo é que o tamanho da chave pode variar entre 128, 192 ou 256 bits, além disso, ele apresenta uma maior agilidade em relação a outros algoritmos uma vez que algumas funções podem ser executadas em paralelo. No AES o processo de criptografar um texto cifrado não manipula operações aritméticas e isso diminui o poder de processamento necessário (ABDULLAH et al., 2017).

2.1.2 Criptografia assimétrica

Como descrito de forma breve anteriormente, na criptografia assimétrica cada agente do canal de comunicação possui um par de chaves. A figura 2.2 apresenta um esquema de criptografia assimétrica:

Figura 2.2: Esquema de criptografia assimétrica



Fonte: elaborado pelo autor (2022).

A criptografia assimétrica é composta pelos seguintes algoritmos (RIVEST, 1990):

- Algoritmo de geração de chaves: é o responsável por produzir uma chave pública e a correspondente chave privada;
- Algoritmo de criptografia: esse algoritmo recebe o texto legível e a chave pública como entradas. A saída resultante é o texto ilegível (cifrado);
- Algoritmo de descriptografia: esse algoritmo recebe o texto ilegível e a chave privada como entradas. A saída resultante é o texto legível;

O criptossistema assimétrico possui uma grande vantagem quando comparado ao modelo simétrico, pois não há a necessidade do compartilhamento das chaves secretas. Porém, os algoritmos envolvidos nesse sistema possuem uma alta complexidade e, por consequência, mais consumo de recursos associado (OLIVEIRA, R. R., 2012). Alguns dos principais algoritmos de criptografia assimétrica são os seguintes: RSA, ElGamal e Diffie-Hellman.

2.1.3 Criptografia homomórfica

Atualmente, existem inúmeras ameaças contra a privacidade e a segurança das informações no armazenamento de dados, especialmente em um cenário de terceirização para ambientes híbridos de *multi-cloud*. Portanto, manter a confidencialidade dos dados torna-se um desafio. É nesse sentido que surge a necessidade dos proprietários dos dados criptografarem os seus dados antes de armazená-los (TANG et al., 2016).

Com os dados armazenados em um formato cifrado surge um novo desafio que tange às técnicas de criptografia que permitem realizar computação em dados cifrados. Os métodos da criptografia homomórfica viabilizam esse tipo de operação.

A criptografia homomórfica foi apresentada primeiramente na literatura em 1978 por Rivest, Adleman e Dertouzos, na qual conceitua-se o termo homomorfismos privados. Esse conceito surgiu

a partir de uma limitação existente nas técnicas da Criptografia Moderna, em que para realizar operações computacionais sobre os dados, eles necessariamente precisam ser descriptografados (RIVEST; ADLEMAN; DERTOUZOS, 1978). Portanto, o objetivo da criptografia homomórfica é permitir a computação em dados criptografados para garantir a privacidade e a segurança da informação durante todo o fluxo de comunicação entre as partes envolvidas (GARIBALDI JUNIOR, 2018).

O criptossistema homomórfico, é baseado em operações de adição e multiplicação, possuindo um conjunto de algoritmos onde sua entrada é uma mensagem cifrada m e a saída uma função criptografada *f(m)*, sendo assim: dado um esquema criptográfico *E(m)* e duas entradas criptografadas, m_1 e m_2, tem-se as funções da seguinte forma (GENTRY, 2009):

- Função aditiva:

$$f(m) \rightarrow E(m_1 + m_2) = E(m_1) + E(m_2) \tag{2.1}$$

- Função multiplicativa:

$$f(m) \rightarrow E(m_1 * m_2) = E(m_1) * E(m_2) \tag{2.2}$$

Caso o esquema *E(m)* processe apenas uma das funções *f(m)*, a aditiva ou a multiplicativa, o sistema é parcialmente homomórfico. Porém, se suportam as duas, ela suporta qualquer circuito aritmético em dados criptografados e é definida como completamente homomórfica (GENTRY, 2009).

Segundo Brakerski e Vaikuntanathan (2011), o estudo de Gentry (2009) mostrou um modelo ideal da criptografia totalmente homomórfica, porém, é inerente que tal sistema é pouco eficiente comparado aos demais algoritmos de criptografia. O modelo apresentado possui uma alta complexidade de implementação devido ao aumento do volume de dados e a necessidade de uma elevada capacidade de processamento das máquinas (BRAKERSKI; VAIKUNTANATHAN, 2011).

Fazer o uso do esquema de criptografia parcialmente homomórfica (PHE) permite a realização de apenas um tipo de operação computacional na mensagem criptografada. Uma das operações que pode ser feita é a aditiva que consiste na operação em textos cifrados que resulta na versão total dos textos correspondentes. Do mesmo modo, se a operação origina uma versão criptografada do produto dos textos então, o esquema homomórfico é multiplicativo (ELGAMAL, 1985).

Os métodos PHE por permitirem apenas alguns tipos de processamento e consultas nos dados criptografados, possuem um menor custo de execução quando comparado ao método totalmente homomórfico. Porém, como o tipo de dado é específico para cada método PHE, é necessário utilizar um método diferente para cada tipo de dado. Assim, o uso desse esquema se torna complexo e muitas vezes inviável (RIVEST; SHAMIR; ADLEMAN, 1978).

2.1.4 Searchable Encryption

Apesar dos criptossistemas homomórficos não revelarem nenhuma informação ao servidor envolvido no fluxo de comunicação, a sua utilização se torna pouco eficiente pelo alto custo envolvido

em um lento processo (BOST, 2016).

A partir disso, esquemas de Criptografia Pesquisável (cujo termo na língua inglesa é Searchable Encryption) tomam destaque por apresentarem aplicabilidade para se tornarem práticos e utilizáveis, permitindo o processo de pesquisas sobre os textos criptografados. De forma prática, esse esquema consiste na codificação das informações contidas nos documentos em uma estrutura de dados e na terceirização do armazenamento dos dados para um servidor mantendo a capacidade de pesquisar sobre eles (CURTMOLA et al., 2011).

Existem duas frentes principais nos esquemas de Searchable Encryption: a Searchable Symmetric Encryption (SSE) e a Public Key Encryption with Keyword Search (PEKS). Na SSE apenas quem possui a chave privada pode criptografar os dados e realizar consultas sobre eles. Enquanto na PEKS, todos que conhecem a chave pública podem cifrar os textos e apenas aqueles que detém da chave privada podem realizar consultas sobre os dados (WANG; WANG; CHEN, 2016).

Na técnica de criptografia simétrica pesquisável (SSE), os dados consistem em um conjunto de documentos que são armazenados de forma criptografada e o cliente pesquisa combinações de palavras-chave nesses documentos. Pensando em um contexto no qual deseja-se realizar uma pesquisa de substring nos dados armazenados, no esquema SSE cada substring seria considerada como uma palavra-chave apartada. Um problema inerente a essa consulta seria o aumento significativo do armazenamento, uma vez que a criptografia simétrica pesquisável é dimensionada de forma linear ao número total de palavras-chave. Assim, o esquema SSE está em constante evolução para se manter eficiente independentemente do tipo de consulta realizada e do tipo de dado envolvido no processo (CHASE; SHEN, 2014). Uma abordagem relevante nesse contexto é a Queryable Encryption.

2.1.4.1 Queryable Encryption

O esquema Queryable Encryption pode ser descrito a partir de uma função de consulta F que recebe como entrada dois parâmetros: uma mensagem M e uma consulta q. Como saída, a função F, gera uma resposta A. Assim, o cliente criptografa a mensagem M utilizando uma chave secreta e armazena o texto criptografado em um servidor. Como o cliente detém a chave secreta ele consegue realizar uma consulta q executando um protocolo interativo com o servidor, dessa forma temos que (CHASE; SHEN, 2014):

$$F : M \times q \rightarrow A \tag{2.3}$$

A Queryable Encryption apresenta um esquema de Searchable Encryption que suporta pesquisas de igualdade, com tipos de consulta adicionais, como intervalo, prefixo, sufixo e substring. Esse esquema permite que os dados sejam criptografados do lado do cliente e armazenados de forma aleatória no lado do servidor de banco de dados. Dessa forma é possível executar consultas expressivas nos dados criptografados sem que o servidor tenha conhecimento das informações contidas nos dados processados (MONGODB, Q., 2022).

2.2 Computação em nuvem

A computação em nuvem designa um modelo que permite o acesso onipresente a um conjunto de recursos configuráveis de computação, compartilhados e conectados através da rede, de acordo com as necessidades dos clientes. Estes recursos caracterizam-se ainda pelo fato de serem alocados e liberados de forma relativamente simples e com mínimo de esforço de gerenciamento ou interação com o prestador de serviço (MELL; GRANCE et al., 2011).

As características essenciais do modelo de computação em nuvem são (MELL; GRANCE et al., 2011):

- Self-service sob demanda: o modelo permite que o cliente provisione sozinho os recursos computacionais que necessita, sem precisar de interação humana com os provedores de cada serviço.
- Amplo acesso: os recursos estão disponíveis para serem acedidos pela rede, e acessados através de mecanismos padronizados que possibilitam o uso por plataformas thin ou thin client, tais como celulares e notebooks.
- Pooling de recursos: o provedor de serviços deve conseguir satisfazer múltiplos usuários simultaneamente por meio de uma alocação dinâmica de recursos. Assim, os usuários não precisam ter conhecimento da localização física dos recursos computacionais, podendo somente especificar a localização em um nível mais alto de abstração.
- Elasticidade rápida: o modelo permite que os recursos sejam alocados de forma rápida e elástica para o usuário. Assim, o recurso, de forma automática, pode ser escalado com o aumento da demanda ou liberado na redução dessa demanda.
- Serviço medido: o modelo de serviços em nuvem deve permitir que a utilização dos recursos seja mensurável. Assim, o uso dos recursos deve ser monitorado a fim de garantir a transparência para o provedor e usuário do serviço utilizado.

2.2.1 Modelos de serviços

A computação em nuvem é composta por três modelos de serviços: o Cloud Software as a Service (SaaS), a Cloud Platform as a Service (PaaS) e a Cloud Infrastructure as a Service (IaaS) (MELL; GRANCE et al., 2011).

O modelo de serviço SaaS caracteriza-se por sistemas de software disponibilizados pelo fornecedor de serviços em que o funcionamento é suportado pela infraestrutura da cloud. Além disso, pensando em um modelo de responsabilidade compartilhada, o SaaS é o modelo que coloca a maior responsabilidade sobre o provedor de nuvem. Assim, o usuário não tem controle sobre o recurso e consegue realizar apenas algumas configurações específicas na aplicação. Alguns exemplos deste modelo são o Google Drive e o Dropbox (MELL; GRANCE et al., 2011).

No modelo PaaS o fornecedor mantém a infraestrutura física, a segurança e os sistemas operacionais que compõem uma solução na nuvem. Assim, o ambiente de desenvolvimento completo é fornecido sem a preocupação de gerenciar a infraestrutura atribuída aos recursos. O usuário tem controle sobre as suas aplicações implantadas e as configurações das aplicações hospedadas nesta infraestrutura. Um exemplo deste modelo seria o Google App Engine (MELL; GRANCE et al., 2011).

Por fim, no modelo IaaS é proposto uma flexibilização, pois esse modelo oferece ao usuário o máximo de controle sobre os recursos. Em um modelo de IaaS, o provedor de nuvem é responsável apenas por manter os recursos computacionais como o hardware, a conectividade de rede e a segurança física. Assim, a responsabilidade do usuário é maior nesse modelo, ele precisa realizar a instalação, a configuração e a manutenção do sistema operacional, bem como a configuração de rede e assim por diante. Um exemplo deste modelo é o serviço Elastic Compute Cloud (EC2) da Amazon Web Services (AWS) (MELL; GRANCE et al., 2011).

É importante ressaltar que os diferentes modelos podem ser suportados em um mesmo serviço, portanto não possuem fronteiras totalmente disjuntas (ARMBRUST et al., 2010).

2.2.2 Modelos de implementação

Em relação ao acesso e a disponibilidade dos ambientes de computação em nuvem, têm-se os seguintes modelos de utilização: a cloud privada, a cloud pública, a community cloud e a cloud híbrida (MELL; GRANCE et al., 2011).

No modelo de cloud pública, a infraestrutura da nuvem é aberta ao público de maneira geral, sendo assim, qualquer usuário que pretende comprar serviços de nuvem pode acessar e utilizar os recursos disponibilizados. A diferença fundamental entre a cloud pública e o modelo de cloud privada é justamente essa. A cloud privada é a evolução natural de um data center corporativo, ou seja, esse modelo é constituído por data centers próprios de uma única entidade de forma que o seu uso não é público (ARMBRUST et al., 2010).

Já o modelo de community cloud diz respeito às infraestruturas compartilhadas por diversas entidades que partilham de interesses em comum, seja requisitos de segurança, políticas aplicadas, entre outros. Na cloud híbrida a infraestrutura de nuvem pode ser composta por dois ou mais modelos de implementação de computação em nuvem (cloud privada, a cloud pública ou a community cloud) em um ambiente interconectado (ARMBRUST et al., 2010).

2.3 Bancos de dados não relacionais (NoSQL)

O termo NoSQL teve a sua primeira aparição em 1998 como o nome de um banco de dados relacional de código aberto criado por Carlo Strozzi. Esse nome foi designado ao banco de dados em questão por ele não utilizar o SQL como uma linguagem de consulta (SADALAGE; FOWLER, 2019).

Porém, o conceito de NoSQL que conhecemos atualmente surgiu posteriormente, a partir de um encontro realizado no dia 11 de junho de 2009, em São Francisco, nos Estados Unidos. Pesquisadores da época entendiam que o modelo de bancos de dados relacionais não era projetado para ser eficiente em armazenamento de grandes volumes de dados e que os bancos tradicionais possuíam alta complexidade ao serem escalados. (SADALAGE; FOWLER, 2019). Assim, podemos citar como os fatores principais que motivaram o surgimento do conceito de bancos de dados não relacionais: os diferentes tipos de dados na web, o processamento de grandes volumes de dados a partir do alto grau de paralelismo atribuído aos sistemas e a distribuição de sistemas em escala global (DE DIANA; GEROSA, 2010).

Percebe-se que a natureza dos dados que compõem a web pode ser semiestruturada e não estruturada, ou seja, não possuem uma estruturação padronizada, como as páginas web e conteúdos multimídias. Dessa forma, ao invés de tentar estruturar esses dados, é possível criar soluções otimizadas para eles. Assim, reconhecer a natureza desses dados não estruturados é um dos fatores que contribuiu para o surgimento de tecnologias de gerenciamento de dados diferentes das tradicionais (DE DIANA; GEROSA, 2010).

O segundo fator citado anteriormente deve-se ao contexto do processamento de grandes volumes de dados. Para atingir uma eficiência nesse processo é necessário projetar sistemas de forma estruturada a um alto grau de paralelismo. Portanto, o uso de muitos processadores baratos oferece melhor performance e também uma solução econômica mais interessante do que o uso de menos processadores poderosos e caros (BARROSO; DEAN; HOLZLE, 2003).

Por fim, as organizações cada vez mais realizam distribuição dos seus sistemas em escala global para atender as necessidades dos usuários de forma eficiente. Com isso, diversos data centers podem estar localizados em países diferentes, o que gera uma preocupação em relação a performance e disponibilidade das informações. Dessa forma, grandes empresas passaram a construir infraestruturas para dar suporte aos requisitos de suas aplicações e implementar softwares que sejam capazes de tolerar possíveis falhas (DE DIANA; GEROSA, 2010).

Assim, os bancos de dados não relacionais atuam sem um esquema predefinido, assim, é possível adicionar campos aos registros do banco de dados, sem ter a necessidade de definir quaisquer mudanças na estrutura (SADALAGE; FOWLER, 2019). Além disso, algumas características importantes diferenciam os bancos de dados não relacionais e garantem a capacidade de manipulação de grandes volumes de dados não estruturados e semiestruturados (OLIVEIRA, S. S. de, 2014):

- Escalabilidade: os bancos de dados não relacionais possuem a capacidade de crescimento do armazenamento de dados com o menor custo possível.
- Alta disponibilidade: os bancos de dados não relacionais se propõe a resistir a possíveis falhas, mantendo os serviços ativos o máximo de tempo viável.
- Esquema flexível: os bancos de dados não relacionais não possuem um esquema determinístico, ou seja, não há necessidade de se estabelecer uma estrutura formal dos dados.

2.4 MongoDB

O MongoDB é um banco de dados de documentos em que persistem documentos no formato BSON (Binary JSON). Em um documento, é possível ter dados de valores simples, como números, data e palavras, mas também pode conter uma lista de valores (HOWS; MEMBREY; PLUGGE, 2019). Com isso, os campos podem variar de documento para documento e a estrutura de dados pode ser modificada ao longo do tempo (MONGODB, 2022).

Os documentos são agrupados em coleções e um conjunto de coleções formam um banco de dados. Nas coleções são efetuadas operações de consulta (queries), essas consultas seguem o padrão de sintaxe JSON e são enviadas ao MongoDB como objetos BSON pelo driver de conexão ao banco. O MongoDB possibilita que os usuários realizem alterações de apenas um atributo no documento, ou seja, não é preciso de interação com o restante da estrutura (HOWS; MEMBREY; PLUGGE, 2019).

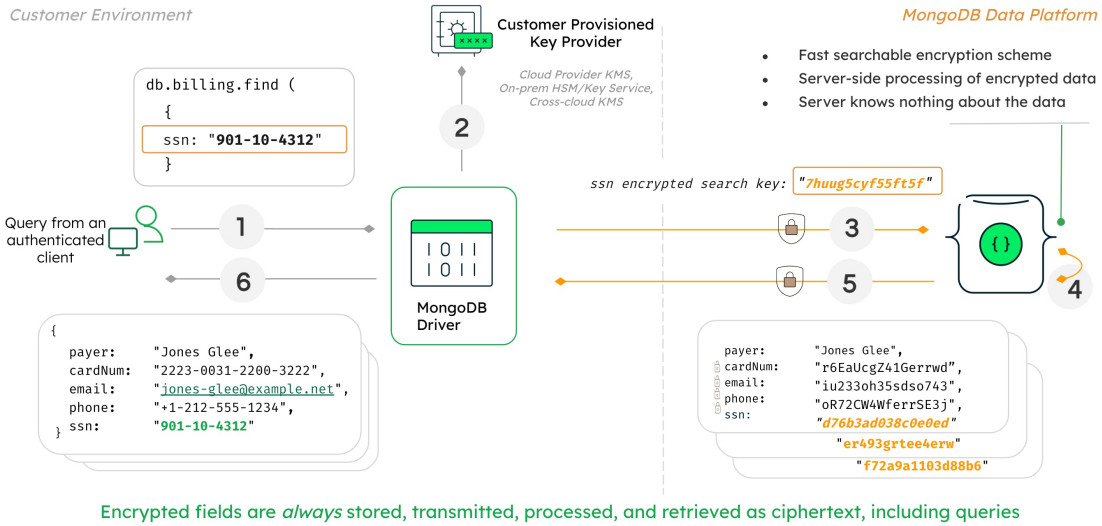
A arquitetura do MongoDB foi projetada com o objetivo de garantir uma alta disponibilidade através de replicação incorporada, uma escalabilidade horizontal, segurança de ponta a ponta no fluxo de informação e um banco de dados com capacidade de elasticidade (MONGODB, 2022).

Atualmente, o MongoDB está disponível na nuvem, como MongoDB Atlas, e também de forma on-premisse a partir das suas versões MongoDB Enterprise e MongoDB Community (MONGODB, 2022). O presente trabalho tem como foco o MongoDB Atlas. Ele é um serviço multi-cloud, ou seja, é possível executar aplicações que abrangem várias regiões ou nuvens ao mesmo tempo. Dessa forma, o MongoDB Atlas possibilita a implementação de cluster de banco de dados nas principais plataformas de computação em nuvem: AWS, Microsoft Azure e Google Cloud (MONGODB, 2022).

2.4.1 Queryable Encryption no MongoDB

O grupo Advanced Cryptography Research Group do MongoDB, desenvolveu um esquema próprio de aplicação da Queryable Encryption discutida anteriormente. A figura 2.3 apresenta um exemplo de como esse esquema é implementado.

Figura 2.3: Queryable Encryption aplicada no MongoDB



Fonte: (BRAUND; BORKAR, 2022)

No processo descrito na Figura 2.3 existem registros criptografados no banco de dados e o usuário autenticado deseja recuperar algumas informações. Na primeira etapa do fluxo, o usuário autenticado envia uma consulta para recuperar os registros que possuem o número SSN informado. Essa consulta é enviada aos drivers do MongoDB que fazem a análise da query solicitada.

Após reconhecer que a consulta é realizada em um campo criptografado, na segunda etapa do fluxo, o driver solicita as chaves de criptografia do provedor de chaves fornecido pelo cliente. O MongoDB suporta diversos provedores de chave como AWS Key Management Service, Google Cloud KMS, Azure Key Vault e também o provedor de chaves local.

Após a solicitação, na terceira etapa, o driver envia a consulta ao servidor MongoDB com o campo criptografado. Na quarta etapa do fluxo, o Queryable Encryption implementa o esquema pesquisável que permite que o servidor realize o processamento das consultas em dados totalmente criptografados, sem saber nada sobre os dados. Portanto, os dados e a própria consulta permanecem criptografados o tempo todo no servidor. Com isso, o servidor MongoDB retorna os resultados criptografados da consulta ao driver. Por fim, na sexta etapa do fluxo os resultados da consulta são descriptografados com as chaves mantidas pelo driver e retornados ao usuário como texto simples.

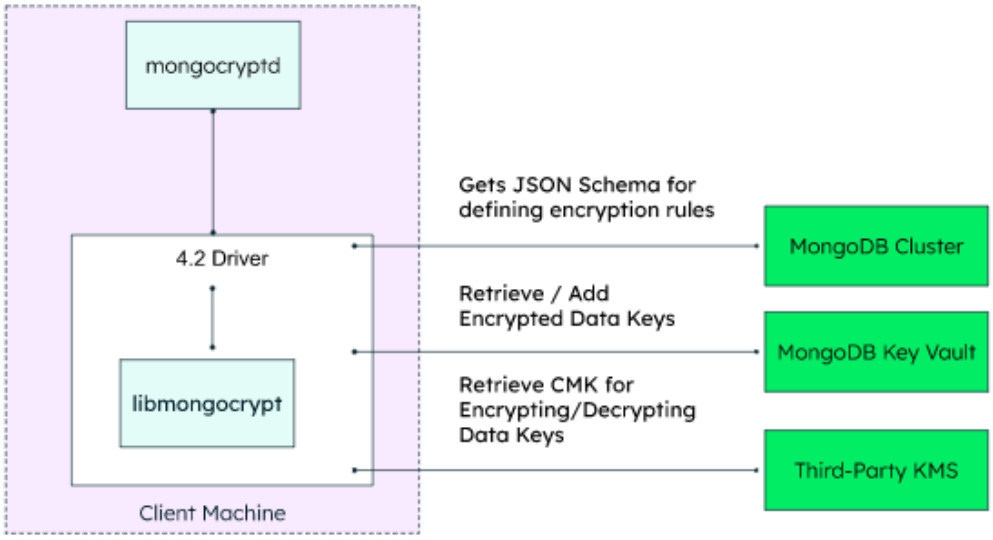
Percebe-se que a criptografia dos dados em uso, ocorre a partir do paradigma em que os dados são criptografados do lado do cliente antes de serem enviados pela rede para o servidor do MongoDB. Esse recurso é pontuado como Client-Side Field Level Encryption (CSFLE) (EMERICH, 2022).

2.4.1.1 Client-Side Field Level Encryption

A partir do CSFLE nenhum produto da stack MongoDB tem acesso aos dados do cliente de forma não criptografada (MONGODB, C., 2022). Portanto, para entender como esse recurso

funciona é importante apresentar os componentes que abrangem a sua arquitetura. Para isso, analisa-se a figura 2.4 a seguir.

Figura 2.4: Componentes do recurso CSFLE



Fonte: (MONGODB, EC, 2022)

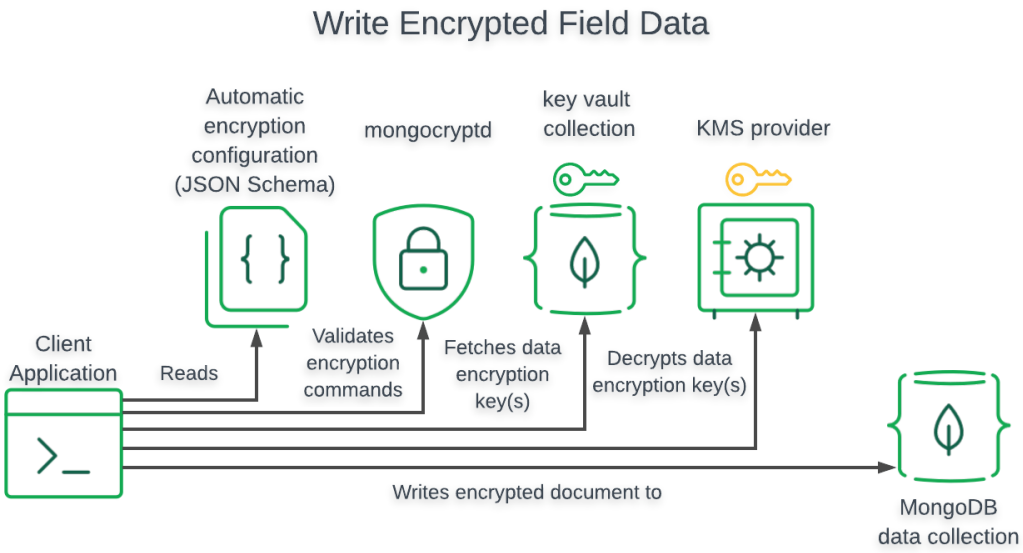
- libmongocrypt: biblioteca de criptografia de código aberto licenciada pelo Apache. Os principais drivers compatíveis com o MongoDB utilizam essa biblioteca para alimentar a criptografia em nível de campo do lado do cliente.
- mongocryptd: recurso responsável por suportar a criptografia automática. Esse recurso realiza etapas de validação de esquemas e operações a partir das regras de criptografia automática. Além disso, ele impede que operações sem suporte sejam executadas em campos criptografados.
- Key Vault collection: coleção padrão do MongoDB que armazena todas as chaves de criptografia de dados. As próprias chaves de criptografia de dados são criptografadas usando a chave mestra do cliente antes do armazenamento na Key Vault collection.
- Key Management System: sistema responsável por armazenar a chave mestra do cliente usada para criptografar as chaves de criptografia de dados.
- MongoDB Cluster: responsável por armazenar os dados criptografados e garantir a infraestrutura para a criptografia em nível de campo do lado do cliente.

Na implementação desse recurso a configuração pode ser realizada a partir de dois mecanismos diferentes (MONGODB, C., 2022): criptografia automática e criptografia explícita.

A **criptografia automática** permite que o usuário realize operações de leitura e gravação sem precisar desenvolver códigos para especificar como a criptografia deve ocorrer nos campos.

A figura 2.5 apresenta o diagrama que mostra as etapas realizadas pela aplicação do cliente e pelo driver para executar a gravação dos dados:

Figura 2.5: Gravação de dados a partir do mecanismo de criptografia automática

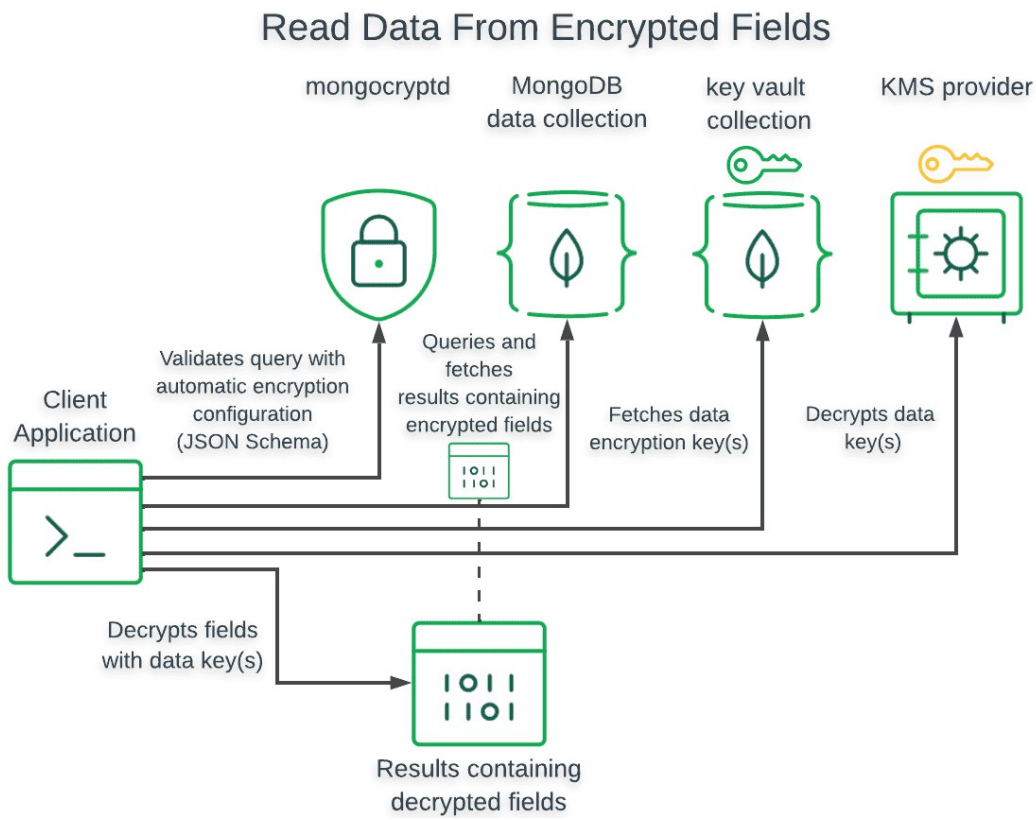


Fonte: (MONGODB, AE, 2022)

Para realizar a gravação dos dados, o fluxo inicia-se com a criação de um cliente MongoDB habilitado para CSFLE. Com isso, é possível realizar a leitura do esquema JSON que contém as configurações e regras da criptografia automática. O processo mongocryptd é inicializado e ele é responsável por analisar o esquema de criptografia especificado. Caso as regras de criptografia automática possuírem alguma sintaxe inválida, ele retorna um erro. Se o esquema for válido, o fluxo prossegue a partir da busca da chave de criptografia dos dados na coleção de chaves armazenada no MongoDB (MONGODB, AE, 2022). Por fim, recupera-se a chave mestra no provedor de chaves do usuário para habilitar a gravação dos dados no banco.

Para as operações de leitura de dados, a figura 2.6 apresenta o diagrama com as etapas realizadas pela aplicação do cliente e pelo driver:

Figura 2.6: Leitura de dados a partir do mecanismo de criptografia automática



Fonte: (MONGODB, AE, 2022)

Para esse tipo de operação, o driver criptografa os valores dos campos contidos na consulta antes de enviar ao servidor do MongoDB a operação de leitura. Após o retorno da consulta, o driver descriptografa automaticamente os valores criptografados somente se ele foi configurado com acesso à chave mestra do cliente e às chaves de criptografia de dados usadas para criptografar esses valores (MONGODB, AE, 2022).

O segundo tipo de mecanismo para implementação da CSFLE é a **criptografia explícita**. Ela permite que o usuário especifique a forma que os campos do documento devem ser criptografados e descriptografados em cada operação realizada no banco de dados (MONGODB, EE, 2022). A criptografia explícita requer uma especificação lógica padrão utilizando a biblioteca *libmongocrypt* do drive MongoDB. Com isso, requisitos como uma chave e um algoritmo para cada campo que será criptografado do documento devem ser designados pelo usuário.

Os algoritmos que podem ser especificados neste mecanismo são algoritmos para criptografia autenticada com dados associados (AEAD), baseados na composição do Advanced Encryption Standard (AES) com o modo de operação Cipher Block Chaining (CBC) para criptografia e com o código de autenticação de mensagem HMAC-SHA (MAC) (MCGREW; FOLEY; PATERSON, 2014).

2.4.1.2 Tipos de operações suportadas

Atualmente, o esquema de Queryable Encryption proposto pelo MongoDB suporta dois tipos de consulta: *equality* e *none* (MONGODB, F., 2022).

Ao definir o tipo de consulta como *none* indica-se que os dados serão criptografados, mas não devem ser consultados. Portanto, as consultas não podem ser executadas em dados criptografados com um tipo de consulta *none* (MONGODB, F., 2022).

Já o tipo de consulta *equality*, que pode ser traduzido como igualdade, permite a consulta em dados criptografados, porém estas consultas devem estar associadas aos seguintes operadores (MONGODB, F., 2022):

- eq: O operador seleciona os documentos em que o valor de um campo é igual ao valor especificado na consulta.
- ne: O operador seleciona os documentos em que o valor de um campo não é igual ao valor especificado na consulta.
- in: O operador seleciona os documentos em que o valor de um campo corresponde a qualquer valor especificado no array.
- nin: O operador seleciona os documentos em que o valor de um campo não corresponde aos valores especificados no array.
- and: O operador executa uma operação lógica do tipo AND.
- or: O operador executa uma operação lógica do tipo OR.
- not: O operador executa uma operação lógica do tipo NOT.
- nor: O operador executa uma operação lógica do tipo NOR.
- expr: O operador permite expressões de agregação.

Capítulo 3

Discussão do problema e Proposta de Solução

Tendo em vista o problema de privacidade de dados em uso, principalmente, no que concerne à computação em nuvem, o presente trabalho propõe uma solução. Esta solução utiliza a técnica de Queryable Encryption para viabilizar consultas em dados criptografados que estão armazenados no banco de dados MongoDB Atlas. Assim, o objetivo desta proposta é dispôr uma arquitetura em que apenas o cliente, proprietário dos dados, possa ter acesso à informação contida nos documentos.

3.1 Discussão do Problema

Com o advento dos modelos de computação na nuvem, diversos conceitos inovadores surgiram. Portanto, foi possível transformar a computação em um serviço contratado de forma simples pelos clientes de acordo com as suas necessidades. Assim, o modelo de negócio atribuído a esses sistemas oferece a possibilidade de aumentar ou reduzir o nível de utilização dos recursos computacionais de maneira simples, sem a necessidade de manutenção constante do sistema físico (ARMBRUST et al., 2010).

Além disso, a computação em nuvem proporciona diversas vantagens em relação à capacidade de processamento e armazenamento. Ambos os pontos são relevantes para os cenários de grandes volumes de dados promovidos pela crescente massa de informação. Nesse sentido, os clientes confiam os seus dados ao fornecedor do serviço de nuvem e, em alguns casos, as informações trafegadas nas redes possuem dados pessoais que devem ser protegidos para garantir a sua privacidade (BOST, 2016).

A tecnologia de virtualização foi primordial para possibilitar o desenvolvimento do modelo de computação em nuvem, permitindo alocar recursos simultaneamente à vários clientes e redistribuí-los de acordo com a necessidade (ARMBRUST et al., 2010). Com isso, surgem os desafios de segurança nas tecnologias de sistemas distribuídos em nuvem. Por conceituar a nuvem como um conjunto de informações provenientes de diferentes clientes, ela pode se tornar um alvo vantajoso

a ataques por possíveis invasores, e como consequência, os requisitos de segurança da informação podem ser afetados (DIAS; RITA DE CÁSSIA; PIRES, 2012).

Assim, a partir da terceirização do armazenamento de dados aos provedores de nuvem e a importância de manter a informação segura durante todo o seu ciclo de vida, se faz necessário a promoção de tecnologias que garantam tal feito. Uma das soluções previstas é cifrar as informações e armazená-las neste formato, garantindo assim a segurança do dado em repouso. Além disso, para que o dado não seja exposto enquanto estiver em uso, as operações computacionais são realizadas sem a necessidade de descriptografar as informações. Com isso, nenhum dado é revelado ao servidor.

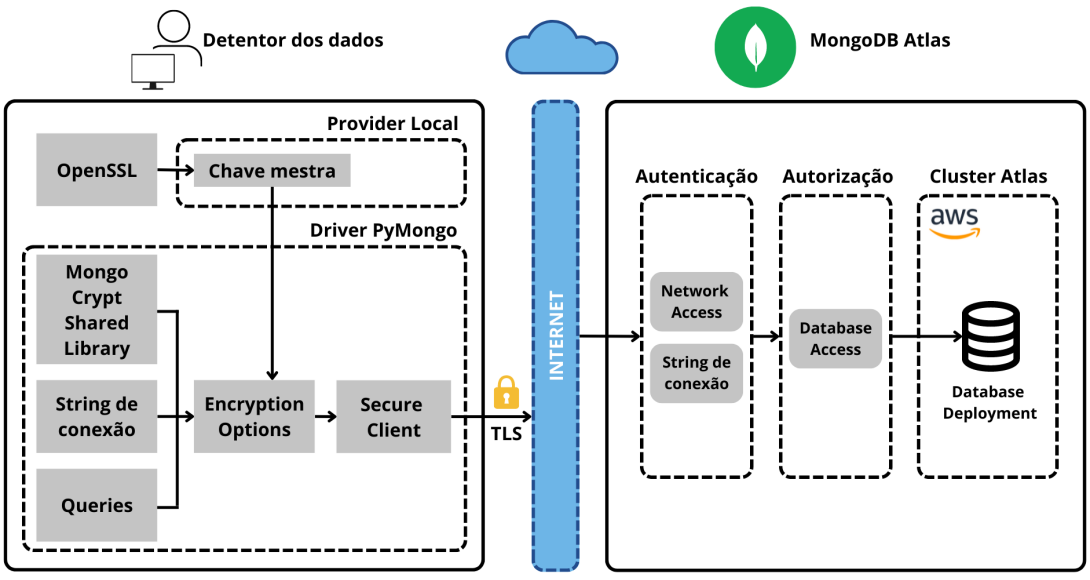
Apesar da computação em nuvem provisionar recursos que garantam uma alta capacidade de poder computacional que estas soluções demandam, é necessário analisar a eficiência desse sistema. O modelo de cobrança da computação em nuvem decorre em função do acesso ao dado, dessa forma quanto maior for o tamanho do dado envolvido no processo, espera-se um custo proporcional. Além disso, para grandes volumes de dados cujo acesso aos documentos seja frequente, é importante validar o impacto das ações de processamento.

3.2 Proposta de Solução

3.2.1 Arquitetura

A arquitetura utilizada para implementar a solução proposta pode ser verificada na Figura 3.1.

Figura 3.1: Arquitetura utilizada



Fonte: elaborado pelo autor (2022).

3.2.1.1 Detentor dos dados

Nota-se na Figura 3.1 que existem recursos essenciais no lado do detentor dos dados responsáveis por garantir o Secure Client. O Secure Client é um cliente que atende os requisitos de segurança e é autenticado para realizar consultas em dados criptografados que estão armazenados no cluster Atlas.

Como prova de conceito, o provedor de chaves local foi utilizado. Desse modo, a chave mestra do cliente, que executa o esquema de Queryable Encryption, está armazenada localmente. Esta chave possui 96 bytes e é gerada a partir da biblioteca OpenSSL.

Além disso, é necessário que a aplicação do cliente possua um driver MongoDB compatível com a Queryable Encryption. Assim, para o presente trabalho, o driver escolhido foi o *PyMongo*. A partir deste driver e das credenciais do usuário de banco de dados criado para acessar o cluster Atlas, é possível gerar a string de conexão. Ela é responsável por garantir a autenticação do cliente ao realizar a conexão com o banco de dados MongoDB.

O Secure Client também precisa receber como parâmetro a biblioteca compartilhada de criptografia automática. Ela permite que este usuário possa executar o esquema de Queryable Encryption de forma automática e descriptografar os resultados das consultas realizadas. Por fim, com todos os recursos provisionados e as devidas opções de criptografia configuradas, cria-se um cliente capaz de executar as queries nos dados criptografados.

3.2.1.2 Internet

O MongoDB Atlas oferece suporte ao *Transport Layer Security* (TLS) e ao *Secure Sockets Layer* (SSL) para comunicações entre o cliente e o servidor, definindo os certificados. Um certificado é um documento eletrônico usado para provar a validade da propriedade da chave privada. Os certificados garantem que os dados sejam criptografados para transporte em uma conexão de rede confiável, evitando atividades de espionagem, como detecção de pacotes ou falsificação de IP/DNS (EMERICH, 2022). Assim, o driver PyMongo suporta a conexão com o MongoDB via TLS/SSL.

3.2.1.3 MongoDB Atlas

O MongoDB Atlas é um serviço de banco de dados NoSQL em nuvem totalmente gerenciado, desenvolvido pela equipe oficial do MongoDB. Os recursos utilizados no MongoDB Atlas, como demonstrado na Figura 3.1, são:

- Recursos de segurança atribuídos às etapas de autenticação e autorização;
- Cluster Atlas provisionado na AWS;
- Banco de dados hospedado no cluster Atlas.

Para acessar o cluster Atlas em que o banco de dados está hospedado é necessário que o cliente atenda aos requisitos de segurança. Portanto, realiza-se as etapas de autenticação e autorização.

A autenticação compreende a etapa em que é verificado se o endereço IP solicitante da requisição está registrado na lista de endereços IPs confiáveis no recurso *Network Access*. Um IP, Internet Protocol, é um protocolo de comunicação da camada de rede que endereça os dispositivos conectados à ela. No Atlas, só é possível se conectar a um cluster e acessar os dados a partir de um endereço IP confiável. Além disso, realiza-se também a autenticação da string de conexão passada pelo cliente no momento da requisição a fim de garantir que as credenciais de acesso ao banco de dados são válidas.

A etapa de autorização é realizada a partir do recurso de *Database Access*. O MongoDB concede acesso aos dados por meio de autorização baseada em funções. Desse modo, uma função concede privilégios a um usuário de banco de dados para que ele execute um conjunto de ações em um determinado recurso. Por exemplo, caso um usuário de banco de dados tenha uma função *Read and write to any database* atribuída, significa que ele possui autorização para leitura e escrita dos dados em todas as bases do cluster.

Para o presente trabalho o MongoDB teve os seus *workloads* hospedados na AWS. Assim, foram automatizadas as tarefas de administração demoradas, tais como provisionamento de hardware, patch de software, atualizações e backups. Finalmente, o cluster Atlas foi criado na AWS, assim como os bancos de dados.

3.2.2 Implementação

3.2.2.1 Ferramentas utilizadas

Para a implementação do sistema proposto, foi necessário configurar os seguintes itens no ambiente de desenvolvimento:

Biblioteca compartilhada de criptografia automática: A biblioteca compartilhada de criptografia automática é uma biblioteca dinâmica que permite que o aplicativo do lado do cliente execute o esquema de Queryable Encryption de forma automática. Uma biblioteca dinâmica é um conjunto de funcionalidades acessadas por um aplicativo em tempo de execução, em vez de em tempo de compilação. A Biblioteca Compartilhada de Criptografia Automática é a *Mongo Crypt Shared Library* e exerce o papel do recurso *mongocryptd*, discutido na seção 2.4.1.1 deste trabalho. Portanto, sabe-se que ela é responsável por realizar a validação dos esquemas de mapeamento dos campos criptografados e impede que o aplicativo execute operações que não são suportadas em campos criptografados (MONGODB, AESL, 2022).

PyMongo: O ambiente de desenvolvimento necessita de um driver do MongoDB compatível com a Queryable Encryption. Para o presente estudo, o driver escolhido foi o *PyMongo* na versão 4.2.0. O *PyMongo* é uma distribuição python que contém ferramentas para trabalhar com MongoDB e a biblioteca de criptografia utilizada por esse driver é a *pymongocrypt* (PYMONGO, 2008).

Cluster Atlas: O MongoDB Atlas fornece uma maneira acessível de hospedar e gerenciar os dados na nuvem. Para o sistema proposto foi necessário criar um cluster Atlas em uma versão com

suporte a Queryable Encryption com criptografia automática. Segundo a documentação do MongoDB essa funcionalidade é suportada por clusters na versão 6.0 ou posterior (MONGODB, AESL, 2022). O MongoDB oferece o provisionamento de clusters gratuitos, eles possuem funcionalidades básicas para projetos iniciantes na plataforma. Porém, estes clusters não atendem o requisito de versão com suporte ao Queryable Encryption com criptografia automática.

Assim, foi provisionado um cluster dedicado, na versão 6.0. Este tipo de cluster oferece controles de configuração avançados, isolamento de rede e criptografia de ponta a ponta. O MongoDB Atlas é um serviço *multi-cloud*, portanto, disponibiliza opções de hospedagem em diferentes provedores como AWS, Microsoft Azure e Google Cloud.

Para o presente projeto, o provedor de nuvem escolhido para hospedar o cluster foi o da AWS. Além disso, a zona de disponibilidade, ou seja, a zona que designa a localização física em que o cluster está instanciado, é Oregon, indicado pela sigla us-west-2. Apesar da possibilidade de hospedar os serviços em São Paulo, o custo operacional é superior comparado a região de Oregon, nos Estados Unidos. Assim, por se tratar de um processo experimental, a escolha da zona de disponibilidade levou em conta a região com menor custo.

O cluster foi configurado na instância M10, que categoriza clusters dedicados para ambientes de desenvolvimento e aplicativos de baixo tráfego. Por fim, definiu-se os seguintes parâmetros de configuração: 2 GB de memória RAM, 10 GB de armazenamento disponível e 2 vCPUs.

3.2.2.2 Conjunto de dados

Como prova de conceito, foi utilizado o dataset seguindo algumas recomendações da Polícia Civil do Distrito Federal quanto ao registro de ocorrências de natureza da Lei nº 11.340, sancionada em 7 de agosto de 2006, intitulada Lei Maria da Penha (PCDF, 2022). Assim, este dataset possui dados pessoais e, segundo o que preconiza a LGPD, Lei nº 13.709, sancionada em 14 de agosto de 2018, o uso de dados pessoais precisam estar protegidos (BRASIL, 2018).

O conjunto de dados possui informações do envolvido que registra a ocorrência como mostrado na Tabela 3.1 e informações relacionadas a descrição do ocorrido, como pode ser visto na Tabela 3.2.

3.2.2.3 Criação das chaves de criptografia

Para realizar a leitura e a inserção dos dados criptografados no banco de dados, é necessário realizar a criação das chaves de criptografia. Estas chaves consistem tanto na chave mestra do cliente quanto nas chaves de criptografia dos dados. A Figura 3.2 apresenta o processo que garante esta etapa.

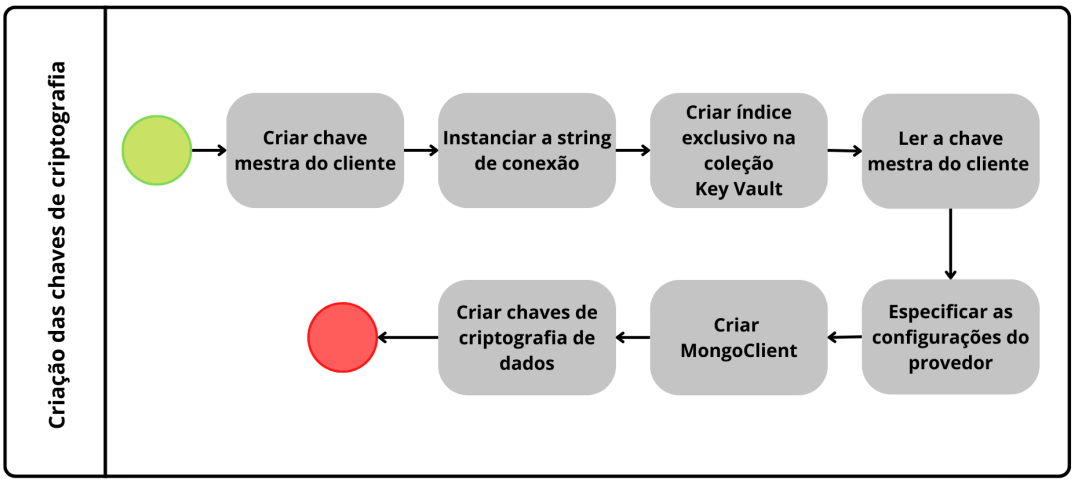
Tabela 3.1: Dados do envolvido

Dado	**Descrição**	**Tipo do dado**
is_vitima	Indica *true* se o envolvido é a vítima e *false* se o envolvido não é a vítima do ocorrido	Booleano
tipo_envolvido	Identifica se o envolvido é a vítima, autor ou testemunha do ocorrido	Texto
nome	Nome do envolvido	Texto
sobrenome	Sobrenome do envolvido	Texto
nascimento	Data de nascimento do envolvido	Data
cpf	Número do CPF do envolvido	Número
nome_mae	Nome da mãe do envolvido	Texto
nome_pai	Nome do pai do envolvido	Texto
endereco	Endereço do envolvido	Texto
UF	Estado referente ao endereço do envolvido	Texto
estado_civil	Estado civil do envolvido	Texto
grau_instrucao	Grau de instrução do envolvido	Texto
raca_cor	Raça que o envolvido se identifica	Texto
genero	Gênero que o envolvido se identifica	Texto

Tabela 3.2: Dados do ocorrido

Dado	**Descrição**	**Tipo do dado**
data_ocorrido	Data que o ocorrido aconteceu	Texto
local	Local que o ocorrido aconteceu	Texto
meio_empregado	Caracteriza o tipo de violência do ocorrido	Texto
descricao_fato	Descreve em texto livre o ocorrido	Texto

Figura 3.2: Processo de criação de chaves



Fonte: elaborado pelo autor (2022).

O processo se inicia a partir da criação da chave mestra do cliente (CMK). Como visto anteriormente, a chave mestra é criada a partir da biblioteca OpenSSL e possui 96 bytes. Como é

utilizado um provedor de chaves local, ela é salva no sistema de arquivos, como *master-key.txt*.

Posteriormente, é necessário instanciar a string de conexão que foi gerada a partir do driver *PyMongo*. Com isso, é possível realizar a criação de um índice exclusivo na coleção Key Vault. O objetivo desta coleção é ser um cofre de chaves, isto é, ela armazena as chaves de criptografia de dados. Além disso, o usuário de banco de dados utilizado para se conectar ao MongoDB deve possuir permissão para realizar leitura e escrita na coleção Key Vault.

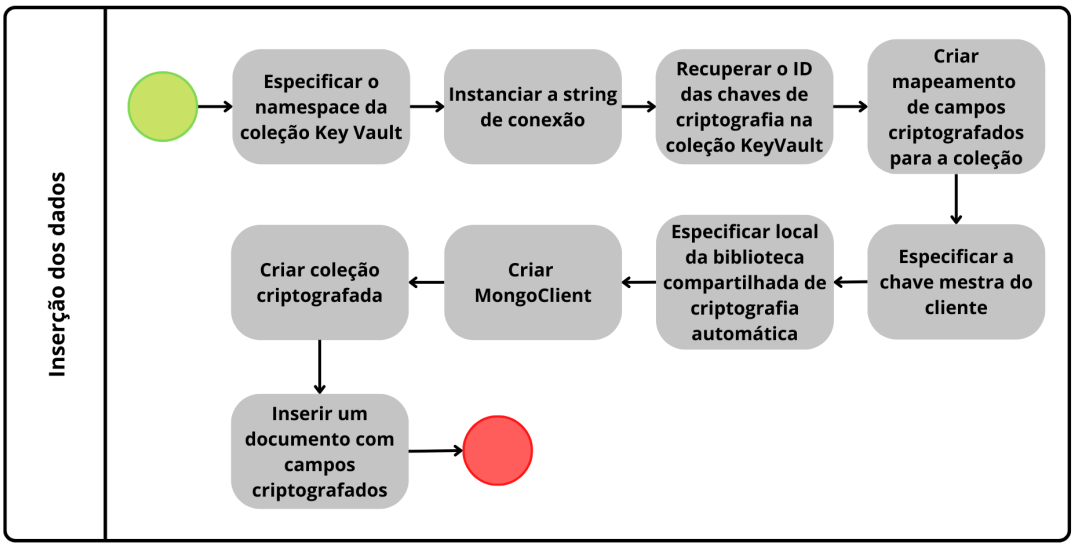
Assim, com o índice implementado na coleção Key Vault, cria-se as chaves de criptografia que serão armazenadas no cofre. Para isso, recupera-se o conteúdo do arquivo da chave mestra do cliente e especifica-se a configuração do provedor de chave local.

Por fim, cria-se um cliente a partir da string de conexão do MongoDB, do namespace da coleção Key Vault e da chave mestra. Este cliente criará as chaves de criptografia de dados. É importante identificar quantos atributos do dataset serão armazenados de maneira criptografada, pois cada atributo deve ter uma chave de criptografia associada.

3.2.2.4 Inserção dos dados

O processo de inserção dos dados está descrito na Figura 3.3.

Figura 3.3: Processo de inserção dos dados



Fonte: elaborado pelo autor (2022).

Para inserir os atributos do dataset de forma criptografada no banco de dados, é necessário recuperar o identificador único (ID) de cada chave de criptografia de dados gerada no processo detalhado na seção 3.2.2.3. Para isso, instancia-se a string de conexão e especifica-se o namespace da coleção Key Vault. Assim, o cliente autenticado realiza a recuperação dos identificadores únicos atribuídos às chaves de criptografia.

Cada chave de criptografia é associada a um campo diferente. Portanto, realiza-se um mapeamento de campos criptografados para a coleção. Este mapeamento é um esquema JSON que configura a Queryable Encryption para criptografar automaticamente esses campos. Assim, são especificados quatro parâmetros principais como mostra a Tabela 3.3.

Tabela 3.3: Parâmetros utilizados para o mapeamento de campos criptografados

Parâmetro	**Descrição**
keyId	Id da chave de criptografia de dados exclusiva
path	Nome do campo a ser criptografado
bsonType	Tipo do dado
queries	Especifica se o campo será consultável ou não

O código apresentado no Anexo I exemplifica um dos mapeamentos de campos criptografados utilizado no presente projeto. Este código foi utilizado para um dos cenários de implementação que será discutido na seção 3.2.2.6.

Após o desenvolvimento do mapeamento para os campos criptografados, inicia-se a etapa de configuração dos requisitos necessários para autenticar o cliente. Este cliente é responsável por criar a coleção criptografada e a inserção dos dados na mesma. Assim, são especificados a chave mestra do cliente e o local em que a biblioteca compartilhada de criptografia automática está instalada na máquina. Portanto, o MongoClient é configurado para suportar a Queryable Encryption automática a partir dos seguintes requisitos:

- Key Vault namespace;
- String de conexão;
- Chave mestra do cliente;
- Mapeamento de campos criptografados;
- Biblioteca compartilhada de criptografia automática.

Durante a implementação, foi criada a classe CRUD. Esta classe é responsável por garantir operações básicas como criação da coleção, inserção, consulta e exclusão dos dados. Portanto, ela possui os seguintes métodos:

1. Create collection
2. Insert
3. Drop database
4. Find one decrypted
5. Find all decrypted
6. Find one encrypted

7. Find all encrypted

8. Count documents

Os métodos 1, 2 e 3 serão detalhados nesta seção. Eles são implementados a partir do cliente configurado para a Queryable Encryption automática. Dessa forma, as operações só são realizadas se o mesmo estiver configurado de acordo com os parâmetros detalhados anteriormente.

O primeiro método, realiza a criação da coleção criptografada. Assim, o cliente autenticado cria, de acordo com o mapeamento, a coleção no banco de dados informado. Já o segundo método, permite a inserção do documento JSON na coleção criptografada criada. Logo, garante-se que os campos sejam previamente criptografados antes de serem inseridos na base de dados.

O terceiro método, realiza a exclusão das bases de dados. Por fim, os métodos 4, 5, 6, 7 e 8 realizam as consultas na coleção. Eles serão discutidos com mais detalhe a partir da seção 3.2.2.5.

O código completo está presente no Anexo II.

3.2.2.5 Leitura dos dados

A funcionalidade da Queryable Encryption é, na sua finalidade, permitir que consultas sejam realizadas em documentos criptografados. Nesta seção será detalhado os métodos para consultas a partir de um cliente configurado para a Queryable Encryption automática, bem como para um cliente que não está configurado para a Queryable Encryption automática.

Assim, de acordo com o parâmetro *queries* especificado no mapeamento de campos, o usuário poderá realizar consultas nos campos habilitados, respeitando as operações suportadas pela Queryable Encryption (seção 2.4.1.2). Na classe CRUD foram criados alguns métodos responsáveis por realizar tais consultas. Eles possuem algumas diferenças que devem ser analisadas com mais detalhe.

O método 4 e 5 se diferenciam em relação à quantidade de registros apresentados no resultado da consulta. Enquanto o método 4 retorna ao usuário um único registro que atenda a query especificada, o método 5 retorna todos os registros que atendam a consulta. Além disso, para ambos os métodos, todos os campos dos registros são visualizados de forma descriptografada independente de como a informação foi armazenada. Isto só é possível pois, a consulta no documento é realizada a partir de um cliente configurado para a Queryable Encryption automática.

O método 6 e 7 também se diferenciam em relação à quantidade de registros que são apresentados como resultado. Porém, o cliente que realiza a consulta não está configurado para a Queryable Encryption automática e, portanto, todos os atributos da base de dados que foram armazenados de maneira criptografada permanecem dessa forma ao serem apresentados.

Por fim, o método 8 realiza a operação de contagem dos registros que corresponderem à consulta. Assim, este método não retorna as informações do registro em si, mas apenas o valor da quantidade de registros correspondentes.

3.2.2.6 Cenários de implementação

O dataset descrito na seção 3.2.2.2 é inserido na *cloud* a partir de três cenários diferentes que viabilizam a avaliação do modelo de Searchable Encryption. Dessa forma, são implementados três bancos de dados para acomodar cada cenário:

- Banco de dados sem criptografia;
- Banco de dados parcialmente criptografado;
- Banco de dados totalmente criptografado.

No banco de dados sem criptografia todas as informações contidas no documento são inseridas como texto simples. Assim, qualquer usuário autenticado ao banco de dados pode ter acesso ao conteúdo do dado por este estar totalmente legível.

No cenário em que o banco de dados está parcialmente criptografado, algumas informações do conjunto de dados são inseridas de forma criptografada e as demais de forma legível. Com isso, foram selecionados sete dados do dataset para serem inseridos de forma criptografada no banco de dados, são eles: **is_vitima**, **data_ocorrido**, **cpf**, **UF**, **endereco**, **nascimento**, **grau_instrucao** e **raca_cor**.

Já no cenário em que o banco de dados está totalmente criptografado, nenhum dado será inserido de forma legível. Assim, todo o conjunto de dados será armazenado de forma criptografada.

Para os cenários parcialmente criptografado e totalmente criptografado, por comportarem dados cifrados, se faz necessário criar um banco de dados em paralelo que comportará a coleção Key Vault. Esta coleção é responsável por armazenar as chaves de criptografia de cada dado, como visto na seção 3.2.2.3.

Capítulo 4

Análise de resultados

Neste capítulo serão apresentados os principais resultados da implementação da Queryable Encryption no MongoDB, bem como as pesquisas de desempenho baseadas nos três cenários discutidos na seção 3.2.2.6.

4.1 Bancos de dados

Para contemplar aspectos experimentais, o conjunto de dados utilizado possui no total 500 registros. Assim, de acordo com os três cenários de implementação foi necessário criar distintos bancos de dados para armazenar as informações. As bases criadas serão discutidas a seguir.

4.1.1 Banco de dados sem criptografia

No banco de dados sem criptografia o conjunto de dados foi armazenado com todos os registros em formato de texto legível. Dessa forma, o processo de ingestão dos dados foi realizado de forma direta, sem a necessidade da criação de chaves de criptografia e mapeamento de campos criptografados.

O banco de dados criado para este cenário foi nomeado de **OcorrenciasMariaDaPenha** e a coleção correspondente nomeada de **Ocorrencias**. A Figura 4.1 apresenta o esquemático de um registro armazenado nesta base de dados.

Figura 4.1: Exemplo de registro armazenado no banco de dados **OcorrenciasMariaDaPenha**

Fonte: elaborado pelo autor (2022).

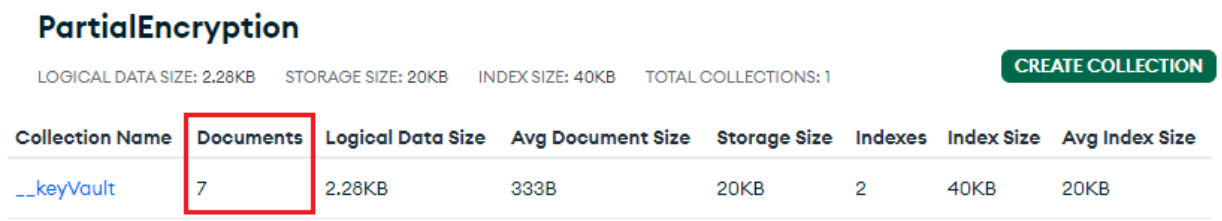
4.1.2 Banco de dados parcialmente criptografado

No banco de dados parcialmente criptografado, cada registro do conjunto de dados foi armazenado com apenas alguns campos criptografados. Nesse sentido, foi necessário criar um banco de dados para armazenar as chaves de criptografia de cada um dos campos que serão cifrados. O Anexo I apresenta o mapeamento de campos criptografados desenvolvido para este cenário. De acordo com este anexo, sete atributos de cada registro do conjunto de dados são mapeados para o armazenamento de forma criptografada.

O banco de dados criado para este cenário foi nomeado de **OcorrenciasMariaDaPenhaParcialmenteCriptografadas** e a coleção correspondente de **OcorrenciasParcialmenteCriptografadas**. Já o banco de dados criado para armazenar as chaves de criptografia foi nomeado de **PartialEncryption** e a coleção de **__keyVault**.

A partir da Figura 4.2, verifica-se algumas informações referentes à coleção **__keyVault**. Destaca-se nesta figura a quantidade de documentos armazenados, equivalente a sete, que corresponde ao número de chaves de criptografia utilizadas.

Figura 4.2: Quantidade de documentos armazenados na coleção Key Vault para o cenário parcialmente criptografado



Fonte: elaborado pelo autor (2022).

A Figura 4.3 apresenta o esquemático de um registro que é armazenado no banco de dados parcialmente criptografado. Percebe-se que as informações que foram parametrizadas no mapeamento de campos criptografados são armazenadas de forma ilegível, sendo representadas por "*********".

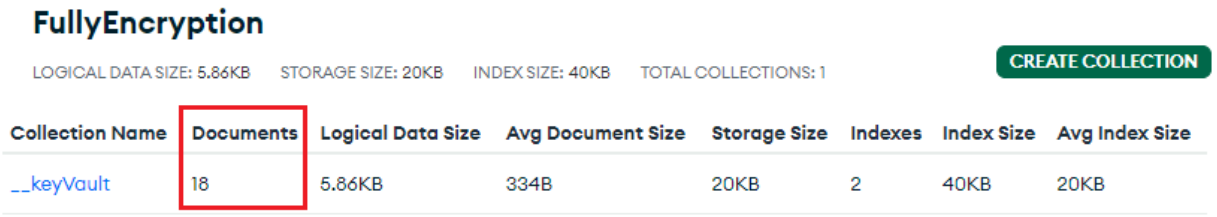
Figura 4.3: Exemplo de registro armazenado no banco de dados **OcorrenciasMariaDaPenhaParcialmenteCriptografadas**

Fonte: elaborado pelo autor (2022).

4.1.3 Banco de dados totalmente criptografado

O último cenário de implementação tem como objetivo armazenar todos os atributos dos registros de forma criptografada. Assim, é possível garantir que nenhuma informação no banco de dados esteja disponível de forma legível. Nesse sentido, foi necessário criar dezoito chaves de criptografia para os campos e estas foram armazenadas na coleção **__keyVault** do banco de dados **FullyEncryption**. A Figura 4.4 apresenta em destaque a quantidade de documentos existentes nesta coleção.

Figura 4.4: Quantidade de documentos armazenados na coleção Key Vault para o cenário totalmente criptografado



Fonte: elaborado pelo autor (2022).

Por fim, a Figura 4.5 apresenta o esquemático de um registro armazenado na base de dados criada para este cenário. O banco de dados foi nomeado de **OcorrenciasMariaDaPenhaTotalmenteCriptografadas** e a coleção correspondente de **OcorrenciasTotalmenteCriptografadas**. Nota-se que, como esperado, todos os atributos estão dispostos de maneira ilegível.

Figura 4.5: Exemplo de registro armazenado no banco de dados **OcorrenciasMariaDaPenhaTotalmenteCriptografadas**

Fonte: elaborado pelo autor (2022).

4.2 Pesquisa de desempenho

Para avaliar os diferentes cenários de implementação em uma perspectiva de análise de desempenho, foram adotadas três consultas para serem executadas nos bancos de dados. Estas consultas foram niveladas da seguinte forma:

- Consulta simples — executada a partir de um único atributo do conjunto de dados;
- Consulta intermediária — executada a partir de diferentes atributos do conjunto de dados e que utiliza operações lógicas;
- Consulta complexa — executada a partir de operações que demandem processamento realizado pelo banco.

4.2.1 Consultas utilizadas

As consultas utilizadas e o grau de complexidade estão detalhadas na Tabela 4.1. Além disso, a explicação para cada consulta está apresentada a seguir.

Tabela 4.1: Consultas utilizadas para pesquisa de desempenho

Consulta	**Nível de complexidade**	**Atributos**	**Resultado**
1) Quantas ocorrências registradas são do Distrito Federal?	Simples	UF	24
2) Quantas vítimas possuem a raça negra, preta ou indígena?	Intermediária	is_vitima e raca_cor	124
3) Quantas vítimas possuem menos de 10 anos e sofreram violência sexual?	Complexa	is_vitima, nascimento e meio_empregado	0

4.2.1.1 Quantas ocorrências registradas são do Distrito Federal?

Para executar esta consulta utiliza-se o método *count documents* da classe CRUD discutido na seção 3.2.2.4. O código desenvolvido pode ser visualizado a partir do Código 4.1.

Código 4.1: Consulta 1

```
CRUD.count_documents({"envolvido.UF": {"$eq": "DF"}}, encrypted_coll)
```

Com isso, é retornada a quantidade de ocorrências do Distrito Federal. A Tabela 4.1 apresenta o resultado desta consulta.

4.2.1.2 Quantas vítimas possuem a raça negra, preta ou indígena?

Esta consulta também utiliza o método *count documents* da classe CRUD. Além disso, ela é executada a partir de operações lógicas AND e OR. Para a operação AND, foi definido dois parâmetros: um parâmetro referente a *raca_cor* e outro parâmetro referente a *is_vitima*. Já para a operação OR, foi utilizado o parâmetro *raca_cor*, verificando se a pessoa é negra, preta ou indígena. Assim, utiliza-se a query visualizada no Código 4.2.

Código 4.2: Consulta 2

```
CRUD.count_documents({
    "$and": [
        {
            "$or": [
                {"envolvido.raca_cor": "Negra"},
                {"envolvido.raca_cor": "Preta"},
                {"envolvido.raca_cor": "Indigena"},
            ]
        },
        {
            "envolvido.is_vitima": {"$eq": True}
        }
    ]
},encrypted_coll)
```

4.2.1.3 Quantas vítimas possuem menos de 10 anos e sofreram violência sexual?

As operações permitidas para os campos criptografados estão detalhadas na seção 2.4.1.2. Com isso, nota-se que não é possível realizar determinadas consultas sem um nível de granularidade e complexidade de código, por exemplo as operações de comparação maior (>) e menor (<).

Logo, para definir a idade da vítima a partir da sua data de nascimento, foi necessário realizar a seguinte abordagem:

- Cria-se, inicialmente, um método responsável por gerar um *array* com todas as datas de nascimento possíveis para qualquer envolvido que tenha atualmente menos de 10 anos;
- A partir disso, realiza-se uma consulta em que a data de nascimento do envolvido pertença ao array gerado;
- Por fim, verifica-se as outras duas condicionais que o envolvido necessariamente precisa contemplar: *is_vitima* igual a "True" e *meio_empregado* igual a "Violência Sexual".

Neste contexto, o Código 4.3 demonstra como foi aplicado essa abordagem. No Anexo III também é possível visualizar o método *generate_dates* utilizado para contornar o problema da falta de suporte às operações de comparação maior (>) e menor (<).

Código 4.3: Consulta 3

```
CRUD.count_documents({
    "$and": [
        {
            "envolvido.nascimento": {"$in": DatesInputs.generate_dates(10)}
        },
        {
            "envolvido.is_vitima": {"$eq": True}
        },
        {
            "meio_empregado": {"$eq": "Violencia Sexual"}
        }
    ]
},encrypted_coll)
```

4.2.2 Tamanho do armazenamento

As Figuras 4.6, 4.7 e 4.8 apresentam algumas informações relacionadas às coleções dos bancos de dados criados para cada cenário de implementação. Todas as bases possuem a mesma quantidade de documentos armazenados, como pode ser visto nas figuras através da informação *Documents*. Neste caso, os bancos de dados possuem 500 documentos no total. No entanto, analisando o *Storage Size* percebe-se que cada base possui um tamanho diferente de armazenamento. A Tabela 4.2 consolida esta informação.

Tabela 4.2: Tamanho do armazenamento dos bancos de dados

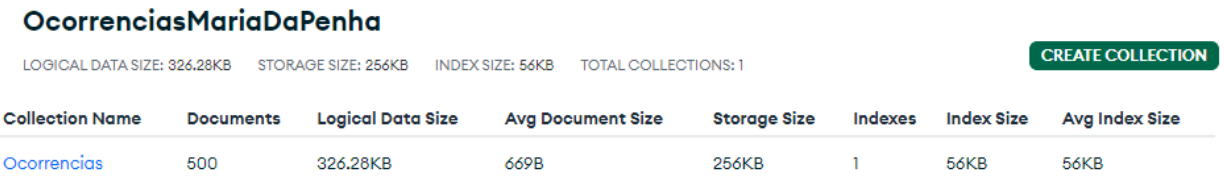
Cenário	**Coleção**	**Tamanho do armazenamento**
Não criptografado	Ocorrencias	256 KB
Parcialmente criptografado	OcorrenciasParcialmenteCriptografadas	1,63 MB
Totalmente criptografado	OcorrenciasTotalmenteCriptografadas	3,33 MB

O banco de dados em que os campos não são criptografados, possui 256 KB de armazenamento utilizado. Já os bancos em que os campos estão parcialmente e totalmente criptografados possuem,

respectivamente, 1,63 MB e 3,33 MB. Ao compararmos estas informações, percebe-se um aumento significativo do tamanho do armazenamento para as bases que possuem campos criptografados.

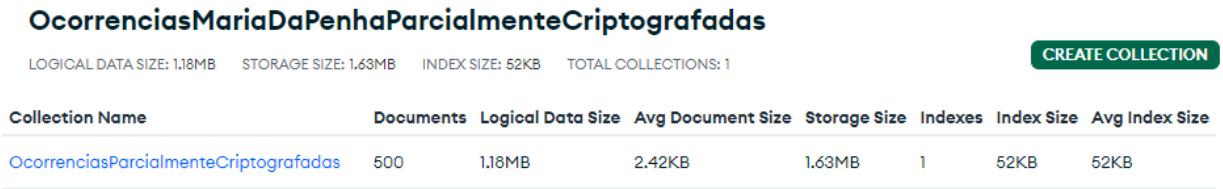
O cluster Atlas possui instâncias com parâmetros padrões. Cada instância apresenta uma determinada configuração de vCPU, capacidade de memória RAM, capacidade de armazenamento e velocidade máxima de armazenamento. Com isso, ao realizar a contratação do cluster Atlas o cliente deve escolher a instância em que o cluster melhor se adapta ao cenário que será implementado. Neste sentido, o custo atribuído a cada instância é baseado na quantidade de recursos que a mesma disponibiliza. Portanto, os bancos de dados criados para os cenários com atributos criptografados terão, consequentemente, um custo operacional maior comparado ao banco de dados sem criptografia.

Figura 4.6: Tamanho do armazenamento do banco de dados sem criptografia



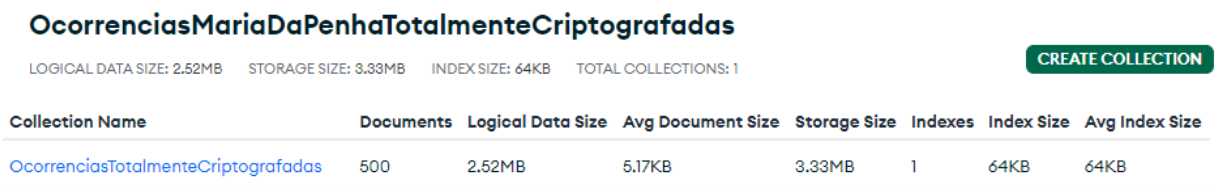
Fonte: elaborado pelo autor (2022).

Figura 4.7: Tamanho do armazenamento do banco de dados parcialmente criptografado



Fonte: elaborado pelo autor (2022).

Figura 4.8: Tamanho do armazenamento do banco de dados totalmente criptografado



Fonte: elaborado pelo autor (2022).

4.2.3 Tempo de execução das consultas

A Tabela 4.3 apresenta o tempo médio de execução das consultas para cada um dos três cenários implementados. O gráfico da Figura 4.9 apresenta um comparativo desses valores. Este gráfico está disposto na escala de $log(s)$, tal que "s" equivale ao tempo em segundos, para proporcionar uma melhor visualização das informações apresentadas.

Tabela 4.3: Tempo médio de execução das consultas para cada cenário

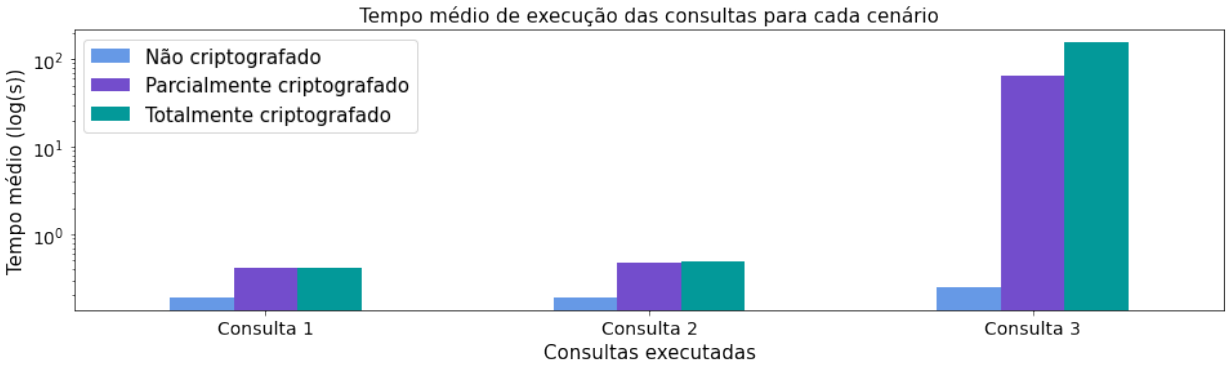
Cenário	**Consulta**	**Tempo médio (s)**
Não criptografado	Consulta 1	0,191412
	Consulta 2	0,189746
	Consulta 3	0,229058
Parcialmente criptografado	Consulta 1	0,413211
	Consulta 2	0,476043
	Consulta 3	72,877514
Totalmente criptografado	Consulta 1	0,415377
	Consulta 2	0,488998
	Consulta 3	143,475952

As Consultas 1 e 2, apresentadas na Tabela 4.1, possuem níveis de complexidade diferentes, isto é, a primeira é **simples** enquanto a segunda, **intermediária**. Porém, a partir Figura 4.9, para um mesmo cenário, esse parâmetro não gerou discrepâncias significativas nos valores de tempo de execução. Por exemplo, a Consulta 1 ao ser executada no cenário totalmente criptografado teve um tempo de execução bem próximo da Consulta 2 executada no mesmo cenário.

Já ao analisar a Consulta 3, percebe-se que o nível de complexidade atribuído a ela interferiu de forma significativa no tempo de execução. Este dado é percebido, principalmente, para os cenários totalmente criptografado e parcialmente criptografado. Para isto, basta analisar o comportamento exponencial do tempo médio de execução da Consulta 3, quando comparado às outras consultas.

Para os cenários de implementação, é possível perceber que os valores do tempo médio de execução das consultas no banco de dados totalmente criptografado são mais expressivos. Apesar disso, na base parcialmente criptografada os valores são bem próximos, exceto pela a Consulta 3, visto que, possui um nível de complexidade maior atribuído.

Figura 4.9: Gráfico comparativo do tempo de execução das consultas para os três cenários



Fonte: elaborado pelo autor (2022).

4.2.4 Tempo de processamento utilizado pela CPU

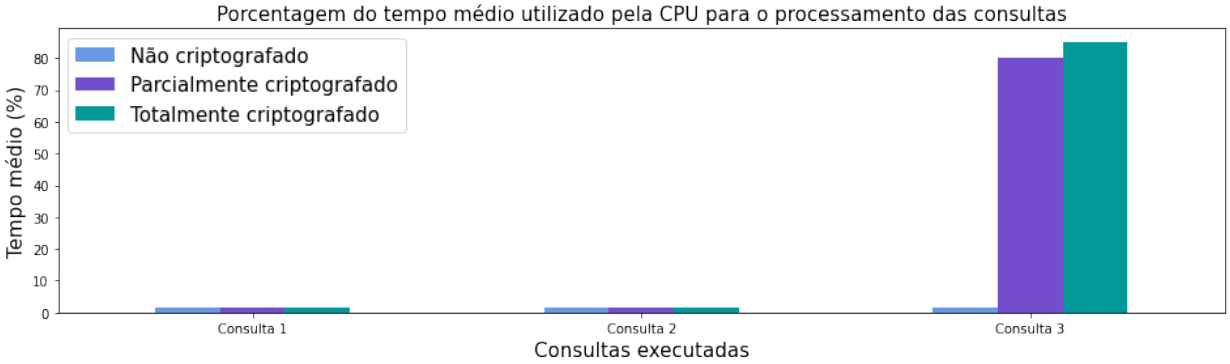
A Tabela 4.4 apresenta os valores referentes à porcentagem média do tempo em que a CPU foi utilizada atendendo os processos do MongoDB. É possível visualizar essas informações de forma gráfica a partir da Figura 4.10. Porém, por questão de escala e para garantir uma melhor análise, foi criado o gráfico da Figura 4.11 apresentando os dados referentes ao processamento das Consultas 1 e 2.

Tabela 4.4: Porcentagem média do tempo utilizado pela CPU para o processamento das consultas

Cenário	**Consulta**	**Porcentagem média (%)**
Não criptografado	Consulta 1	1,50
	Consulta 2	1,49
	Consulta 3	1,55
Parcialmente criptografado	Consulta 1	1,59
	Consulta 2	1,66
	Consulta 3	80,11
Totalmente criptografado	Consulta 1	1,65
	Consulta 2	1,74
	Consulta 3	85,16

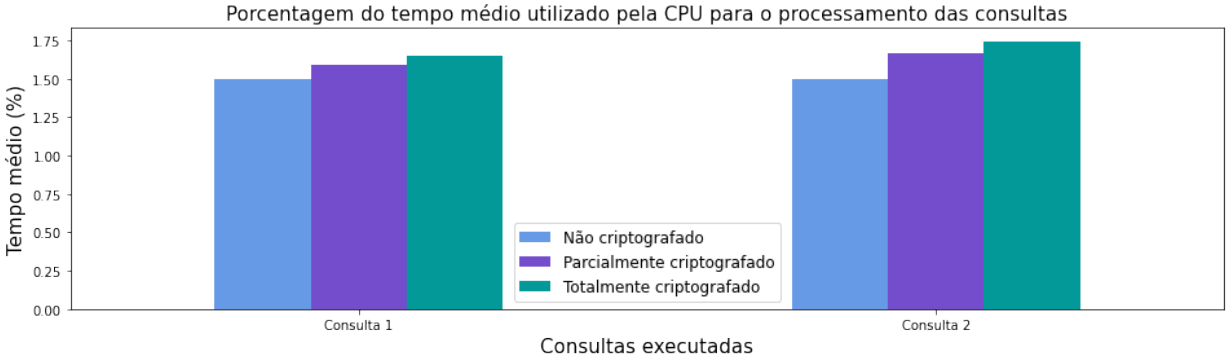
Com isso, verifica-se na Tabela 4.4 que as Consultas 1 e 2 tiveram um baixo consumo de CPU para todos os cenários. Já a Consulta 3, que possui um nível de complexidade significativo, os resultados demonstraram que, de fato, ela demanda um tempo de processamento elevado na CPU. Para o cenário parcialmente criptografado atingiu-se 80,11%, enquanto o cenário totalmente criptografado, 85,16%. Por fim, para comportar dados criptografados e viabilizar a execução de consultas expressivas nos bancos de dados, deve-se levar em consideração o consumo computacional disponível para o cluster.

Figura 4.10: Gráfico comparativo da porcentagem média do tempo utilizado pela CPU para o processamento das Consultas 1, 2 e 3



Fonte: elaborado pelo autor (2022).

Figura 4.11: Gráfico comparativo da porcentagem média do tempo utilizado pela CPU para o processamento das Consultas 1 e 2



Fonte: elaborado pelo autor (2022).

Capítulo 5

Conclusão

Devido a crescente volumetria dos dados e os desafios de segurança nas tecnologias de sistemas distribuídos em nuvem, foi estudado como manter os requisitos de segurança nesse ambiente. Com isso, o presente trabalho apresentou uma proposta de solução que permite a criptografia dos dados do lado do cliente, o armazenamento da informação em formato criptografado e a realização de consultas do lado do servidor sem que esses dados sejam descriptografados.

Assim, foi arquitetado uma solução funcional em que o esquema de Searchable Encryption é implementado a partir da Queryable Encryption no MongoDB. Para isso, se fez necessário a instalação de requisitos para o ambiente de desenvolvimento. Foi instalada a biblioteca compartilhada de criptografia automática do MongoDB e o driver *PyMongo* compatível à Queryable Encryption. Por fim, foi iniciado o cluster no ambiente *multi-cloud* (MongoDB Atlas).

A fim de estabelecer cenários comparativos para posterior análise, um conjunto de dados único foi inserido em três bancos de dados distintos. O primeiro banco de dados armazena todos os registros em formato de texto legível, ou seja, não possui nenhum dado criptografado. O segundo banco de dados armazena os registros com apenas algumas informações criptografadas. Finalmente, o terceiro banco de dados apresenta todos os dados criptografados.

Com isso, foram realizadas consultas nestes cenários de implementação com o objetivo de avaliar a aplicabilidade do esquema Queryable Encryption. Estas consultas possuem níveis de complexidade diferentes, sendo elas de natureza simples, intermediária e complexa. Neste sentido, foi analisado o desempenho de cada banco de dados quanto ao tamanho do armazenamento, o tempo de execução das consultas e o tempo de processamento utilizado pela CPU.

Os resultados apresentados mostram que o armazenamento dos dados de modo totalmente criptografado gerou um tamanho expressivo do banco. A dimensão do banco de dados neste cenário foi aproximadamente treze vezes maior que a dimensão do banco de dados sem criptografia. Assim, levando em consideração o modelo de cobrança dos sistemas de computação em nuvem, existe um *trade-off* quanto ao armazenamento de dados criptografados e ao custo operacional envolvido nesta solução.

Além disso, os bancos de dados com informações criptografadas apresentaram um tempo de

execução elevado quando submetidos à consulta de nível complexo. Portanto, é necessário considerar a eficiência da velocidade máxima de armazenamento ao contratar o serviço de cluster na nuvem. Visto que, este parâmetro influencia diretamente na performance de velocidade das consultas e das operações de leitura e gravação.

Os cenários parcialmente criptografado e totalmente criptografado não apresentaram diferenças significativas entre eles quanto ao tempo de processamento utilizado pela CPU. Ao realizar consultas complexas nestes cenários, o percentual médio de utilização da CPU foi expressivo. Nesta perspectiva, o aumento de vCPUs disponíveis no cluster proporcionaria uma redução na porcentagem de uso de CPUs. Porém, ocorreria impacto no custo operacional.

Por fim, a realização de consultas do lado do servidor sem que os dados sejam descriptografados foi de fato viabilizada. Entretanto, atualmente, as operações suportadas pelo o esquema de Queryable Encryption são limitadas. Com isso, durante o desenvolvimento foi necessário atribuir um nível de complexidade de código para realizar determinadas consultas.

5.1 Trabalhos futuros

Durante o desdobramento da proposta de solução e após as análises de resultados da mesma, foram encontrados pontos de melhoria a serem atribuídos em trabalhos futuros. Nestes trabalhos, o conjunto de dados pode ser ampliado para implementar cenários mais verídicos ao contexto Big Data. Além disso, pode ser observado o comportamento de consultas em dados não estruturados, como vídeos e imagens. Finalmente, pode-se realizar o levantamento comparativo entre bits e bytes do tamanho das informações não criptografadas e criptografadas.

Referências

ABDULLAH, Ako Muhamad et al. Advanced encryption standard (AES) algorithm to encrypt and decrypt data. **Cryptography and Network Security**, v. 16, p. 1–11, 2017.

AGNALDO, João et al. A SEGURANÇA DOS DOCUMENTOS DIGITAIS, mai. 2022.

ARMBRUST, Michael et al. A view of cloud computing. **Communications of the ACM**, ACM New York, NY, USA, v. 53, n. 4, p. 50–58, 2010.

BARROSO, Luiz André; DEAN, Jeffrey; HOLZLE, Urs. Web search for a planet: The Google cluster architecture. **IEEE micro**, IEEE, v. 23, n. 2, p. 22–28, 2003.

BOST, Raphael. $o\varphi o\varsigma$: Forward secure searchable encryption. In: PROCEEDINGS of the 2016 ACM SIGSAC Conference on Computer and Communications Security. [S.l.: s.n.], 2016. P. 1143–1154.

BRAKERSKI, Zvika; VAIKUNTANATHAN, Vinod. Efficient Fully Homomorphic Encryption from (Standard) LWE. In: 2011 IEEE 52nd Annual Symposium on Foundations of Computer Science. [S.l.: s.n.], 2011. P. 97–106. DOI: 10.1109/FOCS.2011.12.

BRASIL, Lei nº 13.709 de 2018. **Lei Geral de Proteção de Dados Pessoais (LGPD)**. 2018. Disponível em: <http://www.planalto.gov.br/ccivil_03/_Ato2015-2018/2018/Lei/L13709.htm>. Acesso em: 5 set. 2022.

BRAUND, Cynthia; BORKAR, Pramod. **MongoDB Releases Queryable Encryption Preview**. Jun. 2022. Disponível em: <https://www.mongodb.com/blog/post/mongodb-releases-queryable-encryption-preview>. Acesso em: 29 jul. 2022.

BRITO, Felipe; MACHADO, Javam. Preservação de Privacidade de Dados: Fundamentos, Técnicas e Aplicações. In: [s.l.: s.n.], jul. 2017. P. 40. ISBN 978-85-7669-374-1.

BURNETT, STEVEN. **Criptografia e segurança: o guia oficial RSA**. [S.l.]: Gulf Professional Publishing, 2002.

CHASE, Melissa; SHEN, Emily. Pattern Matching Encryption. **IACR Cryptol. ePrint Arch.**, Citeseer, v. 2014, p. 638, 2014.

CURTMOLA, Reza et al. Searchable symmetric encryption: improved definitions and efficient constructions. **Journal of Computer Security**, IOS Press, v. 19, n. 5, p. 895–934, 2011.

DE DIANA, Mauricio; GEROSA, Marco Aurélio. Nosql na web 2.0: Um estudo comparativo de bancos não-relacionais para armazenamento de dados na web 2.0. In: WORKSHOP de Teses e Dissertações de Bancos de Dados do Simpósio Brasileiro de Bancos de Dados WTDBD2010. [S.l.: s.n.], 2010.

DIAS, Jean Miguel; RITA DE CÁSSIA, MC; PIRES, Daniel Facciolo. A segurança de dados na computação em nuvens nas pequenas e médias empresas. **Revista Eletrônica de Sistemas de Informação e Gestão Tecnológica**, v. 2, n. 1, 2012.

ELGAMAL, T. A public key cryptosystem and a signature scheme based on discrete logarithms. **IEEE Transactions on Information Theory**, v. 31, n. 4, p. 469–472, 1985. DOI: 10.1109/TIT.1985.1057074.

EMERICH, Alex. **How MongoDB encrypts data**. Disponível em: <https://www.prisma.io/dataguide/mongodb/mongodb-encryption>. Acesso em: 29 jul. 2022.

GARIBALDI JUNIOR, João Lauro. Aplicabilidade de criptografia homomórfica, 2018.

GENTRY, Craig. **A Fully Homomorphic Encryption Scheme**. 2009. Tese (Doutorado) – Stanford, CA, USA. AAI3382729. ISBN 9781109444506.

HEATH, T.; BIZER, C. **Linked Data: Evolving the Web Into a Global Data Space**. [S.l.]: Morgan & Claypool, 2011. (Synthesis Lectures on Web Engineering). ISBN 9781608454303. Disponível em: <https://books.google.com.br/books?id=OFv59Wcfkx8C>.

HOWS, David; MEMBREY, Peter; PLUGGE, Eelco. **Introdução ao MongoDB**. [S.l.]: Novatec Editora, 2019.

KUROSE, James F.; ROSS, Keith W. **Computer Networking: A Top-Down Approach (6th Edition)**. 6th. Boston, MA, USA: Pearson, 2012.

MCGREW, David; FOLEY, John; PATERSON, Kenny. **Authenticated Encryption with AES-CBC and HMAC-SHA**. [S.l.], jul. 2014. 31 p. Work in Progress. Disponível em: <https://datatracker.ietf.org/doc/draft-mcgrew-aead-aes-cbc-hmac-sha2/05/>.

MELL, Peter; GRANCE, Tim et al. The NIST definition of cloud computing. Computer Security Division, Information Technology Laboratory, National ..., 2011.

MONGODB. **What is MongoDB?** Disponível em: <https://www.mongodb.com/pt-br/what-is-mongodb/>. Acesso em: 29 jul. 2022.

MONGODB, AE. **Automatic Encryption**. Disponível em: <https://www.mongodb.com/docs/manual/core/csfle/fundamentals/automatic-encryption/#std-label-csfle-fundamentals-automatic-encryption>. Acesso em: 29 jul. 2022.

MONGODB, AESL. **Automatic Encryption Shared Library**. Disponível em: <https://www.mongodb.com/docs/upcoming/core/queryable-encryption/reference/shared-library/#std-label-qe-reference-shared-library-download>. Acesso em: 29 jul. 2022.

MONGODB, CSFLE. **Client-Side Field Level Encryption**. Disponível em: <https://www.mongodb.com/docs/manual/core/csfle/>. Acesso em: 29 jul. 2022.

MONGODB, EC. **Encryption Components**. Disponível em: <https://www.mongodb.com/docs/manual/core/csfle/reference/encryption-components/#std-label-csfle-reference-encryption-components>. Acesso em: 29 jul. 2022.

MONGODB, EE. **Explicit Encryption**. Disponível em: <https://www.mongodb.com/docs/manual/core/csfle/fundamentals/manual-encryption/#std-label-csfle-fundamentals-manual-encryption>. Acesso em: 29 jul. 2022.

MONGODB, FEQ. **Field Encryption and Queryability**. Disponível em: <https://www.mongodb.com/docs/upcoming/core/queryable-encryption/fundamentals/encrypt-and-query/#std-label-qe-fundamentals-encrypt-query>. Acesso em: 29 jul. 2022.

MONGODB, QE. **Queryable Encryption — MongoDB Manual**. Disponível em: <https://www.mongodb.com/docs/manual/core/queryable-encryption/>. Acesso em: 29 jul. 2022.

OGBURN, Monique; TURNER, Claude; DAHAL, Pushkar. Homomorphic encryption. **Procedia Computer Science**, Elsevier, v. 20, p. 502–509, 2013.

OLIVEIRA, Ronielton Rezende. Criptografia simétrica e assimétrica-os principais algoritmos de cifragem. **Segurança Digital [Revista online]**, v. 31, p. 11–15, 2012.

OLIVEIRA, Samuel Silva de. Bancos de dados não-relacionais: um novo paradigma para armazenamento de dados em sistemas de ensino colaborativo. **Revista da Escola de Administração Pública do Amapá**, v. 2, n. 1, p. 184–194, 2014.

ORDONEZ, Edward; PEREIRA, Fábio; CHIARAMONTE, Rodolfo. **Criptografia em Software e Hardware**. [S.l.: s.n.], mar. 2005. ISBN 85-522-069-1.

PCDF. **Delegacia Eletrônica Maria da Penha On-line**. Disponível em: <https://www.pcdf.df.gov.br/servicos/delegacia-eletronica/violencia-domestica-contra-mulher>. Acesso em: 3 jul. 2022.

PYMONGO. **PyMongo 4.2.0 Documentation**. 2008. Disponível em: <https://pymongo.readthedocs.io/en/stable/>. Acesso em: 29 jul. 2022.

RAI, Rashmi; SAHOO, Gadadhar; MEHFUZ, Shabana. Exploring the factors influencing the cloud computing adoption: a systematic study on cloud migration. **SpringerPlus**, Springer, v. 4, n. 1, p. 1–12, 2015.

RIVEST, R L; ADLEMAN, L; DERTOUZOS, M L. On Data Banks and Privacy Homomorphisms. **Foundations of Secure Computation, Academia Press**, p. 169–179, 1978.

RIVEST, Ronald L. Cryptography. In: ALGORITHMS and complexity. [S.l.]: Elsevier, 1990. P. 717–755.

RIVEST, Ronald L; SHAMIR, Adi; ADLEMAN, Leonard. A method for obtaining digital signatures and public-key cryptosystems. **Communications of the ACM**, ACM New York, NY, USA, v. 21, n. 2, p. 120–126, 1978.

SADALAGE, Pramod J; FOWLER, Martin. **NoSQL essencial: um guia conciso para o mundo emergente da persistência poliglota**. [S.l.]: Novatec Editora, 2019.

SIMMONS, Gustavus J. Symmetric and asymmetric encryption. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 11, n. 4, p. 305–330, 1979.

TANG, Jun et al. Ensuring security and privacy preservation for cloud data services. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 49, n. 1, p. 1–39, 2016.

TERADA, Routo. **Segurança de dados: criptografia em rede de computador**. [S.l.]: Editora Blucher, 2008.

WANG, Yunling; WANG, Jianfeng; CHEN, Xiaofeng. Secure searchable encryption: a survey. **Journal of communications and information networks**, Springer, v. 1, n. 4, p. 52–65, 2016.

WEBER, Raul Fernando. Criptografia contemporânea. In: VI Simpósio de Computadores Tolerantes a Falhas. [S.l.: s.n.], 1995. P. 7–32.

ANEXO I

Mapeamento de campos criptografados

```
encrypted_db_name = "OcorrenciasMariaDaPenhaParcialmenteCriptografadas"
encrypted_coll_name = "OcorrenciasParcialmenteCriptografadas"

encrypted_fields_map = {
    f"{encrypted_db_name}.{encrypted_coll_name}": {
        "fields": [
            {
                "keyId": data_key_id_1,
                "path": "envolvido.is_vitima",
                "bsonType": "bool",
                "queries": {"queryType": "equality"},
            },
            {
                "keyId": data_key_id_2,
                "path": "envolvido.cpf",
                "bsonType": "string",
                "queries": {"queryType": "equality"},
            },
            {
                "keyId": data_key_id_3,
                "path": "envolvido.UF",
                "bsonType": "string",
                "queries": {"queryType": "equality"},
            },
            {
                "keyId": data_key_id_4,
                "path": "envolvido.nascimento",
                "bsonType": "string",
                "queries": {"queryType": "equality"},
```

```
            },
            {
                "keyId": data_key_id_5,
                "path": "data_ocorrido",
                "bsonType": "string",
                "queries": {"queryType": "equality"},
            },
            {
                "keyId": data_key_id_6,
                "path": "envolvido.grau_instrucao",
                "bsonType": "string",
                "queries": {"queryType": "equality"},
            },
            {
                "keyId": data_key_id_7,
                "path": "envolvido.raca_cor",
                "bsonType": "string",
                "queries": {"queryType": "equality"},
            },
        ],
    },
}
```

ANEXO II

Classe CRUD

```
encrypted_coll = secure_client[encrypted_db_name][encrypted_coll_name]

class CRUD:
    def insert_all(dataset, encrypted_coll):
        f = open(dataset, encoding='utf-8')
        arr = json.load(f)

        for item in arr:
            encrypted_coll.insert_one(item)

    def drop_database(secure_client, encrypted_db_name):
        secure_client.drop_database(encrypted_db_name)

    def create_collection(secure_client, encrypted_db_name,
    encrypted_coll_name):
        encrypted_db = secure_client[encrypted_db_name]
        encrypted_db.create_collection(encrypted_coll_name)

    def find_one_decrypted(query, encrypted_coll):
        pprint.pprint(encrypted_coll.find_one(query))

    def find_all_decrypted(query, encrypted_coll):
        cursor = encrypted_coll.find(
            query
        )
        for result_object in cursor:
            pprint.pprint(result_object)

    def find_one_encrypted(query, unencryptedClient):
```

```
    pprint.pprint(unencryptedClient.find_one(query))

def find_all_encrypted(query, unencryptedClient):
    cursor = unencryptedClient.find(
        query
    )
    for result_object in cursor:
        pprint.pprint(result_object)

def count_documents(query, encrypted_coll):
    pprint.pprint(encrypted_coll.count_documents(query))
```

ANEXO III

Classe DataInputs

```
class DatesInputs:
    @staticmethod
    def generate_dates(age):

        today = date.today()

        bithday_year = today.year - age
        bithday_month = today.month
        bithday_day = today.day

        bithday_date = date(bithday_year, bithday_month, bithday_day)

        dates = []

        dates = pandas.date_range(
            bithday_date,
            today-timedelta(days=1),
            freq='d'
            ).strftime('%Y-%m-%d').to_list()

        return dates
```