VOL.39 Nº3 2022
https://doi.org/10.2271\IMECTA.10770.pt

Uso de blockchain no controle e rastreabilidade de acesso a dados armazenados em nuvem

Souza Jonatan[a], Pinto Raquel[b], Bruno Schulze[c]
[a]Instituto Militar de Engenharia (IME), jonatangd.souza@gmail.com
[b]Instituto Militar de Engenharia (IME), raquel@ime.eb.br
[c]Laboratório Nacional de Computação Científica (LNCC), schulze@lncc.br

RESUMO: A computação em nuvem oferece diversos serviços, como armazenamento de dados e máquinas virtuais (VMs). Esses recursos são disponibilizados pela Internet e seu pagamento é dado pelo uso. Embora os serviços de nuvem sejam eficientes, há uma preocupação crescente na segurança e privacidade desses serviços prestados por nuvens computacionais. Entre essas preocupações, pode-se destacar o compartilhamento de dados entre usuários. Estes serviços não possuem um mecanismo de permissão que seja auditável pelo proprietário dos dados. Neste contexto, o blockchain tem se destacado principalmente por sua arquitetura de ledgers distribuídos que permite uma trilha imutável e auditável. Além disso, a arquitetura descentralizada do blockchain elimina a necessidade de confiança em terceiros. Portanto, este artigo apresenta uma arquitetura baseada no uso da tecnologia blockchain como repositório seguro e auditável de registro dos acessos e permissões concedidas aos usuários. Como resultado, este artigo apresenta um estudo de caso validando a arquitetura proposta.

PALAVRAS-CHAVE: Computação em nuvem. Blockchain. Compartilhamento de dados. Segurança. Privacidade.

ABSTRACT: Cloud computing offers a variety of services such as data storage and virtual machines (VMs) . These features are made available over the Internet and your payment is per use. While cloud services are efficient, there is growing concern for the security and privacy of these services provided by cloud computing. Among these concerns, the sharing of data between users can be highlighted. These services do not have a permit system that is auditable. In this context, the blockchain has stood out mainly for its distributed ledger architecture that allows an immutable and auditable trail. In addition, the blockchain's decentralized architecture eliminates the need to rely on third parties. Therefore, this article presents an architecture based on the use of blockchain technology as a secure and auditable repository for recording access and permissions granted to users. As a result, this article presents a case study validating the proposed architecture.

KEYWORDS: Cloud computing. Blockchain. Data sharing. Security. Privacy. Data storage.

1. Introdução

O compartilhamento de dados entre usuários e serviços têm sido cada vez mais comum, principalmente com a crescente adoção da internet das coisas (*IoT - Internet of Things*). Em sistemas IoT, diversos dispositivos inteligentes interagem entre si gerando dados de diversos contextos. Considerando a grande quantidade de dados e a capacidade reduzida de armazenamento dos dispositivos pessoais, a utilização de serviços de armazenamento de dados tem sido cada vez mais comum tanto por usuários comuns como por empresas de grande porte.

Esta migração de armazenamento se dá à medida que as limitações de hardware e infraestrutura para comunicação vão sendo mitigadas. Desta forma, serviços de nuvem são frequentemente usados. Dentre as principais vantagens, podemos destacar: sua simplicidade, baixo custo financeiro e alta disponibilidade dos recursos.

A computação em nuvem pode ser definida como um tipo de sistema distribuído, constituído de um conjunto de computadores interconectados e virtualizados. Estes recursos são disponibilizados dinamicamente como recursos computacionais unificados, cujos serviços são baseados em acordos de nível de serviço [1]. O ambiente de nuvem é compartilhado entre diversos usuários onde a demanda por hardware e software pode ser contratada ou vendida a qualquer momento. Por isso, a nuvem deve ser capaz de crescer elasticamente, conforme a demanda desses recursos [2].

Considerando as vantagens mencionadas, serviços de nuvem tem adoção em larga escala, em diversos contextos, porém é necessário considerar que se trata de um serviço de terceiros para o armazenamento de

dados. Sendo assim, há uma necessidade da confiança do proprietário neste serviço para permitir ou revogar acesso a usuários terceiros, e não apenas no tocante ao armazenamento, uma vez que o controle de acesso e permissões estão do lado do serviço.

Portanto, existe uma preocupação crescente na segurança desses serviços prestados por nuvens computacionais: I) a garantia de que os dados são compartilhados apenas com os usuários autorizados pelo proprietário; II) os serviços, em geral, não possuem um mecanismo de permissão que seja auditável pelo proprietário dos dados.

Neste contexto, o *blockchain* tem ganhado atenção principalmente por sua arquitetura de ledgers (livro-razão) distribuído. O *blockchain* consiste em uma cadeia de blocos interligados através do uso de um ou mais *hashes* do bloco anterior, oferecendo uma trilha imutável e auditável.

Este trabalho apresenta uma arquitetura baseada em *blockchain* que provê um ambiente seguro e auditável de registro dos acessos e permissões concedidas, transferindo assim a necessidade de confiança de compartilhamento a dados com diferentes usuários, para um ambiente onde esses dados sejam auditáveis e imutáveis. Sendo assim, o proprietário dos dados terá as condições de auditar as permissões dadas e revogadas aos seus dados, além de verificar quando estes foram acessados.

Este artigo está organizado da seguinte forma. A Seção II apresenta o cenário atual do *blockchain* e os principais conceitos. A Seção III aborda alguns dos principais trabalhos relacionados. A Seção IV apresenta a arquitetura proposta em detalhes. A Seção V aborda um estudo de caso da arquitetura em um contexto IoT. Por fim, a Seção VI conclui o artigo com algumas discussões e orientações futuras.

2. Conceitos básicos de blockchain

Esta seção apresenta uma breve descrição de conceitos e tecnologias importantes adotados neste estudo de pesquisa, relacionados a *blockchain*.

Blockchain é uma estrutura de dados descentralizada que é replicada e compartilhada entre os membros de uma rede denominados *peers*. Cada bloco contém um conjunto ordenado de transações e um hash que são marcados com um *timestamp*. Cada bloco inclui também o *timestamp* do bloco anterior em seu hash, formando uma lista encadeada de blocos, com cada *timestamp* adicional reforçando os anteriores [14].

A estrutura de dados do bloco é composta por um *header* e a lista de transações. O *header* contém metadados sobre um bloco e o *hash* do bloco anterior. Para cada bloco N, o hash do bloco N-1 é considerado. O bloco de configuração que inicializa os componentes e serve como o primeiro bloco em uma cadeia é chamado de bloco da gênese. Os blocos são criados pelo *Ordering Service* e validados pelos outros elementos da rede.

A descentralização significa que nenhum dos *peers* tem a capacidade de controlar sozinho o processamento de todas as transações na rede.

Blockchains também são arquiteturalmente descentralizados pois não existe nenhum ponto central de falha, porém todos devem concordar com um único estado através de um consenso.

O *ledger* ou livro-razão contém o estado atual de um negócio e funciona como um diário de transações. O livro-razão no *blockchain*, possui basicamente duas atribuições: I) apresentar o valor atual de um conjunto de estados; e II) manter o histórico das transações que determinaram esses estados.

Uma das principais características do *blockchain* é a imutabilidade dos dados armazenados. Isto se deve a recorrência de *hashes*, onde blocos anteriores não podem ter o seu conteúdo violado e continuam válidos. Portanto registros de transações são permanentes, tendo em vista que se alterados toda a cadeia de blocos posterior estará inválida.

Considerando a arquitetura descentralizada, alcançar o consenso neste cenário é um desafio para uma rede *blockchain*. Alcançar consenso garante que todos os nós da rede concordam com um estado global consistente da cadeia de blocos. Isto é importante no *blockchain* pois garante que os dados armazenados não podem ser violados, a menos que o atacante obtenha o controle de (50% + 1) nós da rede, tendo em

vista que esta é a quantidade que é necessária para a validação na maioria das implementações. Além disso, blocos anteriores ao último quando modificado seu conteúdo, tornam todos os blocos à sua frente inválidos. Esta característica é dada pela recorrência de *hash*.

Uma vez que o *blockchain* tem responsabilidade com o *distributed ledger* e toda estrutura de dados imutável, o *smart contract* estende esta função incluindo uma linguagem para termos de acordo e medições garantindo que determinadas condições sejam atendidas.

De forma minimalista, *smart contracts* são scripts que atuam no *blockchain* e possuem uma transação de disparo *(triggering transaction)* que é responsável por executar ações [15]. Como eles residem na cadeia do *blockchain*, possuem um endereço único. Ao executar uma transação que tenha um *smart contract* endereçado, este executa independentemente e automaticamente de forma prescrita em todos os nós da rede, de acordo com os dados que foram incluídos na transação de disparo do *smart contract* [16] [17].

Existem diversas plataformas *blockchain* disponíveis para implementação de soluções. Essas plataformas se dividem basicamente em dois grandes grupos: blockchains públicos e *blockchains* privados ou permissionados [18].

Os *blockchains* públicos, por serem abertos, possuem algoritmos de consenso mais robustos e computacionalmente mais caros. Normalmente, é necessário a inserção da figura do minerador. *Blockchains* privados, são o oposto pois são compostos de nós conhecidos. Desta forma, possuem algoritmos de consenso mais simples, tornando as transações mais rápidas e permitindo mudanças na arquitetura em um tempo menor.

3. Trabahos relacionados

Serviços de nuvem, em geral, oferecem praticidade, escalabilidade do uso dos serviços, alta disponibilidade e gerenciamento de recursos. Embora todos esses benefícios oferecidos sejam relevantes para a decisão da migração de uma solução local para um serviço de nuvem, algumas questões relacionadas à segurança e integridade, requerem o monitoramento permanente dessas informações. Em [3] os autores propõem uma arquitetura de software que permite o armazenamento de arquivos em serviços na nuvem, com garantia de privacidade das informações, além de um monitoramento permanente da integridade dos arquivos, com base em tecnologias como *blockchain*. Neste trabalho, os autores destacam dois obstáculos principais para a adoção de plataformas de *blockchain*, que são, alto consumo de energia e baixa velocidade de processamento de transações, tendo em vista o demasiado uso de criptografia e algoritmos de consenso entre os *peers* Assim justificam o uso do *Hyper Ledger Fabric* como solução mais eficiente para o cenário proposto.

Em [4] é apresentado uma arquitetura baseada em *blockchain*, *smart contracts* e tecnologias de confiança computacional que seja capaz de realizar o monitoramento periódico da integridade dos arquivos armazenados na nuvem. Dentre as possíveis aplicações para a arquitetura proposta, os autores destacam o armazenamento de backups da base de dados dos sistemas de gestão eletrônica de documentos. Em geral, são arquivos grandes e, devido a questões legais, precisam ser armazenados por longos períodos de tempo.

Para a implementação do *blockchain*, os autores optaram pela plataforma *Ethereum*, tendo em vista a disponibilidade de documentação, a facilidade de criar uma rede para realizar testes em um ambiente local, e também pelo número de ferramentas disponíveis para apoiar o uso e a execução do teste. O trabalho também apresenta uma análise da segurança da arquitetura.

O compartilhamento de informações armazenadas nos serviços de nuvem, também tem levantado preocupações. Geralmente, serviços de nuvem não oferecem uma solução auditável, necessitando assim a confiança do contratante no serviço de terceiros. Em [5], os autores destacam que o compartilhamento de dados de grupo dinâmico, onde usuários anonimamente compartilham seus dados com outros membros do grupo, utilizando o serviço de nuvem,

pode comprometer a segurança. Portanto, evidenciam a necessidade de projetar um sistema eficiente e seguro de compartilhamento de dados em grupos dinâmicos. Este trabalho apresenta uma revisão dos desafios encontrados em projetar, de forma eficiente, um compartilhamento de dados em grupo dinâmico. Entre os desafios encontrados, incluem, autenticação de usuário, privacidade e segurança, confidencialidade de dados, integridade e custo de consulta. Também mencionam problemas baseados no provedor de serviços que incluem a identidade do usuário e sua rastreabilidade, e revogação do usuário.

Segurança, privacidade e a integridade de dados em serviços de nuvem, tem motivado diversos pesquisadores. Em [6], é destacado que as técnicas de verificação pública podem permitir que um usuário empregue um auditor terceirizado para verificar a integridade dos dados em seu nome. No entanto, os esquemas de verificação pública existentes são vulneráveis, pois permitem que os auditores não realizem verificações a tempo. Neste contexto, os autores propõem um mecanismo para verificação pública da integridade do armazenamento em nuvem de arquivos resistentes à procrastinação de auditores, sem o uso de certificados, tendo em vista que este tipo de mecanismo utiliza, em sua maioria, infraestrutura de chave pública (PKI) e, portanto, sofrem de problemas de gerenciamento de certificados. O mecanismo denominado esquema de verificação pública sem certificado contra auditores *procrastinates* ou *Certificateless Public Verification scheme against Procrastinating Auditors* (CPVPA), utiliza tecnologia *blockchain*, e tem como objetivo exigir que os auditores registrem cada verificação executada em uma transação no *blockchain*. Tendo em vista que as transações no *blockchain* são sensíveis ao tempo, estrategicamente, a verificação pode receber um carimbo de data/hora após a transação ser registrada no blockchain, o que permite que os usuários verifiquem se os auditores realizaram suas verificações no tempo prescrito.

Devido à capacidade limitada de processamento dos dispositivos que normalmente compõem uma rede IoT, os dispositivos geralmente utilizam serviços de terceiros controlados externamente para executar processamento adicional requerido, como uma nuvem computacional por exemplo. IOT SMART CONTRACT [7], é uma solução proposta para o gerenciamento descentralizado de acesso a dados usando *blockchain* e a proteção de privacidade de dados oferecida pelo Intel SGX. Esta solução tem como objetivo estabelecer confiança entre os provedores de serviços de IoT e os usuários desses serviços. Através dos *smart contracts*, a plataforma proposta provê um gerenciamento de acesso a dados onde os usuários têm privilégio em controlar como seus dados são compartilhados ou usados. Além disso, é possível atribuir regras de acesso a dados que são aplicadas de forma autônoma por serviços de terceiros não confiáveis na rede *blockchain*.

A segurança e a privacidade da Internet das Coisas (IoT) é um desafio iminente, devido à enorme escala e a natureza distribuída das redes IoT. As abordagens baseadas em *blockchain* oferecem segurança descentralizada e privacidade, mas envolvem consumo excessivo de energia e aumento de latência, o que pode ser um problema para a maioria dos dispositivos IoT com recursos limitados.

Em [8], é proposta uma solução utilizando o *blockchain* no contexto de proveniência de dados na nuvem computacional. Em [9], é introduzido o uso de *blockchain* para segurança e privacidade em um cenário de cidades inteligentes *(smart cities)*.

Dispositivos IoT podem sofrer diferentes tipos de ataques, principalmente ecossistemas de acesso público, como por exemplo medidores inteligentes. Os medidores inteligentes ajudam as concessionárias de energia a otimizar sua lucratividade por meio da redução de despesas associadas a roubo de energia e perdas técnicas. Os consumidores, por outro lado, passam a ter acesso a dados de consumo de energia em tempo real, que poderão usar para aumentar sua eficiência energética, reduzir suas contas mensais e ajudar a concessionária a estabilizar a rede durante os períodos de pico. Evitar ameaças à segurança assim como falsificação dos dados é um desafio. Em [13] é apresentada uma solução utilizando o *blockchain* para evitar ameaças à segurança nestes ecossistemas. Neste

trabalho ainda é utilizada a abordagem de *Zero-Knowledge proof*, uma tecnologia de melhoria de anonimato do *blockchain* que mitiga as ameaças à segurança, como violação de informações pessoais. O trabalho ainda propõe o uso de *smart contracts* para evitar adulteração de dados e aumentar a confiabilidade dos medidores.

Observando os trabalhos relacionados nesta seção, conclui-se que embora todos os trabalhos utilizem *blockchain*, a maioria utiliza *blockchain* público. Neste artigo propomos o uso de *blockchain* privado implementado através do *Hyper Ledger Fabric*. Esta solução acrescenta flexibilidade e simplicidade, tendo em vista que soluções de *blockchain* públicas demandam uma infraestrutura robusta com a adição de mineradores.

4. Arquitetura para controle e rastreabilidade de acesso a dados

Tendo em vista as limitações dos serviços de nuvem para prover uma trilha auditável de acesso aos dados, bem como transparência nas permissões e revogação das permissões de acesso aos dados, a arquitetura ilustrada pela **figura 1**, apresenta uma solução que atende esses requisitos.

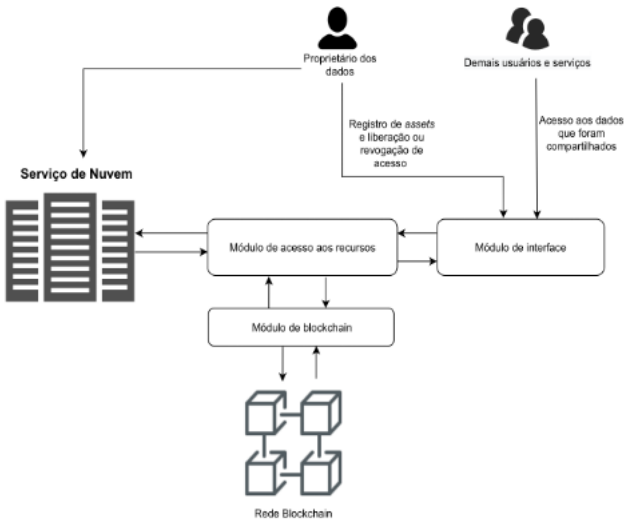

Fig. 1 – Arquitetura proposta

Nesta arquitetura, os proprietários criam contas, cadastram seus *assets* armazenados na nuvem e cadastram o compartilhamento com terceiros através do **Módulo de interface**. Os usuários utilizam esse módulo para se cadastrar e acessar os *assets* compartilhados.

O **Módulo de acesso aos recursos** oferece APIs utilizadas pelo Módulo de interface para acessar os *assets* na nuvem e cadastrar e validar as operações realizadas no *blockchain*.

O **Módulo do *blockchain***, por sua vez, possui uma API específica ao *blockchain* e todos os *smart contracts* necessários para validar e criar transações.

Nas próximas subseções, esses módulos são descritos de forma mais detalhada.

4.1 Módulo de interface

Através do Módulo de interface, o proprietário cria uma conta e cadastra as credenciais necessárias, para que seja possível acesso posterior aos *assets* no local onde os mesmos estão armazenados. Com a conta criada, e com o e-mail e credenciais verificados, o proprietário dos dados pode realizar o login, e assim cadastrar os *assets* que deseja compartilhar com terceiros. Ao cadastrar um *asset* deve ser validado no Módulo de *blockchain*, a fim de verificar se o *asset* já existe. Além disso, deve ser validado pelo Módulo de acesso aos recursos, para verificação da existência dos dados associados ao *asset* na nuvem.

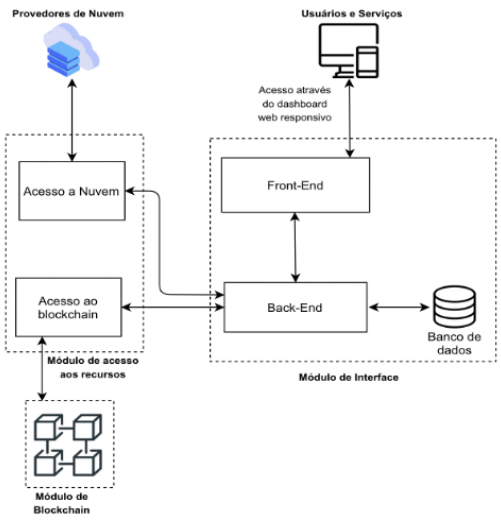

Fig. 2 – Módulo de interface e Módulo de acesso aos recursos

Todos os dados cadastrados precisam ser persistidos no *blockchain*, porém existem dados que não necessitam de uma trilha auditável. Por esta razão, o módulo de interface, conta com um banco de dados, que é usado para cadastro de metadados, informações adicionais, entre outros.

Após o cadastro dos *assets*, o proprietário pode visualizar a lista de *assets* cadastrados, onde é possível visualizar o histórico das operações sobre o *asset*, bem como garantir ou revogar acesso a terceiros. Usuários terceiros recebem um convite para acessar os *assets*. Após seu login, é apresentada uma lista dos *assets* compartilhados. Assim, o usuário pode solicitar acesso aos *assets* compartilhados com ele. Todas as operações nos *assets* são validadas pelo módulo de *blockchain*, o que garante o registro das transações, gerando uma trilha auditável de acesso.

Desta forma, este módulo tem a responsabilidade de ser a porta de entrada dos usuários. A **figura 2** ilustra seus principais componentes.

O *Front-End* é um portal *web* responsivo a diferentes tamanhos de tela, comum a interface intuitiva, onde os usuários realizam as operações propostas pela arquitetura.

O *Front-End* é um SPA *(Single Page Application)*. Essa abordagem permite o desenvolvimento de uma aplicação mais robusta e desacoplada do servidor.

O *Back-End* é uma API REST capaz de atender às demandas do *Front-End* e, por sua vez, ter acesso ao módulo de acesso aos recursos e conexão com um banco de dados. Este banco de dados, idealmente, é não estruturado, orientado a documentos. A escolha desta abordagem se dá pela flexibilidade da estruturação que além de tornar a escalabilidade mais simples, facilita a inserção e acesso aos dados.

O acesso ao *Back-End* é permitido apenas a usuários autorizados via login, onde estes só podem visualizar dados ou ações executadas pelos mesmos. Como exemplo, um usuário autorizado através de login não seria autorizado a acessar detalhes de um *asset* que não lhe pertence, mesmo conhecendo o identificador deste *asset*. O único momento que se é permitido este tipo de acesso, é quando o usuário está na lista de autorizados. Neste contexto, as informações concedidas sobre este *asset* são limitadas, preservando as informações sensíveis, como o *bucket* ou o serviço de nuvem utilizado, por exemplo.

O *Back-End* deve ser capaz de armazenar informações sobre o usuário, bem como nome, e-mail, metadados dos *assets*, *hash* da senha e informações sobre o provedor de nuvem. Além disso, ele deve ser capaz de acessar o Módulo de acesso aos recursos através de uma chave secreta. Portanto, apenas o *Back-End* do módulo de interface é capaz de acessar o módulo de acesso aos recursos.

Através da disponibilização das credenciais cadastradas, é possível mapear os recursos e disponibilizá-los como *assets*. Quando os *assets* são cadastrados, o *Back-End* envia uma transação para o *blockchain* informando o novo recurso que deve ser persistido. Uma vez cadastrado, este *asset* fica disponível para ser compartilhado com outros usuários. Todas as requisições no tocante ao acesso aos *assets*, são persistidas também no *blockchain*, através de uma solicitação do *Back-End*.

4.2 Módulo de acesso aos recursos

O **Módulo de acesso aos recursos**, tem dois objetivos principais: I) Acessar recursos externos, pois o módulo deve ser capaz de acessar os recursos na nuvem para disponibilização; e II) Submeter transações ao módulo de *blockchain*. A **figura 2** ilustra os detalhes do módulo de acesso aos recursos.

Por razões de segurança, o módulo de acesso aos recursos, deve ser acessado unicamente pelo *Back-End* do módulo de interface, estando em uma rede privada.

O **Acesso à nuvem** é responsável por integrar provedores de nuvem e abstrair as diferenças em uma interface comum. O módulo possibilita o acesso aos dados em um *bucket*, bem como disponibiliza-os para acesso a terceiros autorizados.

Os recursos da nuvem estão desacoplados do restante da solução estrategicamente, tendo em vista que uma futura mudança de provedor de nuvem ou mesmo, uma solução de armazenamento local, teria um impacto pequeno e pontual para adaptação da solução.

O **Acesso ao *blockchain***, é responsável por despachar todas as requisições que devem ser validadas pelo *blockchain*, e funcionar como um segundo autenticador, onde os recursos são disponibilizados apenas quando o Acesso ao *blockchain* recebe um *feedback* positivo da transação. Caso contrário, a requisição inicial, receberá o status de não autorizado.

Portanto, o Acesso ao *blockchain* realiza um encapsulamento onde é preparada a requisição HTTP no formato esperado pelo Módulo de *blockchain*, sendo também capaz de interpretar o retorno do Módulo de *blockchain*, e reportar as respostas dos *peers* para ciência do usuário solicitante.

4.3 Módulo de Blockchain

Na arquitetura proposta, utiliza-se uma rede *blockchain* privada para armazenar todas as transações realizadas com os dados armazenados na nuvem. A escolha do uso do *blockchain* ao invés de um banco de dados distribuído, tem duas principais motivações: I) Dada sua natureza de construção em blocos subsequentes, onde seu conteúdo criptográfico possui um hash que aponta para o bloco seguinte, os dados se tornam imutáveis, invalidando a cadeia caso uma registro anterior seja alterado; e II) *Blockchains* são descentralizados, esta característica somada ao mecanismos de consenso, garantem um nível elevado de segurança para solução, uma vez que o invasor precisaria ter em seu controle a maioria (50% + 1) dos nós da rede em seu controle.

A decisão do uso de uma rede *blockchain* privada, se dá, tendo em vista a natureza da solução. Ambas as redes *blockchain*, privada e pública, são soluções descentralizadas, possuem mecanismo de consenso e conferem a alta segurança das informações e transações [18]. Porém no *blockchain* público, qualquer entidade pode participar da rede, enquanto no privado, existem mecanismos que controlam a entrada de novos nós na rede. Além disso, na rede *blockchain* pública, existe um incentivo aos usuários que contribuem com as validações, denominado criptomoeda. Esses usuários são chamados de mineradores, pois o processo de trabalho para criptografar dados em troca de pequenas frações de criptomoedas é denominado processo de mineração. Em *blockchains* privadas não existe esse incentivo, uma vez que a rede normalmente é criada para um propósito específico. Por fim, como o número de nós na cadeia de blocos privada é menor, as transações demandam menos tempo para serem registradas, pois o algoritmo de consenso converge mais rapidamente. Diferentemente das cadeias públicas, com milhares de participantes.

Tendo em vista que uma rede *blockchain* privada não permite acesso direto aos seus *peers*, faz-se necessária a criação de um módulo que encapsule as funcionalidades do *blockchain* e as tornem disponíveis para os aplicativos utilizarem. O **Módulo de *blockchain*** tem a responsabilidade de tornar disponível essas funcionalidades de forma segura, para que outros serviços possam acessar os recursos providos pelo *blockchain*, a **figura 3** ilustra seu funcionamento.

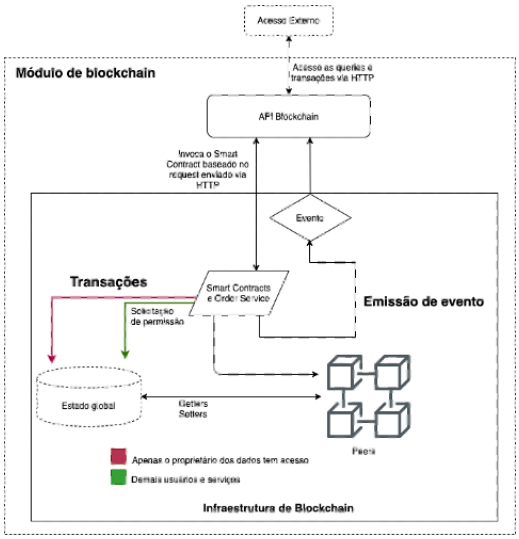

Fig. 3 – Módulo de blockchain

A fim de ter uma trilha auditável, toda a interação do mundo externo com o *blockchain* é considerada uma transação. Portanto, o módulo recebe todas as transações via HTTP, valida a integridade da informação e em seguida, a proposta de transação é construída.

Cada *peer* possui uma cópia dos *smart contracts* responsáveis por garantir a correta validação de cada transação. No contexto da arquitetura proposta, os seguintes *smart contracts* devem ser implementados:

- **Consultar *asset*:** esta transação realiza a leitura de um asset baseado em seu identificador;
- **Cadastrar *asset*:** esta transação cria um *asset* no *ledger*, com os dados necessários para a permissão e revogação de acesso;
- **Garantir acesso a *asset*:** transação na qual o proprietário garante acesso a um terceiro;
- **Revogar acesso a *asset*:** transação na qual o proprietário revoga acesso a um terceiro;
- **Consulta por proprietário:** esta transação retorna todos os *assets* do proprietário solicitante;
- **Solicitar permissão de acesso a *asset*:** esta transação é responsável por garantir ou revogar acesso a um *asset*, tendo como solicitante um terceiro; e
- **Consultar histórico do *asset*:** esta transação retorna a trilha auditável de todas as operações realizadas em um *asset*;

Todas as transações mencionadas, realizam as seguintes validações: I) se o identificador do *asset* solicitado existe no *ledger*; e II) se o requisitante pode realizar a operação solicitada. Para a solicitação de permissão é gerada uma nova atualização no *asset*, registrando os detalhes do acesso solicitado.

No momento de cadastro do usuário no sistema, o Módulo de *Back-End*, aciona o *Módulo de blockchain*, que por sua vez, é responsável por criar o usuário e gerar todo material criptográfico necessário para que este usuário tenha a permissão de acesso aos dados. Neste momento, todos os usuários são criados com o mesmo perfil, sendo capazes de submeter novas transações ao *blockchain*. No entanto, o usuário pode apenas submeter transações referentes aos assets que são proprietários. Sendo assim, qualquer usuário tem permissão para cadastrar novos *assets*, bem como submeter transações para alterá-los. Através da validação dos *smart contracts*, um usuário ao tentar submeter uma transação relacionada a um *asset* de um terceiro, essa transação terá o status de falha, e um registro dessa tentativa estará disponível na trilha auditável fornecida pelo *blockchain*.

Os *peers* endossantes recebem as entradas da proposta de transação como argumentos para invocar o *smart contract*. Cada *peer* anexa um bloco à cadeia do canal e, para cada transação válida, os conjuntos de gravação são confirmados no banco de dados de estado atual. Um evento é emitido por cada *peer* para notificar o aplicativo cliente de que a transação foi imutavelmente anexada à cadeia, bem como uma notificação se a transação foi validada ou invalidada.

Tendo as devidas respostas, o Módulo de *blockchain* devolve a resposta para o aplicativo solicitante. Este fluxo permite que todos os acessos sejam transformados em transações, formando a trilha imutável para futuras auditorias.

4. Estudo de caso

Para aplicação da solução, foi desenvolvido um estudo de caso em um contexto IoT, onde o proprietário dos dados gerados pelo dispositivos, tem o interesse em compartilhar os mesmos com outros usuários e serviços. Porém o acesso a esses dados deve ser controlado e o sistema em questão deve prover uma trilha auditável dos usuários que acessaram os dados.

Para implementação foi criada uma aplicação *web* responsiva, fornecendo uma solução intuitiva para os usuários. Esta aplicação *web*, acessa uma API denominada *Gateway Service* API, responsável por acessar a infraestrutura de *blockchain* e a infraestrutura de nuvem. Também foi disponibilizado um ambiente configurado com o *Hyper Ledger Fabric*, onde foram implementados os *smart contracts*. Para tornar as funcionalidades dos *smart contracts* disponíveis fora do contexto do *Hyper Ledger*, foi criado o *smart contract service* API. Esta API é acessada unicamente pelo *Gateway Service* API, a figura 4 ilustra como está organizada a implementação.

O *Gateway Service* API foi integrado com o provedor de nuvem *Google Cloud*, através da disponibilização de credenciais. Os diretórios com os dados que deseja-se

compartilhar são cadastrados como *assets* na API do *blockchain*. Uma vez realizado esse procedimento de cadastro, o proprietário pode compartilhar seus *assets* com outros usuários.

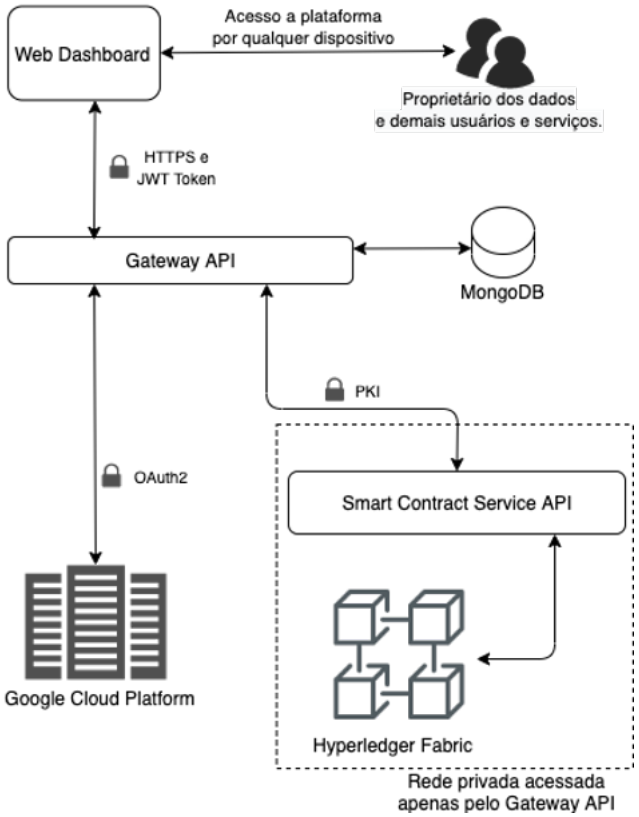

Fig. 4 - Implementação estudo de caso

O usuário que vai receber autorização para acessar os dados compartilhados, recebe um convite via e-mail para cadastro na plataforma. Uma vez feito o cadastro através do *web dashboard*, o usuário visualiza a lista de *assets* que foram compartilhados com ele. Ainda através do *web dashboard*, o usuário solicita acesso ao *asset* desejado. Nesse momento, o *Gateway Service* API submete uma transação para a API do *blockchain*. A transação é validada pelos *peers* e inserida no *ledger*. Uma vez que o retorno da transação é validado, o *Gateway Service* API acessa o provedor de nuvem *Google Cloud*, obtêm os dados referentes ao asset e disponibiliza uma página para *download*. Nas próximas subseções os componentes da implementação são descritos com mais detalhes.

5.1 Web Dashboard

O *Web Dashboard* implementa o Módulo de Interface da arquitetura. Ele é responsivo, permitindo sua usabilidade em qualquer dispositivo que possua um navegador, independente do tamanho de tela. O *dashboard* foi desenvolvido utilizando o *Angular Framework*, que permite a criação de *Single Page Applications* (SPA). Para estilização e usabilidade, foi utilizado o *Ionic Framework* que é um *framework* formado por uma série de componentes *mobile-first*, que permitem a criação de páginas com uma abordagem semelhante à utilizada em aplicativos mobile nativos.

O *Web Dashboard* oferece funcionalidades para os usuários, e possui dois perfis: proprietários dos dados e usuários convidados.

Para proprietários, é possível realizar o cadastro dos dispositivos em forma de *assets*. Os *assets* ficam disponíveis em uma lista, onde é possível acessar os detalhes do *asset* selecionado. Na página de detalhe, o proprietário dos dados é capaz de conceder ou revogar acesso a um terceiro, editar informações e acessar o histórico de operações realizadas. Essas operações incluem: qualquer tentativa de acesso do proprietário ou de um terceiro, edições nas informações ou permissões. Esta trilha auditável fica disponível em forma de linha do tempo contendo a data e hora em que a operação ocorreu.

Os usuários que receberam o compartilhamento, visualizam apenas os *assets* que foram compartilhados, não podendo realizar nenhuma operação de edição. Ao selecionar um *asset*, o usuário recebe o OTP (*One Time Password)*, e assim, consegue fazer o *download* do conteúdo. É importante ressaltar que cada acesso é único, e ao ser realizado, gera uma nova transação no *blockchain*, criando assim a trilha auditável dos acessos concedidos.

5.2 Gateway Service API

O Módulo de acesso a recursos foi implementado através do *Gateway Service API*. Esta API fornece acesso através da internet ao serviço de nuvem, onde os dados do Sistema IoT estão armazenados, e ao *blockchain*.

A API tem três responsabilidades principais: I) Estabelecer a conexão com o *smart contract service API;* II) Disponibilizar os dados armazenados no provedor de nuvem; e III) Expor *end-points* protegidos por login para aceitar requisições HTTP e executar as operações correspondentes.

Tendo em vista que as operações do *blockchain* são mais lentas, especialmente quando se aumenta o número de *peers*, não é boa prática utilizar essa estrutura, para armazenamento de todos os dados que tange o funcionamento de aplicações. Por esta razão é comum as soluções utilizarem bancos de dados convencionais como apoio ao *blockchain* para operações menos críticas ou que não necessitam das características disponibilizadas pelo *blockchain*.

Portanto, o API gateway, além da estrutura de *blockchain* para armazenamento, conta também com o *MongoDB*, que é um banco de dados não relacional, no *SQL* e orientado a documento. A decisão de utilizar um banco orientado a documento, se dá pela versatilidade que esta estrutura permite, considerando que a solução pode demandar a adição de novas propriedades não apenas nos metadados, mas também na adição de novas coleções de documentos. O objetivo deste banco de dados é armazenar dados que não se tem interesse em serem auditáveis ou criar uma trilha imutável. Portanto, são armazenados no *MongoDB* os dados de login de cada usuário, além dos metadados dos usuários ou *assets*.

5.3 Smart Contract Service API

Entre as plataformas privadas de *blockchain*, o *Hyper Ledger Fabric* [19] apresenta uma arquitetura de *blockchain* que fornece flexibilidade, escalabilidade e confidencialidade. Outra característica do *Fabric* é que sua arquitetura segrega o fluxo de transações em três estágios, cujas execuções podem ser realizadas por diferentes entidades: execução da transação e verificação de sua exatidão, ordenação das transações usando um algoritmo de consenso e validação de transações a partir do consenso da rede [20]. Desta forma, foi utilizado o *Hyper Ledger Fabric* como implementação do *blockchain*.

O *Smart Contract Service API* implementa o Módulo de *blockchain*. A infraestrutura do *blockchain* criada no contexto deste trabalho é privada, e além disso, não é capaz de se comunicar através do protocolo HTTP. Portanto, faz-se necessário a criação de uma API REST responsável por disponibilizar as funcionalidades dos *smart contracts*. O *Smart Contract Service API* é uma API REST responsável por disponibilizar as transações do *blockchain* via protocolo HTTP.

Através do SDK do *Hyper Ledger Fabric*, utiliza-se uma das APIs disponíveis para gerar uma proposta de transação. A proposta é uma solicitação que irá acionar um *smart contract* com os parâmetros de entrada, com a intenção de ler e/ou atualizar o *ledger*.

O SDK do *Hyper Ledger*, neste contexto, funciona como um invólucro para empacotar a proposta de transação no formato e arquitetura corretos (protocolo buffer sobre gRPC) e entregar também as credenciais criptográficas do usuário para produzir uma assinatura exclusiva para esta proposta de transação.

Embora a API disponibilize as funcionalidades do *blockchain*, por questões de segurança a mesma deve ficar em uma rede privada, não permitindo assim acesso externo, permitindo apenas que o API *gateway* tenha autorização para consumir os recursos.

A API foi desenvolvida com *NodeJS*, utilizando o Framework de roteamento *ExpressJS*, e utiliza os verbos HTTP para realizar as operações no *blockchain*. Esta integração é realizada através do SDK disponibilizado pelo framework *Hyper Ledger Fabric*.

5.3 Experimentos Realizados

Após popular o sistema com usuários e *assets*, foram realizados alguns experimentos para demonstrar o correto funcionamento da implementação.

No primeiro experimento, um usuário deu permissão de acesso aos seus *assets* para usuários terceiros. Em seguida, estes usuários verificaram a adição desses *assets* na lista de *assets* compartilhados com eles e conseguiram acessar os dados correspondentes no *Google Cloud*. Posteriormente, o usuário proprietário revogou o acesso a um *asset* a um usuário terceiro. Este terceiro não consegue mais acessar o recurso, tendo em vista que nenhum

dos *peers* valida esta transação de solicitação de acesso A API retorna à requisição como 403 *Forbidden*, antes mesmo de estabelecer uma conexão com o provedor de nuvem. Este código de resposta de status de erro HTTP 403 *Forbidden* indica que o servidor entendeu o pedido, porém se recusa a autorizá-lo.

O usuário proprietário de um *asset* consegue visualizar todas as operações realizadas no *asset* através de um trilha auditável armazenada no *blockchain*.

O segundo experimento teve como objetivo demonstrar a resiliência da implementação. O banco MongoDB, responsável pela autenticação da aplicação utilizada pelos usuários, foi corrompido, e nenhum dado no contexto do *blockchain* foi alterado. Sendo assim, após a restauração do serviço de autenticação e do MongoDB, cada usuário deve se cadastrar novamente no sistema com o mesmo login e senha. Em seguida, a aplicação volta a seu estado anterior, apresentando os dados dos *assets*, bem como seus respectivos históricos que são restaurados a partir do *blockchain*. Isto demonstra que a solução é robusta, tendo em vista que os dados persistidos no *blockchain* são resilientes por se tratar de um ambiente distribuído e não centralizado.

6. Conclusão

Neste artigo, foi proposta uma arquitetura que controla o acesso a dados armazenados através de serviços de terceiros oferecendo rastreabilidade desses acessos aos proprietários dos dados. A arquitetura propõe o uso de *blockchain* para registrar as informações sobre os acessos realizados, tendo em vista sua característica de imutabilidade.

O estudo de caso permitiu colocar em prática a arquitetura proposta em um cenário com integração real com um provedor de Nuvem e uma estrutura *blockchain*. A implementação permitiu demonstrar, de forma simplificada, como os usuários são capazes de cadastrar seus *assets*, permitir ou revogar acesso. Além disso, o usuário tem permissão de acessar o histórico das operações realizadas nos *assets* que é proprietário, demonstrando a auditabilidade proposta.

Além disso, a implementação também demonstrou a tolerância a falhas da arquitetura e sua capacidade de recuperação.

Finalmente, a revisão bibliográfica revelou que a maioria dos trabalhos disponíveis utiliza *Ethereum* como implementação de redes *blockchains* públicas. Esta escolha se dá pela quantidade de material disponível para desenvolvimento, quando comparado com o *Hyper Ledger Fabric*, cuja versão 2.0 é recente. Entretanto, o uso de *blockchain* público envolve custos adicionais, devido à inserção da figura do minerador, resultando em vários desafios e limitações. Por outro lado o *Hyper Ledger Fabric*, permite a criação de *blockchains* conforme a disponibilização de *hardware*, pois quem controla os *peers* que compõem a rede *blockchain* é o administrador da rede.

Como resultado adicional, este trabalho demonstra que o uso do *Hyper Ledger Fabric*, é uma alternativa interessante, pois além de permitir a utilização de uma rede completamente conhecida, também fornece segurança dos dados e resiliência da rede. Com isto, o uso deste tipo de abordagem pode ser incentivado para auxiliar ainda mais os pesquisadores e desenvolvedores no campo da computação distribuída e nuvem, com o objetivo de mitigar o acesso de invasores a sistemas através da rede, tendo em vista a dificuldade inserida pelo *blockchain* para se realizar invasões.

Referências

[1] BUYYA, R., YEO, C. S., VENUGOPAL, S., BROBERG, J., & BRANDIC, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. Future Generation computer systems, 25(6), 599-616.

[2] BACHIEGA, N. G. (2014). Algoritmo de escalonamento de instância de máquina virtual na computação em nuvem.

[3] PINHEIRO, A., CANEDO, E. D., ALBUQUERQUE, R. D. O., & DE SOUSA JÚNIOR, R. T. (2021). Validation of Architecture Effectiveness for the Continuous Monitoring of File Integrity Stored in the Cloud Using Blockchain and Smart Contracts. Sensors, 21(13), 4440.
[4] PINHEIRO, A., CANEDO, E. D., DE SOUSA, R. T., & ALBUQUERQUE, R. D. O. (2020). Monitoring File Integrity Using Blockchain and Smart Contracts. IEEE Access, 8, 198548-198579.
[5] KOTHA, S. K., RANI, M. S., SUBEDI, B., CHUNDURU, A., KARROTHU, A., NEUPANE, B., & SATHISHKUMAR, V. E. (2021) A comprehensive review on secure data sharing in cloud environment. Wireless Personal Communications, 1-28.
[6] ZHANG, Y., XU, C., LIN, X., & SHEN, X. S. (2019).Blockchain-based public integrity verification for cloud storage against procrastinating auditors. IEEE Transactions on Cloud Computing.
[7] AYOADE, G., KARANDE, V., KHAN, L., & HAMLEN, K. (2018, JULY). Decentralized IoT data management using blockchain and trusted execution environment. In 2018 IEEE International Conference on Information Reuse and Integration (IRI) (pp. 15-22). IEEE.
[8] LIANG, X., SHETTY, S., TOSH, D., KAMHOUA, C., KWIAT, K., & NJILLA, L. (2017, May). Provchain: A blockchain-based data provenance architecture in cloud environment with enhanced privacy and availability. In 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID) (pp. 468-477). IEEE.
[9] BISWAS, K., & MUTHUKKUMARASAMY, V. (2016, December).Securing smart cities using blockchain technology. In 2016 IEEE 18th international conference on high performance computing and communications; IEEE 14th international conference on smart city; IEEE 2nd international conference on data science and systems (HPCC/SmartCity/DSS) (pp. 1392-1393). IEEE.
[10] DORRI, A., KANHERE, S. S., & JURDAK, R. (2016). Blockchain in internet of things: challenges and solutions. arXiv preprint arXiv:1608.05187.
[11] DORRI, A., KANHERE, S. S., JURDAK, R., & GAURAVARAM, P. (2017, March). Blockchain for IoT security and privacy: The case study of a smart home. In 2017 IEEE international conference on pervasive computing and communications workshops (PerCom workshops) (pp. 618-623). IEEE.
[12] HUH, S., CHO, S., & KIM, S. (2017, February).Managing IoT devices using blockchain platform. In 2017 19th international conference on advanced communication technology (ICACT) (pp. 464-467). IEEE.
[13] LEE, C. H., & KIM, K. H. (2018, January). Implementation of IoT system using block chain with authentication and data protection. In 2018 International Conference on Information Networking (ICOIN) (pp. 936-940).
[14] IEEE.NAKAMOTO, S. (2008). Bitcoin: A peer-to-peer electronic cash system. Decentralized Business Review, 21260.
[15] CONG, L. W., & HE, Z. (2019). Blockchain disruption and smart contracts. The Review of Financial Studies, 32(5), 1754-1797.
[16] CHRISTIDIS, K., & DEVETSIKIOTIS, M. (2016).Blockchains and smart contracts for the internet of things. Ieee Access, 4, 2292-2303.
[17] ALHARBY, M., & VAN MOORSEL, A. (2017). Blockchain-based smart contracts: A systematic mapping study. arXiv preprint arXiv:1710.06372.
[18] WÜST, K., & GERVAIS, A. (2018, June). Do you need a blockchain?. In 2018 Crypto Valley Conference on Blockchain Technology (CVCBT) (pp. 45-54). IEEE.
[19] SAJANA, P., SINDHU, M., & SETHUMADHAVAN, M. (2018). On blockchain applications: HyperLedger fabric and ethereum. International Journal of Pure and Applied Mathematics, 118(18), 2965-2970.
[20] ANDROULAKI, E., BARGER, A., BORTNIKOV, V., CACHIN, C., CHRISTIDIS, K., DE CARO, A., ... & YELLICK, J.(2018, April). HyperLedger fabric: a distributed operating system for permissioned blockchains. In Proceedings of the thirteenth EuroSys conference (pp. 1-15).