Controle de acesso a serviços usando Smart Contracts

Marcos Vinicius M. R. de Souza[1], **Hélio Crestana Guardia**[1]

[1]Depto. de Computação – Universidade Federal de São Carlos (DC-UFSCar)
São Carlos – SP – Brazil

marcos.souza@dc.ufscar.br, helio@dc.ufscar.br

Abstract. *Advances in ubiquitous computing and in the Internet of things (IoT) bring new challenges to access control systems, such as not having to rely on a centralized trusted party, and the ability to use contextual data. Such characteristics may be required to enable IoT services in smart homes and cities, and in electronic health systems. In these environments, service-based device to device interactions reinforce the need for flexible and scalable authenticity and confidentiality mechanisms. This work presents an access control model to address such challenges using a blockchain infrastructure. In the proposed model, services and contextual data are defined using smart contracts, and blockchain logs are used for reliable information management. Assets are associated with permissions, and their transfers prove access control rights in service invocation requests. A reference implementation on Hyperledger shows the viability of the model.*

Resumo. *Avanços na computação ubíqua e na Internet das Coisas (IoT) trazem novos desafios aos sistemas de controle de acesso, como a não dependência de unidades centrais seguras, e a necessidade de identificar dados de contexto. Estas características podem ser necessárias para serviços em IoT, como em casas e cidades inteligentes e em sistemas eletrônicos de saúde. Nesses ambientes, autenticidade e confidencialidade devem ser providas de forma flexível e escalável. Este trabalho apresenta um modelo de controle de acesso que atende estes desafios baseando-se na infraestrutura de blockchain. Nesse modelo, serviços e dados contextuais são definidos via smart contracts e registros na blockchain provêem gerenciamento seguro de informações. Assets são associados às permissões e suas transferências provam direitos de acesso a serviços. Uma implementação de referência do modelo sobre Hyperledger mostra sua viabilidade.*

1. Introdução

O acesso à informação comumente deve ser controlado, a fim de garantir privacidade e segurança. Diversos modelos podem ser adotados para definir o controle de acesso e suas políticas e regras, destacando-se o RBAC (*Role-Based Access Control*), baseado em papéis, e o ABAC (*Attribute-Based Access Control*), baseado em atributos.

RBAC [Ferraiolo e Kuhn 1992] baseia-se na relação entre papéis assumidos por um usuário e os recursos que ele pode acessar. Esse modelo se enquadra em situações em que as permissões de acesso são definidas pelo cargo ocupado pelo usuário. Já no ABAC [Hu et al. 2014], o acesso é concedido àqueles usuários que possuírem os atributos necessários para acessar cada recurso. Esse modelo propicia uma granularidade mais fina nas regras de acesso, ao custo de maior complexidade de gerenciamento das regras.

Apesar de o modelo ABAC permitir uma granularidade mais fina, sua complexidade de implementação e manutenção dificulta sua adoção. Assim, o modelo RBAC é mais adotado, incluindo extensões para atender necessidades adicionais de controles de acessos. A sensibilidade ao contexto (*context-awareness*), por exemplo, é considerada essencial na computação ubíqua [Kuhn, Coyne e Weil 2010] e extensões no modelo RBAC buscam viabilizar sua adoção.

Em sistemas computacionais, contexto é definido como qualquer informação que possa ser usada para caracterizar estado, situação, ou atividade de uma entidade, ou ainda do ambiente no qual a entidade opera [Dey 2001, Toninelli et al. 2006]. Informações de contexto podem incluir, entre outros aspectos, tempo, localização, condições gerais do ambiente, intenções do usuário, estado da rede e estado de execução. Os modelos de controle de acesso que utilizam tais informações de contexto são classificados como CAAC (*Context Aware Access Control*).

O desenvolvimento das tecnologias mobile, da internet das coisas e da computação ubíqua de forma geral, gerou um aumento na quantidade de formas e locais possíveis para acessar informações, consequentemente, aumentando a variedade de situações nas quais as informações serão acessadas. No entanto, uma informação não estará segura se não for visualizada apenas pela pessoa certa, no momento certo e nas circunstâncias certas.

A preocupação em tratar variáveis de contexto no controle de acesso, trouxe várias propostas sobre o assunto. Exemplos disso são trabalhos como [Cleeff, Pieters e Wieringa 2010], com um estudo sobre LBAC (*Location-Based Access Control*), e [Bertino e Kirkpatrick 2011] com os GEO-RBAC (*Geo-spatial RBAC*). Ambos trabalhos mostram que modelos de controle de acesso com uso de localização têm sido usados para melhorar a segurança em TI. Ao estabelecer localizações específicas para atuação de sistemas e usuários, atacantes terão mais dificuldade em comprometer um sistema.

[Cleeff, Pieters e Wieringa 2010] ainda apresentam metas para os LBAC, sendo: o princípio da separação de deveres (*principle of separation of duty*), princípio do privilégio mínimo (*principle of least privilege*), capacidade de responsabilização (*accountability*), capacidade de manutenção (*maintainability*) e usabilidade (*usability*). No entanto, tais metas podem ser visadas por outros modelos de controle de acesso.

O avanço das tecnologias com dispositivos móveis ainda trouxe mais preocupações, como a discutida por [Gupta, Kirkpatrick e Bertino 2012] que ressaltam que a localização relativa de um usuário pode ser mais importante do que sua localização absoluta, pois a posição do usuário em relação à outras pessoas é uma potencial ameaça a segurança das informações.

Com a mesma preocupação com localização relativa, [Choi et al. 2013] propõem um modelo de controle de acesso baseado em proximidade. Chaves e assinaturas parciais são propagadas por dispositivos Bluetooth e o leitor precisa de todas essas chaves parciais e da composição das assinaturas para obter acesso. Para isto, ele deve estar na área determinada como confiável, que é delimitada pelos dispositivos.

Novas abordagens para controle de acesso também têm incluído o uso de blockchain, como visto em [Zyskind, Nathan e Pentland 2015], que apresenta uma forma de administrar o controle de acesso sem a necessidade de confiança em terceiros. Para tanto, implementam um serviço que armazena de maneira distribuída dados assinados e, even-

tualmente, cifrados. MedRec [Azaria et al. 2016] e FairAccess [Ouaddah, Elkalam e Ouahman 2016] são exemplos de frameworks desenvolvidos para modelos de controle de acesso baseados em blockchain.

Essas abordagens visam a aproveitar princípios e práticas já testados no ambiente das criptomoedas (*crypto currencies*), trazendo para o controle de acesso a possibilidade de uma arquitetura de sistema descentralizada, distribuída, auditável, transparente e robusta, incrementando a segurança e privacidade. Essas características tornam o uso das tecnologias de blockchain promissor nos ambientes de Internet das Coisas (*IoT - Internet of Things*), com aplicação em serviços de cuidados de saúde e *Smart Cities* [Hashemi et al. 2016], por exemplo.

Esse trabalho apresenta uma nova proposta de uso da tecnologia blockchain para prover controle de acesso na interação entre dispositivos usando invocações de serviço. De maneira geral, a abordagem proposta procura agregar características da rede como rastreabilidade, eliminação do terceiro confiável e imutabilidade. Na seção 2 apresentam-se considerações sobre blockchains e alguns sistemas que utilizam essa infraestrutura para controle de acesso. A seção 3 apresenta um novo modelo de controle de acesso a serviços baseado em blockchains. Já na seção 4 são descritos aspectos da implementação do modelo proposto e, na seção 5 são discutidos resultados da aplicação do modelo. Por fim, na seção 6 são apresentadas as conclusões sobre a proposta e alguns aspectos de trabalhos futuros.

2. Controle de acesso e blockchain

A estrutura de blockchain foi primeiramente apresentada como um registro público e descentralizado para transações financeiras com o Bitcoin [Nakamoto 2008]. A blockchain apresentada com o Bitcoin utiliza um conceito de transações focadas na transferência de valores entre contas. Outras plataformas blockchain expandiram o conceito de transações de acordo com suas aplicações. Ethereum [Wood 2014], por exemplo, apresentou uma plataforma blockchain mais generalizada, expandindo as transações para contratos complexos que passaram a ser chamados de *smart contracts*. Dessa forma, transações passam a ter descrições algorítmicas (*smart contract*), representando operações computacionais que serão feitas no blockchain [Wood 2014, English, Auer e Domingue 2016]. *Smart contracts* passaram a ser implementados em outras plataformas blockchain, como no Hyperledger Fabric do projeto Hyperledger, onde são chamados de *chaincodes* [Cachin et al. 2016].

Ao criar uma arquitetura descentralizada, distribuída, auditável, transparente e robusta, com potencial para prover elevada segurança e privacidade, a estrutura de blockchain passou a atrair interesses em outras áreas de aplicação, como no gerenciamento e controle de acesso de informações.

[Zyskind, Nathan e Pentland 2015], implementam um protocolo que transforma um blockchain em um gerenciador de controle de acesso automatizado que não requer confiança em terceiros. Diferente do Bitcoin, as transações no sistema não são financeiras, mas são usadas para carregar instruções, como armazenar, consultar e compartilhar dados. Os autores discutem ainda possíveis futuras extensões ao blockchain para aproveitá-lo como solução para problemas de confiança em computação na sociedade.

[Azaria et al. 2016] apresentam MedRec, um sistema de gerenciamento de regis-

tros descentralizado para manusear Registros Médicos Eletrônicos (RMEs), usando tecnologia blockchain. A estratégia aproveita propriedades únicas de blockchain, fazendo MedRec gerenciar autenticação, confidencialidade, capacidade de responsabilização e compartilhamento de dados. MedRec ainda possui uma estrutura modular que o integra com provedores existentes e com soluções de armazenamento local de dados.

[Azaria et al. 2016] ainda incentivam as partes médicas interessadas a participar da rede como "mineradores" de blockchain, e que obtenham em troca acesso aos dados, agregados e anônimos, como recompensa pela mineração, por sustentar e assegurar a rede por prova de trabalho (*Proof of Work*). Incentivam também que pacientes e provedores liberarem esses metadados anonimamente.

[Ouaddah, Elkalam e Ouahman 2016] fornecem um modelo de referência para um framework proposto para sistemas *IoT*, com Objetivos, Modelos, Arquitetura e Mecanismo (camadas OM-AM). Assim, apresentam o FairAccess, um framework de gerenciamento de autorizações descentralizado, com pseudônimos e preservação da privacidade, que permite aos usuários possuírem e controlarem seus dados. Então, adaptam o blockchain como um gerenciador de controle de acesso descentralizado. Diferente das transações financeiras do bitcoin, o FairAccess apresenta novos tipos de transações que são usadas para conceder, receber, delegar e revogar acesso.

Considerando as características de uma infraestrutura de blockchain, contudo, novos mecanismos de segurança para controle de acesso podem ser providos sobre essa plataforma, potencialmente combinando consenso e consistência dos dados, dinamismo de geração de informações para tratamento de aspectos de localização e flexibilidade no gerenciamento de múltiplos recursos lógicos.

3. Modelos para controle de acesso baseado em blockchain

No cenário de computação distribuída considerado neste trabalho, investiga-se a comunicação via solicitações de serviço entre dispositivos que queiram compartilhar recursos perante certas condições e outros que queiram ter acesso a estes recursos e que, para tal, precisam satisfazer as condições estabelecidas.

Tendo em vista as flexibilidades do modelo de controle de acesso baseado em papéis e usando dados de contexto em cenários de *IoT*, ou em qualquer situação de interação direta entre dispositivos, bem como o modelo de autenticação distribuída provido pelos sistemas de blockchain, esse trabalho propõe a criação de um sistema de autenticação que integre essas plataformas.

Neste cenário, adotou-se o uso de uma plataforma blockchain por sua capacidade de estabelecer uma relação de confiança distribuída, entre nós que não possuem confiança pré-estabelecida entre si, nem com uma terceira parte centralizada. Considerando sistemas como o Hyperledger e o Ethereum, uma infraestrutura de blockchain também permite registrar serviços e operações que se deseja realizar de forma verificável, e que podem ser criados, armazenados e acessados de maneira distribuída.

Para tratar dados de contexto e a autenticação de forma distribuída, propõe-se o uso de smart contracts para a especificação de códigos dos serviços de forma confiável entre múltiplos pares, sem o uso de uma entidade central confiável. As operações previamente indicadas nos smart contracts e solicitadas pelos participantes são armazenadas no

blockchain e podem ser verificadas. As verificações sobre as operações para registro e alteração do estado atual da rede são feitas por meio de consenso entre os nós participantes.

Blockchain é utilizado para o registro de recursos, de políticas de acesso, de dados de contextos e de transações, bem como para obtenção de acesso, atuando também como registro (*logging*) auditável das operações. Para o desenvolvimento de modelos de controle de acesso neste trabalho foi utilizada a plataforma blockchain Hyperledger Fabric, pela sua versatilidade na construção da rede blockchain e na execução de transações, sua não dependência de criptomoedas e seu suporte para criação e execução consensual de *smart contracts*.

3.1. Hyperledger Fabric

Hyperledger Fabric é um dos projetos Hyperledger com desenvolvimento hospedado pela *The Linux Foundation*. Trata-se de uma implementação de estrutura blockchain com a finalidade de atuar como base para aplicações, com funcionalidades como consenso na execução de códigos e serviços de associação entre membros, aos moldes de tecnologia *plug-and-play* (conectar e usar).

No Hyperledger Fabric, os *smart contracts* são chamados de *chaincodes* e são utilizados para compor toda a lógica da aplicação do sistema e suas transações. Suas especificações podem ser implementadas como códigos executáveis com funções para manipular os dados de entrada, o estado e os registros na rede.

As transações são as operações sobre a rede. No Hyperledger, toda transação tem uma entrada com dados novos e/ou dados já registrados, e resulta na criação, leitura, ou atualização de informações para registrar na blockchain, ou ainda apagá-las. As transações são definidas como funções contidas em *chaincodes*. A real aplicação das transações só ocorre caso haja consenso entre os *peers* de endosso, definidos na criação do *chaincode*, quanto ao resultado da execução.

A rede é composta por nós lógicos nos papéis de **cliente**, **endorsing peer** e **ordering peer**. Nós atuando como clientes são responsáveis por criar propostas de execução de transações, aguardar assinaturas de endosso e propagar ao *ordering service* as transações assinadas. *Endorsing peers* ficam em execução permanente, recebendo propostas de transação e simulando a execução dessas propostas, assinando propostas endossadas, realizando *commits* e mantendo seus *PeerLedgers*. *Ordering peers* são responsáveis pelo *ordering service*; eles garantem a ordem cronológica das transações, coletam propostas assinadas, verificam o endosso e redistribuem as transações válidas, ordenando *commits*.

Com o sistema Hyperledger, trabalha-se com os conceitos de *Participant* e *Asset*. *Participants* são os atores que interagem com a rede, usados para definir as classes dos participantes na comunicação. *Assets* são os objetos utilizados na interação dos *participants* com as transações, que podem ser criados, lidos, atualizados ou deletados. Neste trabalho, utilizam-se os termos **requerente** (*Requester*) e **proprietário de recurso** (*ResourceOwner*) para identificar os *Participants* do sistema de acesso.

Usando a infraestrutura do sistema Hyperledger, este trabalho apresenta duas estratégias para o gerenciamento de permissões, uma baseada em *smart contracts* e outra que utiliza *assets* como indicadores de permissões. Em ambos os casos, as permissões são verificadas em invocações de serviço (*smart contracts / chaincodes*).

3.2. CAAC usando smart contracts

Na primeira proposta de controle de acesso, *assets* são definidos para representar os recursos (*Resource*), as políticas de acesso (*AccessPolicy*), dados de contexto (*Context*) e permissões de acessos (*Access*).

O modelo aqui proposto trata interação entre nós numa rede via invocação de serviço. Os usuários de serviços são os usuários da rede blockchain, enquanto os prestadores de serviços são os *peers* que executam as operações (*chaincodes*). As transações definem os requisitos necessários e as regras de execução do serviço (também definidos nos *chaincodes*). Em caso comum neste modelo, um usuário de serviço solicitando acesso deve prover seus parâmetros e informações (dados de contexto) ao fazer a solicitação. Caso os parâmetros do requerente atendam os requisitos definidos (políticas de acesso) o serviço é executado e retorna os dados apropriados.

Três transações foram definidas como base para implementar esse modelo de controle de acesso: *AddResource*, *ComposeContext* e *RequestAccess*.

AddResource é a transação pela qual um Usuário (*ResourceOwner*) registra um recurso ao qual pretende conceder acesso perante determinadas condições. As entradas são as informações do recurso e as definições da política de acesso, gerando registro de um objeto *Resource* com uma política de acesso (*AcessPolicy*) vinculada.

ComposeContext é a transação que um requerente deve fazer para registrar suas atuais variáveis de contexto. Esta transação tem as informações do requerente como entrada e gera um objeto do tipo *Context* que é tanto usado nas requisições de acesso, quanto permanece registrado para futuras verificações e auditorias.

RequestAccess é a transação em que um requerente solicita o acesso a um determinado recurso. O requerente provê o *Asset* contendo seus dados de contexto validados e o recurso que pretende acessar como entrada para a transação. As verificações se o requerente satisfaz as regras de acesso são definidas nas operações dos *smart contracts* (*chaincodes*). O código da transação verifica se o contexto atual do requerente satisfaz as condições estabelecidas na política de acesso do recurso. Se as condições não são satisfeitas, nega o acesso. Caso as condições forem satisfeitas, a transação emite uma permissão de acesso para o requerente e cria um objeto *Access* para manter o registro daquela ação. O acesso ocorre pela verificação da existência desse objeto *Access* e o requerente tem permissão (*READ* e *UPDATE*) até que o objeto seja removido.

3.3. CAAC com smart contracts baseado em transferências de recursos

Na segunda proposta de uso de blockchain para controle de acesso, assets são associados às permissões para execução de serviços registrados na rede pelos nós / dispositivos servidores.

No Hyperledger Fabric, existe um tipo de transação que são as *deploy transactions*, que permitem a criação dinâmica de novos *smart contracts* (*chaincodes*). Isso possibilita a criação de um sistema de controle de acesso dinâmico, já que novas lógicas de operação podem ser incluídas por esses *chaincodes*.

Um modelo de controle de acesso que varie com o tempo, traz a necessidade de adaptação no tratamento das informações de contexto, já que novas informações de

contexto podem ser modeladas e pode haver novas lógicas de validação. Assim, propõe-se que a posse de recursos (*Assets*) equivalha à satisfação dos requisitos de permissão necessários para solicitar a execução de um serviço.

Para isso, tanto é possível que um *Asset* único, represente todas informações de contexto de um cliente, quanto utilizar múltiplos *Assets* para representar um contexto, cada um referindo-se à prova de uma informação de contexto diferente. Ao representar diferentes informações de contexto com *Assets* independentes, torna-se possível a introdução de modelagens de novas variáveis de contexto já durante a operação da rede.

Neste modelo, ao invés de um *Requester* possuir um *Asset* do tipo *Context*, obrigatório para provar seu contexto, esse *Requester* passa a ser proprietário (*owner*) de vários *Assets*, cada um comprovando uma informação de contexto diferente. A associação de permissões com *Assets* deve ser realizada pelos proprietários de recursos compartilhados na forma de serviço. Diferentes (infinitos) *assets* podem ser criados e gerenciados por uma infraestrutura de blockchain, assim como há mecanismos para transferência desses *assets* de forma válida e auditável entre os participantes.

Nesse sentido, apresenta-se a ideia de um sistema com chaincodes adicionados em tempo de execução e *Assets* utilizados como "moeda de troca" a ser gasta diretamente na obtenção de uma informação (ou invocação de qualquer serviço / *chaincode*).

Neste cenário, um usuário recebe *Asset(s)* que prova(m) que ele está em um contexto, ou satisfaz requisito(s) de contexto e, portanto, tem as permissões necessárias para a execução do serviço. Diferentes mecanismos, inclusive externos, podem ser utilizados para a atribuição desses *Assets* de forma confiável. Assim, a transação de requisição requer que a posse desse(s) *Asset(s)* tenha sido transferida para o provedor do serviço desejado e que o registro dessa transferência na blockchain seja fornecido como parâmetro da invocação do serviço. Essa transferência indica que o cliente do serviço satisfaz as condições necessárias, ou seja, tem as permissões requeridas. Esta alternativa visa evitar buscas constantes por permissões salvas e o reúso indevido destas permissões.

Nesta abordagem, o proprietário de um recurso implementa um *chaincode* especificando como as informações do seu recurso serão acessadas, como o contexto do requerente será avaliado para concessão do *Asset* necessário para o acesso, e como esse *Asset* será gasto na requisição do acesso. Todos esses requisitos, validáveis pelos mecanismos de transferências de *Assets*, passam a ser embutidos numa avaliação lógica executada na parte inicial do código de cada *chaincode* de serviço.

Resumidamente, antes de executar a lógica associada a um serviço, é preciso que o código de um chaincode inclua a verificação da transferência dos *Assets* necessários, do solicitante para o dono do serviço.

Os identificadores das transferências realizadas, e devidamente salvas nas cadeias de informação do Hyperledger, são passados pelo cliente na solicitação de execução do serviço desejado. Retomando as operações do servidor, vê-se que ele será capaz de validar que os *Assets* foram transferidos e pode prestar o serviço.

4. Implementação

Na operação dos dois modelos de controle de acesso propostos neste trabalho, o seguinte cenário é previsto:

- Há um conjunto de nós interligados em rede, disponíveis para execução das atividades de manutenção da blockchain (neste caso, do Hyperledger): dois ou mais *endorsing peers* para execução do serviço e ao menos um *ordering server*;
- Um nó que deseja participar da rede provendo serviço deve registrar-se na rede, criando uma identidade e registrando seus serviços em algum "*channel*" do Hyperledger;
- Um cliente que deseja usar qualquer serviço provido na rede também deve criar uma identidade e conectar-se a um "*channel*" do Hyperledger.

Para o modelo de CAAC usando *smart contracts*, as operações de clientes e servidores são as seguintes:

- Ao cadastrar um recurso, um servidor deve cadastrar uma política de acesso vinculada, que deverá ser cumprida para geração da permissão.
- Um cliente que deseja executar um serviço deve cadastrar seu contexto e solicitar a permissão. Isso é feito através do serviço *RequestAccess*, que verifica se o contexto do cliente satisfaz a política de acesso do serviço, e gera um registro de permissão. Esse registro será verificado para a realização do acesso.

De maneira geral, o fluxo de controle de acesso a um serviço é definido de acordo com a Figura 1.

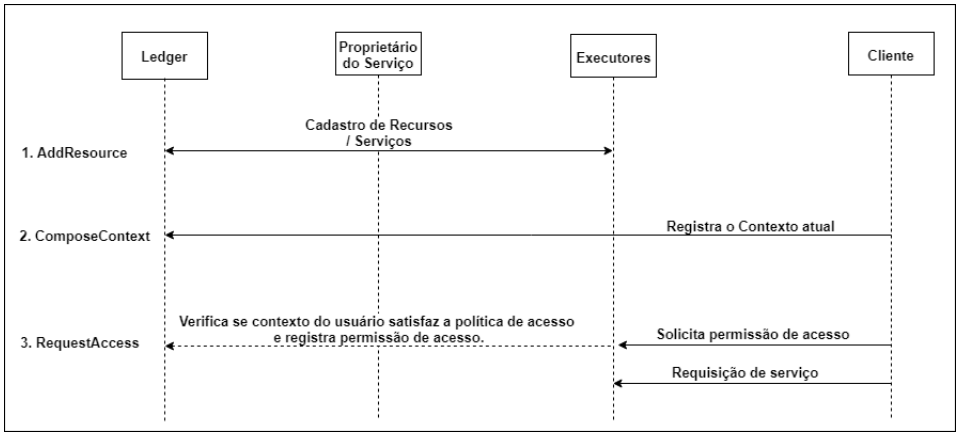

Figura 1. Fluxo de operações no CAAC com *smart contracts*.

Para o modelo CAAC com *smart contracts* baseado em transferências de recursos, as operações de clientes e servidores são as seguintes:

- Na fase inicial de qualquer serviço provido deve existir uma verificação de requisitos que correspondem a transferência(s) de *Asset(s)*.
- Um cliente que deseja um serviço deve satisfazer os critérios que levam ao recebimentos dos Assets necessários. Assim, solicitar permissão significa obter os *Assets* apropriados.
- Para solicitar um serviço, um cliente transfere os *Assets* associados aos privilégios necessários para o dono do serviço ou entidade responsável.

De maneira geral, o fluxo de controle de acesso a um serviço é definido de acordo com a Figura 2.

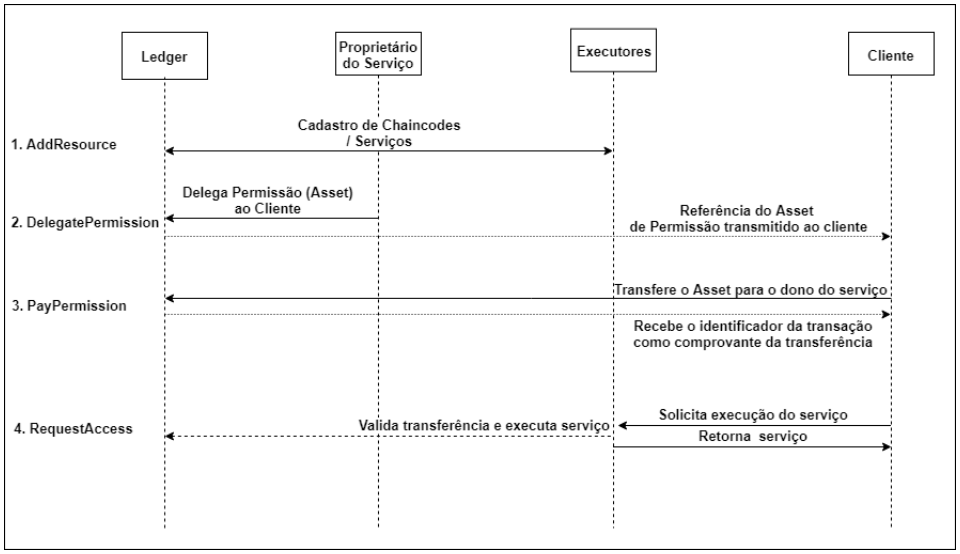

Figura 2. Fluxo de operações no CAAC com *smart contracts* baseado em transferências de recursos.

Vale observar que os códigos executados pelo Hyperledger são todos implementados como *chaincodes*. Assim, há os *chaincodes* de serviço, associados às operações de transferências e criações de *Assets*, por exemplo, disponíveis na biblioteca / API do Hyperledger, ou criados pela aplicação que vai usar essa infraestrutura. Usando os serviços do Hyperledger também é possível criar dinamicamente novos chaincodes, que serão os serviços providos por nós participantes do sistema e cujas execuções serão solicitadas pelos clientes que satisfizerem os requisitos de permissão.

Para demonstrar a implementação dos *chaincodes* que viabilizam o modelo de controle de acesso proposto, foram criadas funções em *JavaScript* utilizando a ferramenta Hyperledger Composer, que determinam o funcionamento das etapas de comunicação.

Para viabilizar os modos de controle de acesso previstos, serviços específicos (*chaincodes*) também foram desenvolvidos na plataforma Hyperledger.

No modelo CAAC com *smart contracts*, os *chaincodes* são:

- **addResource**: cadastra um novo recurso e uma política de acesso associada a ele, como no Algoritmo 1.
- **composeContext**: cadastra as informações de contexto do usuário, como no Algoritmo 2.
- **requestAccess**: verifica se o contexto do usuário satisfaz a política de acesso do recurso e gera o registro da permissão, como no Algoritmo 3.

Para o modelo CAAC com smart contracts baseado em transferências de recursos foram criados os seguintes chaincodes:

- **addResource**: cadastra um novo serviço, como no Algoritmo 4.
- **delegatePermission**: gera *Asset* de permissão para um cliente, como no Algoritmo 5.
- **payPermission**: cliente transfere *Asset* de permissão para o proprietário ou gestor do serviço, como no Algoritmo 6.
- **requestAccess**: verifica se as permissões foram corretamente transferidas e invoca serviço, como no Algoritmo 7.

Algoritmo 1: AddResource

Entrada: $resourceId, address, accessPolicy, owner$
Saída: registro de $Resource$ e atualização de $ResourceOwner$

início
  Cria $Resource$ com $resourceId$;
  $Resource.addrees = address$;
  $Resource.accessPolicy = accessPolicy$;
  $Resource.owner = owner$;
  $owner.resources.push(Resource)$;
fim
retorna $add(Resource)$ *e* $update(owner)$

Algoritmo 2: ComposeContext

Entrada: $contextId, role, location, proximity, ..., owner$
Saída: registro de $Context$ e atualização de $Requester$

início
  Cria $Context$ com $contextId$;
  $Context.role = role$;
  $Context.location = location$;
  $Context.proximity = proximity$;
  ... `// Quaisquer variáveis de contexto definidas`
  $Context.owner = owner$;
fim
retorna $add(Context)$ *e* $update(owner)$

Algoritmo 3: RequestAccess

Entrada: $Resource, accessId, Requester$
Saída: registro de $Access$ e atualização de $Requester$

início
  if $Requester.Context$ *satisfaz* $Resource.accessPolicy$ **then**
    Cria $Access$ com $accessId$;
    $Access.Resource = Resource$;
    $Access.permissionedRequester = Requester$;
    $owner.accesses.push(Access)$;
fim
retorna $add(Access)$ *e* $update(owner)$

Algoritmo 4: AddResource

Entrada: $resourceId, address, owner$
Saída: registro de $Resource$ e atualização de $ResourceOwner$

início
  Cria $Resource$ com $resourceId$;
  $Resource.addrees = address$;
  $Resource.owner = owner$;
  $owner.resources.push(Resource)$;
fim
retorna $add(Resource)$ *e* $update(owner)$

Algoritmo 5: DelegatePermission

Entrada: $permissionId, resourceId, requesterId$
Saída: registro de $Permission$

início
  Cria $Permission$ com $permissionId$;
  $Permission.resourceId = resourceId$;
  $Permission.permissionedRequester = requesterId$;
  $Permission.owner = requesterId$;
fim
retorna $add(Permission)$

Algoritmo 6: PayPermission

Entrada: $permissionId, provider$
Saída: atualização de $Permission$ e $Requester$
início
    $Permission.owner = provider$;
    $Permission.permissionedRequester.recibo = transactionId$;
fim
retorna $update(Permission)$ *e* $update(Permission.permissionedRequester)$

Algoritmo 7: RequestAccess

Entrada: $resourceId, Requester$
Saída: JSON de resposta do serviço
início
    $tranx$ = busca('selectPayment', $recibo : Requester.recibo, resourceId : resourceId,$
      $userId : Requester.userId$);
    if *tranx* **then**
        Executa serviço solicitado $Resource.resourceId$;
        Gera $response$ com dados de resposta do serviço;
        Formata $response$ como JSON;
fim
retorna $response$

Na operação dos chaincodes de serviço criados, diferentes serviços do Hyperledger foram utilizados:

- getFactory().newResource(parâmetro): cria um *Asset*.
- getFactory().newConcept(parâmetro): cria *Concept* (Estrutura conceitual que pode ser definida no modelo. Por exemplo, uma estrutura para JSON de retorno de serviço).
- getAssetRegistry(parâmetro): busca registro de Assets no *ledger*.
- getParticipantRegistry(parâmetro): busca registro de Participants no *ledger*.
- add(parâmetro): registra informação no *ledger*.
- update(parâmetro): atualiza valor de registro no *ledger*.
- query(parâmetro): usado para implementação de buscas específicas no *ledger*. Esse tipo de mecanismo foi particularmente utilizado na verificação da transferência de *Asset*, através do identificador da transação de transferência.

5. Resultados

As propostas de controle de acesso a serviços apresentadas neste trabalho são baseadas no mecanismo de *smart contracts*, na transferência de recursos e em aspectos de autenticação providos por uma infraestrutura de blockchain. Sobre essa plataforma, implementada sobre o sistema Hyperledger Fabric, provê-se controle de acesso sensível ao contexto em que se encontram os nós comunicantes, como pode ser necessário em cenários de *IoT*, por exemplo.

No primeiro modelo apresentado, buscou-se a criação de um modelo CAAC seguindo a mesma linha de desenvolvimento apresentada na bibliografia com os trabalhos relacionados. Portanto, utiliza-se uma estrutura baseada no modelo RBAC e adiciona-se novas restrições sobre as políticas e permissões de acesso de acordo com variáveis de contexto.

Este modelo foi apresentado utilizando-se um *Asset* que representa o recurso a ser acessado, com uma política de acesso que dita as variáveis de contexto para o acesso.

Outro *Asset* foi utilizado para representar o papel (*role*) do usuário e suas informações de contexto. Assim, utilizou-se um *smart contract* (*chaincode*) para verificar se o papel e as informações de contexto do usuário satisfazem as condições da política de acesso do recurso.

No segundo modelo apresentado, idealizou-se um mecanismo baseado em interação por serviços com controle de permissões de forma distribuída e segura. Tipicamente, num sistema distribuído utiliza-se: *broker*, para fazer os registros e resolução de endereços; provedor, para registrar e oferecer os serviços; e cliente, que realiza a descoberta dos serviços e provedores, via *broker*, e invoca os serviços diretamente aos provedores.

Levando isto em consideração, no modelo apresentado o *brokering* é feito a partir do *ledger* com os registros distribuídos; o provedor é representado pelos peers que cadastram *chaincodes* e serviços e que gerenciam as permissões, além dos *endorsing peers*; já os clientes são os *peers* que interagem com o *ledger* para validações e consultas, e interagem com os mecanismos de execução (*endorsing* e *ordering*).

A infraestrutura utilizada conta com o uso do Hyperledger e do conjunto de serviços que viabilizam o modelo de permissões elaborado. Assim, utiliza *Assets* como moeda de troca para obtenção de acesso. Um usuário recebe um *Asset* de permissão caso satisfaça as políticas de acesso e, posteriormente o utiliza, transferindo-o para o provedor, em troca do acesso.

Dessa forma, o modelo possui os seguintes benefícios:

- *brokering* distribuído no *ledger*;
- mecanismo de consenso (*endorsement*);
- mecanismo de permissões flexível, com quantos requisitos forem desejáveis por serviço, através dos *smart contracts* (*chaincodes*);
- permite a transferência de permissões como moeda de troca;
- evita reúso de permissões;
- pode usar validações externas e circunstanciais para dados de contexto;
- e garantia de autenticidade e consistência através das assinaturas e chaves criptoassimétricas.

6. Conclusões

A avaliação das propostas de controle de acesso com informação de contexto (CAAC) usando a plataforma Hyperledger mostrou a flexibilidade e simplicidade dos modelos. Utilizando o CAAC com *smart contracts*, a permissão de acesso é gerada a partir da verificação de um *Asset* de contexto em relação a uma política de acesso vinculada ao recurso. A permissão é gerada na forma de um registro na rede blockchain, que precisa ser verificado para obter o acesso.

Já no CAAC com *smart contracts* baseado em transferências de recursos, cada requisito para permissão é associado a um *Asset* a ser transferido ao dono do serviço como comprovação dos requisitos necessários. Informações de contexto, que podem ser validadas por elementos externos ao sistema, podem estar associadas a atribuição dos *Assets* necessários, inclusive de maneira dinâmica, como em situações que requerem presença e proximidade.

As execuções com o Hyperledger mostraram que a verificação dos requisitos de acesso podem estar embutidas na própria lógica dos serviços disponibilizados na rede.

Considerando que os mecanismos de verificação de autenticidade e garantia de confidencialidade do Hyperledger estão de acordo com a lógica definida nos sistemas de blockchain, as transações previstas nessa proposta atendem os requisitos de segurança envolvidos com as comunicações. Providas as autenticidades das informações transmitidas, as propostas de controle de acesso desenvolvidas podem ser facilmente implementadas e embutidas nas lógicas de quaisquer serviços que queiram utilizá-las. Para isso basta especificar requisitos como *Assets* e suas transferências como validações de seus cumprimentos.

Os mecanismos de consenso e tolerância a falhas provido pelo Hyperledger na execução dos serviços e atualização do *ledger* provêm uma segurança adicional ao modelo vislumbrado.

A proposta do modelo de segurança apresentado atende também as metas de segurança apresentados por [Cleeff, Pieters e Wieringa 2010]:

- há uma clara separação de deveres (*SoD*), provendo-se o gerenciamento das permissões aos serviços diretamente aos seus provedores;
- ao definir os *assets* necessários como permissões, usadas exclusivamente para o acesso a um serviço, atende-se o requisito de mínimo privilégio;
- o uso dos registros no livro razão da blockchain permite a responsabilização dos atores de forma praticamente inviolável;
- a possibilidade de gerar novos *assets* para concessões de privilégios de acesso, ou mesmo mudar os requisitos na prestação do serviço, permite o manutenção e revogação de direitos de acesso;
- ao basear suas operações em requisições existentes em uma infraestrutura de blockchain, a proposta atende também a requisitos de usabilidade.

Aspectos de escalabilidade e operação das transações em tempo real, precisam ser melhor investigados nos modelos propostos. Isso vale especialmente para plataformas de blockchain nas quais o *proof-of-work* demanda processamento significativo, assim como as operações de verificação completa das operações em relação aos blocos de registro.

Referências

AZARIA, A. et al. Medrec: Using blockchain for medical data access and permission management. In: *2016 2nd International Conference on Open and Big Data (OBD)*. [S.l.: s.n.], 2016. p. 25–30.

BERTINO, E.; KIRKPATRICK, M. S. Location-based access control systems for mobile users: Concepts and research directions. In: *Proceedings of the 4th ACM SIGSPATIAL International Workshop on Security and Privacy in GIS and LBS*. New York, NY, USA: ACM, 2011. (SPRINGL '11), p. 49–52. ISBN 978-1-4503-1032-1.

CACHIN, C. et al. Architecture of the hyperledger blockchain fabric*. In: . [S.l.: s.n.], 2016.

CHOI, S. et al. Secure and resilient proximity-based access control. In: *Proceedings of the 2013 International Workshop on Data Management & Analytics for Healthcare*. New York, NY, USA: ACM, 2013. (DARE '13), p. 15–20. ISBN 978-1-4503-2425-0.

CLEEFF, A. v.; PIETERS, W.; WIERINGA, R. Benefits of location-based access control: A literature study. In: *Proceedings of the 2010 IEEE/ACM Int'L Conference on*

Green Computing and Communications & Int'L Conference on Cyber, Physical and Social Computing. Washington, DC, USA: IEEE Computer Society, 2010. (GREENCOM-CPSCOM '10), p. 739–746. ISBN 978-0-7695-4331-4.

DEY, A. K. Understanding and using context. *Personal Ubiquitous Comput.*, Springer-Verlag, London, UK, UK, v. 5, n. 1, p. 4–7, jan. 2001. ISSN 1617-4909.

ENGLISH, M.; AUER, S.; DOMINGUE, J. Block chain technologies & the semantic web: A framework for symbiotic development. In: *Computer Science Conference for University of Bonn Students, J. Lehmann, H. Thakkar, L. Halilaj, and R. Asmat, Eds*. [S.l.: s.n.], 2016. p. 47–61.

FERRAIOLO, D.; KUHN, R. Role-based access controls. In: BALTIMORE, MARYLAND: NIST-NCSC. *Proceedings of 15th NIST-NCSC National Computer Security Conference*. [S.l.], 1992. v. 563.

GUPTA, A.; KIRKPATRICK, M.; BERTINO, E. A formal proximity model for rbac systems. In: *8th International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom)*. [S.l.: s.n.], 2012. p. 1–10.

HASHEMI, S. H. et al. World of empowered iot users. In: *2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI)*. [S.l.: s.n.], 2016. p. 13–24.

HU, V. C. et al. Guide to attribute based access control (abac) definition and considerations (draft). *NIST special publication*, v. 800, n. 162, 2014.

KUHN, D. R.; COYNE, E. J.; WEIL, T. R. Adding attributes to role-based access control. *Computer*, v. 43, n. 6, p. 79–81, June 2010. ISSN 0018-9162.

NAKAMOTO, S. Bitcoin: A peer-to-peer electronic cash system. November 2008.

OUADDAH, A.; ELKALAM, A. A.; OUAHMAN, A. A. Fairaccess: a new blockchain-based access control framework for the internet of things. *Security and Communication Networks*, v. 9, n. 18, p. 5943–5964, 2016. ISSN 1939-0122. SCN-16-0184.

TONINELLI, A. et al. A semantic context-aware access control framework for secure collaborations in pervasive computing environments. In: ____. *The Semantic Web - ISWC 2006: 5th International Semantic Web Conference, ISWC 2006, Athens, GA, USA, November 5-9, 2006. Proceedings*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006. p. 473–486. ISBN 978-3-540-49055-5.

WOOD, G. Ethereum: A secure decentralised generalised transaction ledger. *Ethereum Project Yellow Paper*, v. 151, 2014.

ZYSKIND, G.; NATHAN, O.; PENTLAND, A. S. Decentralizing privacy: Using blockchain to protect personal data. In: *Proceedings of the 2015 IEEE Security and Privacy Workshops*. Washington, DC, USA: IEEE Computer Society, 2015. (SPW '15), p. 180–184. ISBN 978-1-4799-9933-0.