Como a plataforma ION garante a segurança das identidades digitais descentralizadas

Rodrigo S. P. Hirao, Marco A. Amaral Henriques
r186837@dac.unicamp.br, maah@unicamp.br

Departamento de Engenharia de Computação e Automação (DCA)
Faculdade de Engenharia Elétrica e de Computação (FEEC)
Universidade Estadual de Campinas (Unicamp)
Campinas, SP, Brasil

Resumo – Sistemas de gestão de identidades digitais são usualmente feitos utilizando um sistema central para o armazenamento e prova dos dados, o que pode causar desconfiança sobre a segurança e privacidade envolvida durante o processo. Foi feito o esforço pela W3C de especificar uma arquitetura padrão para identidades digitais descentralizadas, que se caracterizam por colocar sob a guarda e controle dos seus usuários seus dados de identificação e demais metadados relacionados, gerando assim o conceito de identidade autossoberana. Essa especificação foi utilizada para a definição do protocolo Sidetree. A plataforma ION é a implementação do protocolo Sidetree utilizando a blockchain da criptomoeda Bitcoin (como âncora de confiança), o sistema de arquivos distribuído IPFS (como sistema de armazenamento endereçado por conteúdo) e a base de dados MongoDB (como cache da blockchain). Utilizando as ferramentas e especificações disponibilizadas pela ION foi estudado, por diversos métodos, o fluxo de uma identidade desde sua criação até sua resolução, para, assim, poder verificar como o sistema garante a segurança envolvida em cada etapa.

Palavras-chave – identidade descentralizada, identidade autossoberana, plataforma ION, blockchain

1. Introdução

Em diversos contextos é necessária a identificação do usuário para o uso de um serviço e para isso é usado uma identidade acompanhada de uma credencial. A identidade possibilita a identificação única do usuário dado o contexto e a credencial atesta a autoridade do usuário sobre essa identidade [4].

As identidades podem ser classificadas em 3 categorias [7]. A primeira é a identidade física, que está sob o controle do usuário [8]. Um exemplo é o RG, que pode ser guardado na carteira e usado quando o usuário achar preciso, tendo como credenciais a foto do rosto e a assinatura. A segunda categoria é a do documento de identidade digital, que nada mais é que uma representação digital do documento físico. A terceira é a do documento de identidade eletrônica, a identidade que foi criada digitalmente e não possui uma representação física. Esta identidade precisa de um provedor de identidades (*IdP*) relacionada a um provedor de serviço (*SP*).

Uma identidade centralizada contém um *SP* que também é o *IdP*, assim o SP tem o controle total dos dados do usuário. Isso pode criar problemas como o usuário ter que gerenciar diversas identidades para diversos *SP*s.

Uma outra solução então é uma identidade federada, onde um *IdP* é feito por um serviço terceiro de confiança e o *SP* se comunica com o *IdP* para provar as credenciais. Embora esse método evite o problema anteriormente citado ele mantém o controle sobre a identidade por parte de um provedor, comprometendo a privacidade e segurança [10].

Uma outra solução, mais recente, é a adoção do conceito de identidades autossoberanas (*SSI*) [4], onde o usuário tem um maior controle de suas informações privadas armazenadas consigo e não depende totalmente de um terceiro para se autenticar [8]. Porém tal solução pode ser mais complexa na implantação e uso cotidiano, pois usuários leigos podem ter maior dificuldade para provar o controle de sua identidade.

Para obter uma *SSI* digital foi proposto o modelo de Identidades Digitais Descentralizadas (*DID*), com os objetivos de ser descentralizado, persistente, verificável criptograficamente e resolvível [5]. Dispondo de uma solução para o modelo de *SSI* que consiga provar sua identidade sem a necessidade de uma IdP, um usuário poderá fazer a autenticação diretamente com o provedor de serviços (*SP*).

Foi feito um esforço pela *W3C* para definir padrões na implementação de *DIDs*[9], de modo a deixar as tecnologias usadas acima do protocolo como uma escolha da implementação do protocolo. Assim foi criado o protocolo *Sidetree*[2] que segue as especificações da *W3C* possibilitando a escolha de uma tecnologia para sistema de ancoragem de dados sensíveis, uma para armazenamento endereçado por conteúdo (*CAS*) e uma para banco de dados (*DB*) de cache, o que resultou na implementação

da plataforma *ION*, que usa a *Sidetree* com a *blockchain Bitcoin* como sistema de ancoragem, o sistema de arquivos distribuído *IPFS* (*InterPlanetary File System*) como *CAS* e a base de dados *MongoDB* como cache.

2. Proposta

Após o entendimento da especificação de DID pela W3C e do protocolo da Sidetree, a plataforma ION foi estudada detalhadamente por meio de seu código e documentações, com o fim de executar o protocolo diretamente acima da bitcoin e ipfs, sem utilizar as ferramentas fornecidas pela Microsoft, fortalecendo sua descentralização pela independência de ferramentas centralizadas.

Ao conseguir usar a DID em um nível mais baixo de implementação foi possível analisar as âncoras de confiança atreladas às tecnologias envolvidas, como a bitcoin e ipfs. Assim conseguimos verificar as assinaturas digitais e a gestão das chaves que as garantem e avaliar a segurança dessa implementação do protocolo.

Após a avaliação das âncoras de confiança, avaliaremos se há possíveis melhorias no protocolo a que possam ser feitas sem comprometer a segurança já estabelecida.

3. Especificações de DID pela W3C

A *W3C* define uma *DID* como uma *URI* (*Uniform Resource Identifier*) **did:{método}:{identificador}** Figura 1, sendo o **{método}** a definição de como a *DID* foi implementada e o **{identificador}** o identificador único da *DID* relativo ao método utilizado. Tal *URI* referencia um documento *DID* formatado como *JSON* (*Javascript Object Notation*) que possui provas criptográficas para garantir as propriedades da *DID*, como chaves públicas relacionadas à chave privada que apenas o controlador possui. Além disso, o documento também possui informações sobre todos os serviços referenciados pela *DID*, que podem também usar alguma chave do documento como prova criptográfica de que o serviço existiu e criou tal documento.

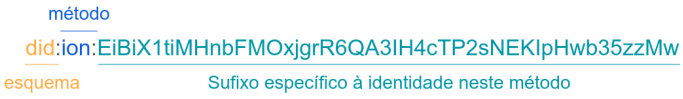

Figura 1. URI de uma DID de exemplo

4. O protocolo Sidetree

A *Sidetree* é o protocolo que implementa as especificações fornecidas pela *W3C* disponibilizando uma biblioteca em *NodeJS*, que pode ser implementada com qualquer interface para seu sistema de ancoragem, *CAS* e base de dados de cache. Para tanto a implementação da *Sidetree* precisa definir esses módulos de sistema de ancoragem, *CAS* e *DB* de cache. Na biblioteca disponibilizada já existem módulos de exemplo que são implementados com a *blockchain* da *Bitcoin* como sistema de ancoragem, o *IPFS* como *CAS* e o *MongoDB* como base de dados de cache.

4.1. Criação de uma DID

Durante o fluxo de criação de uma *DID*, é criada uma lista de objetos de atualização do documento e essa atualização pode ser uma criação de uma nova *DID*. Cada documento é derivado de um *"delta"* que define a diferença entre o estado anterior da *DID* e o atual, onde o estado anterior pode ser vazio no caso da inexistência da *DID*.

Cada operação do delta poderá adicionar, remover ou atualizar chaves públicas no formato de *JSON Web Key* (*JWK*) [6] e serviços, bem como definir seus papéis e objetivos. Após a criação do *delta*, este deverá passar por um processo de *hash sha256* e codificação em *base64url* resultando no que é chamado de *"deltaHash"*. Tendo o *delta* e o *deltaHash*, podemos enviar os dados usando *HTTP* a um servidor executando uma implementação da *Sidetree* com o método escolhido, que por sua vez irá inserir estes dados em uma fila onde o *deltaHash* é a chave de busca e o delta é o resultado (Figura 2).

Um programa chamado Escritor em Lote é executado no nó em paralelo (Figura 3), a fim de consumir a fila para inserção e criar o registro no *CAS* desejado (*IPFS* no caso do ION), o qual é chamado de *Chunk File*. Em seguida é criado um arquivo de provas para operações de atualização chamado de *Provisional Proof File* e um terceiro arquivo apontando para os endereços tanto do *Chunk File* quanto do *Provisional Proof File*, chamado de *Provisional Index File*. Por fim é criado um arquivo para provas de linhagem da *DID* chamado de *Core Proof File* e um arquivo final chamado de *Core Index File* que terá os endereços do *Core Proof File* e do *Provisional Index File*, criando uma árvore com o *Core Index File* como raiz.

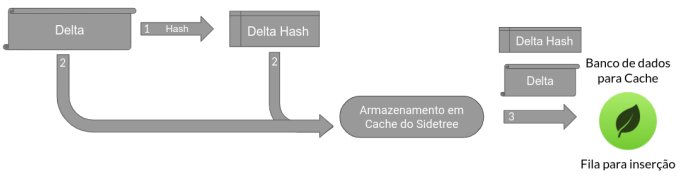

Figura 2. Criação de uma DID na fila de cache.

Feitas as operações desejadas no *CAS*, o *Core Index File* será adicionado, concatenado com o número de

operações existentes na forma **{número de operações}.{endereço do core index file}** no sistema de ancoragem, como pode ser visto na Figura 3.

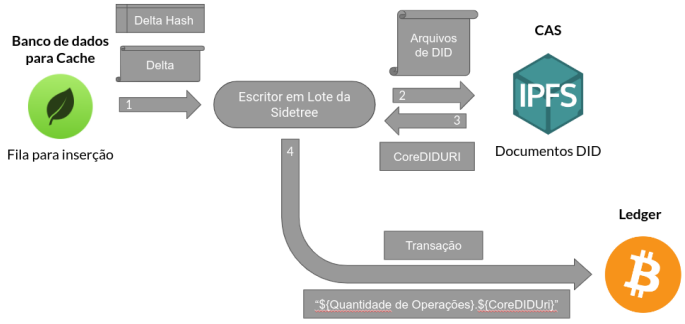

Figura 3. Criação de uma DID no CAS e no sistema de ancoragem a partir da fila.

4.2. Resolução de uma DID

Para resolver uma *DID* pela *URI* podemos seguir 2 caminhos, dependendo de sua formatação: caso ela esteja no formado de *URI* longa (**did:{método}:{deltaHash}:{base64(delta)}**) basta decodificar a {base64(delta)}. Porém, para uma *URI* curta (Figura 1) não conseguimos obter de imediato o conteúdo da *DID*.

É preciso percorrer o sistema de ancoragem procurando pelas operações *DID* do método indicado pela *URI* e, um por um, devemos pegar o *Core Index File* e percorrer a árvore no *CAS* até chegarmos nos *Chunk Files*, que irão descrever os deltas dessa transação no sistema de ancoragem. Em seguida, pegamos os deltas obtidos e guardamos em um banco de dados intermediário de cache usando o *hash* do delta como chave, como pode ser visto na Figura 4.

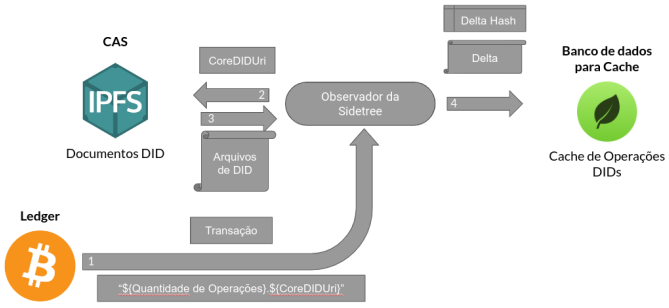

Figura 4. Resolução de uma DID para o cache

Após ter adicionado todos os deltas encontrados no sistema de ancoragem no banco de dados intermediário é possível encontrar rapidamente o documento desejado (por meio do hash do delta presente na *URI* da *DID*) no cache criado usando o banco de dados intermediário (Figura 5).

Figura 5. Resolução de uma DID a partir do cache.

5. ION: uma implementação da Sidetree

A Sidetree possibilita uma implementação com escolha de *CAS* e Sistema de ancoragem, assim surgindo implementações diversificadas, como a *Element* que usa a *Ethereum* com a *IPFS*, ou a photon que usa a *Amazon QLDB* com a *Amazon S3*. A *ION* utiliza a *blockchain* da *Bitcoin*, afirmando que essa é a *blockchain* aberta mais segura [3], em conjunto com o *IPFS* (como *CAS*).

A *ION* oferece uma ferramenta para implementação de um servidor *ION* (chamado de nó), usando a *API* da *Sidetree*, que já possui a blockchain de *Bitcoin* e o sistema de arquivos distribuído *IPFS* implementados como padrão, criando assim pontos de conexão (*endpoints*) *http* do tipo *rest* para gerenciamento básico de uma *DID*, como criação e resolução, por exemplo [1].

6. Resolução de URI em ION

Para avaliar se o entendimento do protocolo está correto e se o estabelecimento das âncoras de confiança foi feito corretamente para a *DID* da Figura 1, utilizamos as ferramentas disponibilizadas pela *ION* e acompanhamos passo a passo a resolução de uma *DID*, conforme a Figura 6.

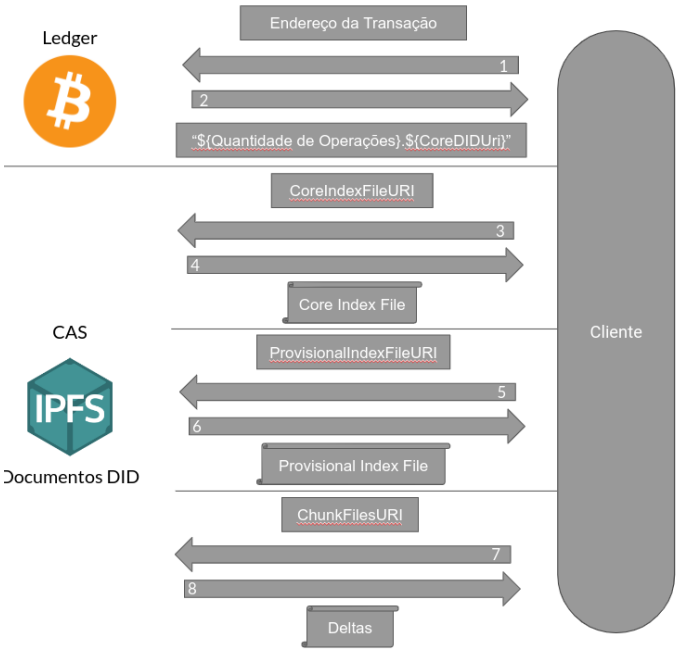

Figura 6. Fluxo para resolução manual da DID.

O problema inicial encontra-se ao tentar descobrir o endereço de transação, para tanto, foi necessário observar a bitcoin durante a ancoragem da *DID*, o que resultou em uma única transação com o padrão *"ion:1.{código}"* durante um intervalo de vários minutos, indicando similaridade com o que foi ancorado recentemente.

Usamos o *hash {código}* para buscar o Core Index File em um domínio disponível para busca na IPFS, o que retorna um arquivo compactado com o *Core Index File*. Este possui o endereço para o *Provisional Index File*.

Assim, repetindo o passo anterior e buscando pelo arquivo na *IPFS* temos o *Provisional Index File* compactado, que, após descompactado, resulta no *JSON* com o endereço de um *Chunk File*.

Com o endereço do *Chunk File* podemos repetir novamente o processo e buscar o arquivo compactado na *IPFS*, para assim conseguir o delta.

Com esses 3 arquivos podemos derivar o documento *DID* referente à Figura 1, por estes possuírem todas as informações necessárias para a composição do documento. Assim, constatamos que foi possível percorrer todos os passos necessários para identificar e encontrar um recurso na blockchain e no IPFS a fim de verificar como estão asseguradas as garantias de confiança no documento *DID*.

7. Conclusões e trabalhos futuros

Devido à busca por soluções autossoberanas de identidades a solução de *DIDs* se mostrou interessante, tendo um padrão criado pela *W3C* para sua implementação e um protocolo acima desse padrão (*Sidetree*) fornecendo uma fácil implementação de diferentes métodos de *DID*, como a *ION*, que funciona usando a *Bitcoin* como sistema de ancoragem e depende da *IPFS* para armazenar os documentos *DID* gerados.

O armazenamento *IPFS*, ao indexar seus arquivos usando um *hash* do conteúdo do arquivo, impediu o arquivo de ser alterado sem mudar seu índice. Porém evidenciou a necessidade de um sistema de ancoragem imutável para provar a ordem e atualidade de seus documentos *DID*, uma vez que a *DID* pode ter sido removida ou atualizada em uma transação futura, mas o documento antigo continua na rede *IPFS*.

Observando o funcionamento passo a passo da resolução de uma *DID ION* foi possível perceber sua dependência da imutabilidade e persistência da *Bitcoin*, uma vez que é possível adquirir uma *DID* desatualizada, mesmo se for encontrado o bloco correto na blockchain da *Bitcoin*. Assim, é necessário fazer uma leitura completa da *blockchain* pelo menos uma vez.

Este fato torna difícil a resolução de DIDs sem a utilização de um servidor executando um serviço *ION* com um banco de dados de cache.

Não foi contemplado neste artigo o funcionamento das operações de atualização, remoção e recuperação da *DID*. Outro trabalho futuro seria interessante analisar as diferentes possibilidades de sistemas de ancoragem, no lugar da *Bitcoin*, que possui um elevado tempo entre suas transações e pode ter um custo muito elevado para o contexto de *DIDs*.

Referências

[1] Daniel Buchner. Did method implementation using the sidetree protocol on top of bitcoin, 2020. https://github.com/decentralized-identity/ion Acessado em 26 Junho 2022.

[2] Daniel Buchner, Orie Steele, and Troy Ronda. Sidetree v1.0.0, 2021. identity.foundation/sidetree/spec/ . Acessado em 26 Junho 2022.

[3] Daniel Buchner and Henry Tsai. Q&a about ion, 09 2021. https://github.com/decentralized-identity/ion/blob/66813123cf81ace05cea2039e93ef263952d6283/docs/Q-and-A.md Acessado em 26 Junho 2022.

[4] Md Sadek Ferdous, Farida Chowdhury, and Madini O. Alassafi. In search of self-sovereign identity leveraging blockchain technology. *IEEE Access*, 7:103059–103079, 07 2019.

[5] Kim Hamilton-Duffy, Ryan Grant, and Adrian Gropper. Use cases and requirements for decentralized identifiers, 2021. https://www.w3.org/TR/did-use-cases/ . Acessado em 26 Junho 2022.

[6] M. Jones. Json web key (jwk), 05 2015. https://datatracker.ietf.org/doc/html/rfc7517 Acessado em 26 Junho 2022.

[7] Frederico Schardong and Ricardo Custódio. Self-sovereign identity: A systematic map and review. *ACM Comput. Surv. 1*, Article 1, 08 2021.

[8] Johannes Sedlmeir, Reilly Smethurst, Alexander Rieger, and Gilbert Fridgen. Digital identities and verifiable credentials. *Bus Inf Syst Eng*, 63:603–613, 12 2020.

[9] Manu Sporny, Dave Longley, Markus Sabadello, Drummond Reedand, Orie Steele, and Christopher Allen. Decentralized identifiers (dids) v1.0, 2021. https://w3c.github.io/did-core/ . Acessado em 26 Junho 2022.

[10] Andrei Volkov. Addressing the challenges facing decentralized identity systems. *iSChannel*, pages 10–15, 09 2020.