Identidade e Autenticação de Serviços através de Ambientes e Protocolos de Descoberta Heterogêneos: Uma Solução baseada em *Self-Certifying Names*

Paulo L. S. Brizolara[1], **Leonardo Cunha de Miranda**[1]

[1] Departamento de Informática e Matemática Aplicada
Universidade Federal do Rio Grande do Norte (UFRN) – Natal, RN – Brazil

pl.briz@proton.me,leonardo@dimap.ufrn.br

Resumo. *Sistemas de descoberta de serviços tem sido utilizados para localizar serviços, de forma automatizada, em diferentes aplicações e ambientes de rede. Para cada ambiente ou aplicação, diferentes protocolos de descoberta (SDPs) podem ser utilizados, frequentemente incompatíveis entre si. A literatura descreve soluções para descoberta através de múltiplos ambientes e/ou SDPs, mas a falta de mecanismos para identificar e reconhecer os serviços localizados, termina impedindo a descoberta integrada através desses meios distintos. Para tratar essa questão, o presente trabalho propõe uma solução, baseada no princípio dos self-certifying names, para identificar e autenticar serviços, através de ambientes e protocolos de descoberta heterogêneos.*

Abstract. *Service discovery systems has been used to find services automatically, in multiple applications and network environments. For each environment or application, different service discovery protocols (SDPs) may be used, often incompatible with each other. The literature describes solutions for discovery across multiple environments and/or SDPs, but the lack of mechanisms to identify and recognize discovered services prevents the integrated discovery across these distinct environments. To handle this issue, the current paper proposes a solution, based on the principle of self-certifying names, to identify and authenticate services across heterogeneous environments and discovery protocols.*

1. Introdução

Em sistemas distribuídos, estabelecer a comunicação com um componente remoto (como um serviço) requer uma forma de identificar e localizar esse componente. Protocolos de descoberta de serviços (SDP, do inglês *Service Discovery Protocol*) tem sido aplicados para a localização automatizada de serviços através de ambientes e contextos de uso diversos: desde sistemas *peer-to-peer* [Ahmed e Boutaba 2011, Khatibi e Sharifi 2021], e Internet das Coisas (IoT) [Pourghebleh et al. 2020, Achir et al. 2020], até *clusters* de alto processamento e sistemas em nuvem [Zarrin et al. 2018, Mohammed et al. 2020]. No entanto, a grande diversidade entre esses ambientes e aplicações faz com que sejam necessários SDPs especializados, que de modo geral são incompatíveis entre si. Dessa forma, para interagir com ambientes diversos, aplicações necessitam de uma solução ca-

paz de identificar e localizar serviços através de ambientes e SDPs heterogêneos.

A literatura descreve soluções para descoberta de serviços através de múltiplos ambientes e/ou múltiplos SDPs, i.e. soluções multi-ambiente ou multi-protocolo (e.g., [Lardies et al. 2009, Battaglia e Lo Bello 2018, Pantazoglou et al. 2006, Frank et al. 2008, Flores et al. 2011, Rodrigues et al. 2011, Siebert et al. 2007, Raverdy et al. 2006]). Falta, no entanto, uma solução para identificar e integrar os serviços descobertos através de ambientes e SDPs diferentes. Sem essa capacidade, os serviços se tornam específicos aos SDPs ou ambientes pelos quais foram localizados, o que torna impossível reconhecer, localizar e, portanto, utilizar um serviço através de ambientes e SDPs distintos.

Para tratar esse desafio, foi desenvolvido um sistema para descoberta integrada multi-ambiente e multi-protocolo. Em seu núcleo está a solução descrita neste trabalho, responsável por identificar e autenticar serviços, de forma descentralizada e independente do ambiente ou SDP. Esta solução se baseia no princípio dos chamados *self-certifying names*, identificadores derivados de chaves criptográficas que incluem as informações suficientes para autenticar os objetos referenciados e identificá-los unicamente.

Este artigo está organizado da seguinte maneira: a Seção 2 introduz o *background* do trabalho, revisitando como a atribuição de nomes é feita tradicionalmente nos sistemas de descoberta e apresentando os *self-certifying names*; a Seção 3 descreve os requisitos necessários para a solução de identidade proposta, e como os trabalhos relacionados da literatura atendem a esses requisitos; a Seção 4 descreve os *schemas* de identidade e de autenticação de serviços, e os mecanismos de distribuição de chaves projetados; a Seção 5 discute os resultados alcançados, assim como limitações e desafios relacionados à solução; e a Seção 6 conclui o artigo.

2. Background: Nomes de Serviços e Self-Certifying Names

Segundo Sundramoorthy et al. (2009), sistemas de descoberta de serviços são um tipo de sistema de descoberta de nomes. Nesses sistemas, cada entidade recebe um nome (ou identidade), que permite referenciá-la e acessá-la. A autoridade de nomes é a entidade responsável pela atribuição e gerenciamento de nomes em um *namespace* [Ahmed et al. 2005]. Essa autoridade pode ser centralizada, distribuída através de diferentes domínios, ou mesmo auto-gerenciada pelo próprio serviço [Ahmed et al. 2005]. A solução tradicional para atribuição de nomes na Internet é o DNS. No DNS, nomes são definidos de forma hierárquica, a partir da zona raiz [Wachs et al. 2014b,a]. A autenticidade desses nomes é verificada através de hierarquias de *trusted third parties*, como autoridades certificadoras (CA, do inglês *Certificate Authority*), no protocolo TLS (TLS, do inglês *Transport Layer Security*)[1], ou "*cadeias de autenticação*" no DNSSEC[2]. Mas apesar dessa estrutura ser o padrão de fato na Internet, diferentes fatores dificultam sua aplicação em sistemas descentralizados e ambientes heterogêneos, tais como: a dependência de disponibilidade da infraestrutura, a ausência de auto-configuração e o custo (financeiro e computacional) para o registro e manutenção de novos nomes.

Os chamados *self-certifying pathnames* [Mazières e Kaashoek 1998] ou, de forma mais genérica, *self-certifying names* [Benet 2019] representam uma solução descentralizada para essas limitações. Nos *self-certifying names*, chaves públicas criptográficas são

[1] Conforme TLSv3, definido pelo RFC 8446.

[2] Conforme DNSSEC, definido pelo RFC 4033.

utilizadas tanto para derivar IDs criptográficos, que atuam como *namespaces* [Wachs et al. 2014a], permitindo a definição de nomes globalmente únicos; quanto como autoridades certificadoras, para verificar a autenticidade dos dados registrados sobre esses nomes [Rivest e Lampson 1996]. Dessa forma, os *self-certifying names* dispensam *trusted third parties*, como CAs, pois trazem embutidos os IDs criptográficos, utilizados para validar a autenticidade dos nomes e dos objetos apontados por eles.

Duas técnicas são utilizadas para derivar IDs, a partir de chaves criptográficas: o uso direto das chaves, como é feito no *Gnu Name System* (GNS) [Wachs et al. 2014a] e na versão 3 dos *Tor Hidden Services* [TorProject 2017]; ou o uso de um *hash* (*fingerprint*) da chave pública, como no *InterPlanetary File System* (IPFS) [Benet 2019] e SFS [Mazières e Kaashoek 1998]. O uso direto da chave permite que a distribuição dela seja feita diretamente com o identificador, mas requer chaves com tamanho reduzido, Por exemplo, GNS utiliza chaves de curvas elípticas [Wachs et al. 2014a], com a *curva 25519* [Bernstein 2006], que tem 32 *bytes* de comprimento. Já os *fingerprints* (ou *thumbprints*), i.e. *hashs* gerados a partir de uma chave criptográfica [Finney et al. 2007, Jones e Sakimura 2015], tem um tamanho fixo, dependendo do algoritmo de *hash*, o que permite representar, de forma compacta, chaves com tamanhos arbitrários. Nesse caso, entretanto, é preciso ter também um outro mecanismo para distribuição das chaves.

Sendo derivados de chaves criptográficas, os *self-certifying names* proveem algumas propriedades desejáveis para a construção de identificadores de serviços: (i) um controle descentralizado sobre a atribuição de nomes, eliminando a dependência de terceiros; (ii) a possibilidade de auto-gerenciamento e auto-configuração de identidades, de forma transparente para os usuários; (iii) um custo mínimo (computacional e financeiro) para a atribuição e manutenção de nomes; e (iv) a independência da infraestrutura de resolução e distribuição de nomes, que pode ser adaptada ao ambiente de execução.

3. Trabalhos Relacionados

Soluções de identidade baseadas em *self-certifying names* tem sido empregadas em sistemas descentralizados, como IPFS[3] [Benet 2019], GNS[4] [Wachs et al. 2014a], *Tor Hidden Services*[5] e SDSI/SPKI[6] [Rivest e Lampson 1996]. Além disso, padrões de identidade seguras, como SPIFFE (spiffe.io), e descentralizadas, como *Decentralized Identifier*s (DIDs)[7], tem sido propostas. No entanto, essas soluções não abordam a mesma problemática deste trabalho. Dessa forma são consideradas, a seguir, apenas soluções de descoberta que tratam dessa questão, isto é, identificar e integrar serviços descobertos através de SDPs e ambientes heterogêneos.

Para analisar essas soluções foram considerados alguns requisitos importantes para lidar com a problemática em questão: *descentralização* e *autonomia*, que permitem a operação em contextos sem infraestrutura dedicada; *segurança* na descoberta e acesso de serviços, para lidar com ambientes não confiáveis; e o suporte a serviços *móveis* e *multi-dispositivo*, necessários, respectivamente, para aplicações em ambientes dinâmicos

[3] ipfs.io.
[4] www.gnunet.org/en/gns.html.
[5] gitweb.torproject.org/torspec.git/tree/rend-spec-v3.txt.
[6] União da *Simple Distributed Security Infrastructure* (SDSI) e *Simple Public Key Infrastructure* (SPKI).
[7] Uma recomendação do W3C definida em: w3.org/TR/did-core.

ou com múltiplos dispositivos.

Nenhuma das soluções multi-protocolo ou multi-ambiente encontradas na literatura, no entanto, atendem a todos esses requisitos. A maior parte dessas soluções não especificam uma forma de identificar serviços através de ambientes e/ou protocolos heterogêneos (e.g. [Pantazoglou et al. 2006, Frank et al. 2008, Flores et al. 2011, Rodrigues et al. 2011])[8]. Lardies et al. (2009) e Battaglia e Lo Bello (2018) empregam identificadores de serviços independentes do ambiente, mas que não suportam o uso de múltiplos SDPs. Apenas MUSDAC [Raverdy et al. 2006, Del Campo et al. 2006] define um identificador para as entidades do sistema que pode ser gerado de forma independente do ambiente e SDP utilizados (um UUID[9]). MUSDAC também provê autenticação, integridade e confidencialidade para a descoberta, através da infraestrutura de chaves públicas (PKI, do inglês *Public Key Infrastructure*) e modelo de confiança de Almenárez et al. (2004). No entanto, a solução não explicita a possibilidade de reconhecer um serviço descoberto através de SDPs distintos, nem define uma forma de distribuição de chaves adaptável a diferentes ambientes de execução. Além disso, a arquitetura da solução depende de um ponto de centralização na rede local, o chamado "*network manager*", que trata as requisições de descoberta e acesso [Raverdy et al. 2006]. Essa centralização reduz a autonomia dos componentes do sistema e cria um ponto único de falha, ainda que local. A solução proposta no presente trabalho busca lidar com essas limitações e atender aos requisitos levantados para identificar serviços através de ambientes e SDPs heterogêneos.

4. A Solução Proposta

O princípio dos *self-certifying names* é base para os *schemas* de identidade e de autenticação de serviços descritos neste trabalho. Desenvolvidos como parte de uma solução de descoberta cujo objetivo é integrar a descoberta de serviços através de ambientes e SDPs heterogêneos, os *schemas* de identidade e de autenticação proveem o "elemento de ligação" que permite a um serviço publicado através de diferentes meios, ser reconhecido como uma entidade única. Antes de detalhar esses *schemas*, entretanto, um cenário de uso é apresentado, na subseção a seguir, para ilustrar o uso da solução proposta. Em seguida são descritos o ***schema* de identidade**, que especifica as informações que identificam um serviço; e o ***schema* de autenticação**, que define os passos para verificar a autenticidade de um serviço descoberto, e acessá-lo de forma segura. Em conjunto com estes *schemas*, a solução proposta permite integrar diferentes **mecanismos de distribuição de chaves**, que definem os protocolos para localizar e compartilhar chaves criptográficas. Dois mecanismos implementados são descritos nessa seção.

4.1. Cenário de Uso

O cenário a seguir ilustra o uso da solução proposta em uma aplicação hipotética para compartilhar fotografias utilizadas como registros de memórias [Brizolara e Miranda 2018]. O cenário inicia com Alice conversando com seu amigo Bob. Alice lembra de uma foto engraçada que tiraram juntos na última SBSeg. Para compartilhar a foto com Bob, ambos inicializam seus aplicativos no *smartphone*. Através da descoberta local, o cliente de Alice localiza o serviço oferecido pelo dispositivo de Bob, que é selecionado

[8]Por uma questão de espaço, nem todas as soluções multi-ambiente e multi-protocolo encontradas foram listadas aqui.

[9] UUIDs são especificados pelo RFC 4122.

por Alice. Após o compartilhamento, seus clientes armazenam o contato um do outro. Mais tarde, Bob fala com Carol, uma amiga em comum. Bob compartilha o contato de Alice, que contém o identificador de serviço dela. A partir desse identificador, o cliente de Carol localiza o computador de Alice, através de um dos mecanismo de descoberta disponíveis, um SDP remoto *peer-to-peer* (P2P).

O cenário ilustrou capacidades da solução proposta, como o uso de identificadores para reconhecer e localizar serviços, e a descoberta através de múltiplos SDPs, ambientes e dispositivos. A seguir são descritos os componentes que possibilitam essas capacidades.

4.2. *Schema* de Identidade de Serviços

O *schema* para identificar serviços foi projetado para: unificar os registros de um mesmo serviço publicados através de diferentes SDPs e permitir referenciar e localizar serviços de forma independente do SDP utilizado. Esse *schema* aplica o princípio dos *self-certifying names* (vide Seção 2) para gerar identificadores globalmente únicos, cuja autenticidade pode ser verificada, de forma descentralizada, a partir de chaves públicas criptográficas.

A base de cada identificador de serviço, nesse *schema*, é a chave pública associada a um provedor. A partir dessa chave, é derivado o ID do provedor, que compõe o identificador de um serviço, em conjunto com o tipo, nome do serviço e, opcionalmente, um ID de instância, que pode ser utilizado para diferenciar múltiplas execuções de um mesmo serviço. O identificador de um serviço é imutável e persistente, mas o tempo de vida das instâncias do serviço é independente. Execuções do serviço podem ser efêmeras, definindo ou não novos IDs de instância, de acordo com a aplicação.

Um identificador de serviço pode ser representado de um modo compacto através de uma URL. Esse formato permite também apontar um determinado recurso no serviço descoberto, cujo significado varia com o protocolo. Além disso, a URL permite especificar atributos para alterar a sua interpretação ou representação. A URL na Figura 1, por exemplo, representa um serviço Web ("*myservice*"), em que o ID da instância e do provedor são representados em *base64url*[10] (*fmt=b64u*).

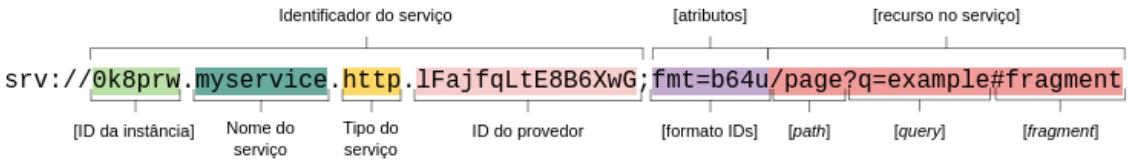

Figura 1. Partes da URL que representa um identificador de um serviço.

O *schema* de identidade proposto contribui para: (i) descentralização e segurança, ao prover uma base para a autenticação de serviços, sem depender de *trusted third parties*; e (ii) autonomia e suporte a serviços móveis e multi-dispositivo; ao permitir identificar múltiplas instâncias independentes de um mesmo serviço, seja em um único nó ou através de dispositivos diferentes. A seguir são detalhados os componentes do *schema* de identidade e como eles contribuem para alcançar essas propriedades.

[10] Base64Url é definido no RFC 4648.

4.2.1. ID do Provedor

O principal elemento no identificador de um serviço é o ID do provedor, que delimita um *namespace* globalmente único e serve de base para a autenticação dos serviços. Esse ID é definido como a *fingerprint* da chave pública do provedor. Isso permite que chaves criptográficas de diferentes tamanhos sejam utilizadas (pois o *fingerprint* não cresce com a chave). Por sua vez, para suportar diferentes tipos de chaves, foi adotado como padrão o algoritmo de *fingerprint* do *JWK Thumbprint* [Jones e Sakimura 2015]. Mas, outros algoritmos podem vir a ser suportados, como o definido no OpenPGP [Finney et al. 2007].

4.2.2. Tipo do Serviço

Para identificar as instâncias de serviços que são equivalentes entre si, o *schema* de identidade emprega o mesmo conceito de *tipo de serviço* adotado pelo DNS Service Discovery (DNS-SD) [Cheshire e Krochmal 2013]. Assim como no DNS-SD, cada serviço tem um tipo que expressa tanto "o quê" o serviço oferece, i.e. seu propósito ou função, quanto o "como", i.e. o protocolo utilizado pelo serviço [Cheshire e Steinberg 2005]. Com isso, os clientes podem buscar apenas os serviços que apresentem a funcionalidade desejada e com os quais eles possam de fato se comunicar [Cheshire e Steinberg 2005].

Para tanto, esses tipos de serviços, e os protocolos utilizados por eles, precisam ser definidos e conhecidos previamente pelas aplicações que irão utilizá-los. A Internet Assigned Numbers Authority (IANA) mantém um registro desses tipos de serviço[11], o que contribui para a padronização e, consequentemente, para a interoperabilidade. Mas, o *schema* de identidade não impõe restrições estritas quanto aos tipos de serviço utilizados.

4.2.3. Nome do Serviço

Assim como no DNS-SD [Cheshire e Krochmal 2013], o nome do serviço foi pensado para ser legível por humanos, de modo a facilitar que os usuários distinguam e reconheçam os serviços oferecidos por um provedor. Como o nome do serviço está contido no *namespace* definido pelo ID do provedor, esse nome não precisa ser globalmente único. Dessa forma, os usuários (ou aplicações) podem definir nomes arbitrários, compostos por quaisquer caracteres (UTF-8) imprimíveis. Nomes binários podem também ser definidos através de codificações textuais, como *base64* ou *base32*.

4.2.4. ID da Instância

Na solução de descoberta proposta, um serviço representa uma "entidade conceitual", e não um processo específico em uma determinada máquina. Isto é, um serviço pode ser oferecido (inclusive simultaneamente) por diferentes processos, i.e. instâncias daquele serviço. Essas instâncias são consideradas como independentes entre si (autônomas) e podem tanto estar distribuídas geograficamente, quanto contidas em um mesmo *host*. Com

[11] O RFC 6335 define os procedimentos para gerenciamento de tipos de serviços (ou "*service names*") no registro da IANA.

isso, a solução dá suporte à mobilidade do serviço, ao uso de múltiplos dispositivos e também contribui para a autonomia entre instâncias da solução.

Para suportar essa visão de múltiplas instâncias de um serviço, o *schema* de identidade projetado define um ID que permite distinguir instâncias específicas de um serviço. Esse ID da instância pode ser um valor binário definido de forma arbitrária pelas aplicações. Não há risco quanto a colisões hostis, pois a geração desses IDs é controlada por um mesmo provedor. No caso de colisões acidentais, as aplicações podem definir como resolver os conflitos. Por sua vez, durante a descoberta de serviços, as aplicações podem escolher como lidar com múltiplas instâncias de um serviço. De acordo com as necessidades, podem ser selecionadas uma ou mais instâncias de um serviço. Um cliente pode também (re-)localizar uma instância específica de um serviço, se necessário.

4.3. *Schema* de Autenticação de Serviços

O objetivo da autenticação, segundo Stallings (2014), é garantir a autenticidade de uma comunicação, o que se reflete em: garantir a autenticidade da fonte de uma mensagem; e evitar que terceiros possam interferir em uma conexão e se passar por uma das entidades comunicantes (*impersonate attacks*). No contexto da descoberta de serviços, esses objetivos se traduzem em: garantir a autenticidade das informações descobertas (i.e., descrições de serviços); e assegurar a conexão segura com os serviços selecionados. A seguir são descritas as medidas adotadas para tratar essas questões na solução proposta.

4.3.1. Autenticação das Descrições de Serviços

Para permitir a comprovação das descrições publicadas, o provedor utiliza sua chave privada para gerar uma assinatura da descrição do serviço, que é publicada junto com a descrição. Desse modo, quando um cliente recebe essa descrição, ele pode verificar a assinatura e, com isso, comprovar a autenticidade e integridade das informações recebidas.

No entanto, uma descrição válida pode ainda ser reutilizada de forma maliciosa, nos chamados ataques de *replay*. Para minimizar a possibilidade desse tipo de ataque, a descrição de um serviço inclui também: os "endereços válidos" do servidor e o "período de validade" da descrição. Com isso, para um ataque de *replay* ser efetivo, um atacante precisaria assumir um desses endereços, durante o tempo de validade da descrição. Esse período de validade também faz com que publicações de nós que tenham se desconectado de forma inesperada sejam invalidadas naturalmente. Com isso, no entanto, atualizações regulares precisam ser enviadas para manter as publicações válidas, o que requer que o tempo de validade seja escolhido de forma a equilibrar segurança e eficiência. A Figura 2 representa o processo de autenticação e verificação das descrições de serviços.

4.3.2. Autenticação das Conexões

Além de autenticar as descrições de serviços, para garantir a conexão segura com um servidor descoberto e evitar ataques como *IP spoofing* ou *Man in the Middle*, é necessário autenticar também o processo de conexão com o servidor, como é feito, por exemplo, no TLS. Entretanto, não é possível estabelecer um procedimento único compatível com qualquer tipo de protocolo de comunicação. Dessa forma, para não impor limitações, a

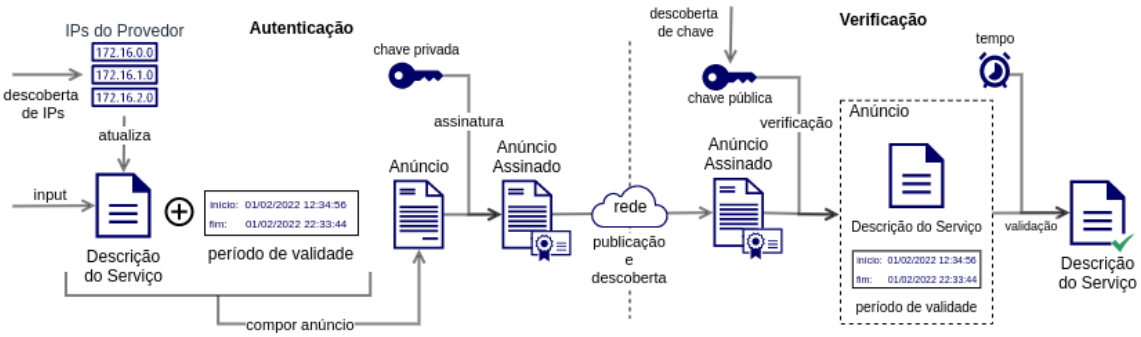

Figura 2. Processo de autenticação e verificação de descrições de serviços.

solução de descoberta proposta deixa a cargo das aplicações estabelecer e autenticar as conexões, mas permite que sejam distribuídas as informações necessárias para que esse processo possa ser realizado de forma segura.

Para tanto, a descrição de um serviço pode incluir, para cada canal de comunicação oferecido, o conjunto de chaves válidas para acessar esse canal de forma segura. O significado e representação dessas chaves pode variar de acordo com o protocolo de comunicação. Por exemplo, para um canal de comunicação baseado em TLS, pode ser utilizado o *fingerprint* da chave presente no certificado do servidor. Ao estabelecer a conexão, o cliente pode então verificar se o certificado contém a chave esperada. Se suportado pela aplicação e protocolo de comunicação, a própria chave do provedor, empregada na publicação do serviço, pode também ser utilizada para autenticar as conexões. Além disso, alguns protocolos podem omitir as informações sobre as chaves, por exemplo, quando o próprio endereço já descreve (direta ou indiretamente) a chave a ser utilizada. No caso dos Onion Services [TorProject 2017], por exemplo, a chave já está inclusa no próprio endereço; já protocolos como TLS e HTTPS, quando empregam a infraestrutura tradicional de CAs, definem implicitamente como obter e validar as chaves do servidor.

4.4. Mecanismos para Distribuição de Chaves Criptográficas

Para autenticar um serviço descoberto, clientes precisam ter acesso à chave pública do provedor. Isso requer uma forma de distribuição de chaves, adequada ao ambiente de execução e à infraestrutura de comunicação utilizados. Para lidar com essa variabilidade, a solução de descoberta proposta permite a integração de diferentes mecanismos de distribuição de chaves.

Cada mecanismo de distribuição de chaves é responsável pela publicação e localização de chaves públicas, através de um protocolo específico. Dessa forma, durante a publicação ou descoberta de serviços, todos os mecanismos de distribuição de chaves disponíveis são utilizados para publicar ou buscar uma chave. Uma busca se encerra quando qualquer dos mecanismos localiza uma chave válida, ou seja, uma chave com um formato suportado e cujo *fingerprint* corresponda ao expresso pelo ID do provedor. A solução proposta não exige dos mecanismos de distribuição de chaves garantias de segurança específicas, já que a autenticidade das chaves pode ser verificada a partir do identificador do serviço (daí a capacidade de auto-verificação dos identificadores).

A seguir são descritos os mecanismos de distribuição de chaves desenvolvidos: o primeiro, com base no *multicast DNS* (mDNS), foi desenvolvido para descoberta lo-

cal; já o segundo pode operar através da Internet, por meio de uma DHT (tabela hash distribuída, do inglês *Distributed Hash Table*), um *overlay* P2P que permite publicar e localizar conteúdos identificados a partir de um *hash* criptográfico [Meshkova et al. 2008].

4.4.1. Distribuição de Chaves através do Multicast-DNS

Como o DNS-SD/mDNS não define uma forma de representar ou distribuir chaves criptográficas, é preciso ou utilizar um serviço externo para isso, ou definir um *schema* para representação e descoberta de chaves através do mDNS. No mecanismo implementado, optou-se pela segunda abordagem. Para tanto, foram definidos: uma representação para chaves criptográficas através de registros DNS, respeitando as limitações do mDNS; e um protocolo para localização dessas chaves, através de requisições mDNS.

A representação das chaves criptográficas é feita através de dois conceitos: um descritor que caracteriza a chave, através de um conjunto de metadados; e o conteúdo da chave propriamente dito, que é particionado através de uma sequência de registros DNS, para facilitar sua transmissão no mDNS. O identificador de um serviço permite derivar o endereço do descritor. Por sua vez, o descritor permite localizar as partes da chave. Uma vez coletadas, essas partes são concatenadas, resultando em uma representação da chave. O *fingerprint* da chave obtida é gerado para verificar a validade desta.

4.4.2. Distribuição de Chaves através de DHTs

Como as capacidades e interfaces de DHTs variam, foi definido um modelo genérico de DHT que pode ser adequado a diferentes implementações. Esse modelo assume que múltiplos valores podem ser publicados/localizados em uma determinada chave (*hash*), que pode ser escolhida pela aplicação. O modelo de DHT adotado permite que as chaves criptográficas sejam distribuídas diretamente através da DHT. Para tanto, a publicação se baseia em três passos: o endereço (chave) para publicação é derivado da *fingerprint* da chave pública; a chave pública é serializada; e esses dados são publicados através da DHT. O processo de busca por uma chave é análogo, i.e. o endereço é derivado do *fingerprint* da chave procurada; uma requisição é enviada a esse endereço; e os resultados são deserializados e entregues ao núcleo da solução, que valida a chave e finaliza a busca.

A simplicidade dessa abordagem facilita a sua implementação e pode também contribuir para maior eficiência, tendo em vista que a publicação e busca pela chave demandam apenas uma requisição cada. Por sua vez, o uso da *fingerprint* como base para determinar o endereço da publicação provê uma boa distribuição do conteúdo através da DHT, o que consequentemente auxilia na distribuição de carga.

Uma solução mais eficiente poderia ser definida ao utilizar um modelo de DHT capaz de escutar e responder sob demanda por requisições. Essa capacidade não está disponível em todas DHTs, mas pode ser oferecida, por exemplo, por *overlays* que implementem *publish-subscription*. A abordagem implementada assume, no entanto, que a DHT possa distribuir as chaves publicadas diretamente, o que pode ser difícil para chaves grandes. Dessa forma, para garantir a entrega, a DHT utilizada deve ter essa capacidade.

4.5. Implementação

A solução para identidade de serviços foi implementada como parte da biblioteca que contém a solução para descoberta de serviços, escrita em Haxe[12]. Haxe é uma linguagem de programação de alto nível que pode ser compilada para outras linguagens (*targets*), como Java, C++, Python ou Javascript, o que facilita o reuso da solução. Códigos em Haxe podem também acessar as capacidades e bibliotecas do *target* para o qual são compilados.

A implementação (Figura 3) separa o *core*, responsável pelas funções principais da solução, dos módulos que implementam os mecanismos de distribuição de chaves e descoberta de serviços, e outros dependentes de tecnologias ou protocolos específicos, como o módulo de criptografia. Esses módulos podem ser substituídos pelas aplicações, de acordo com a linguagem de programação, contexto de uso e tecnologias utilizadas. Na implementação realizada, por exemplo, foram criados módulos voltados para NodeJS.

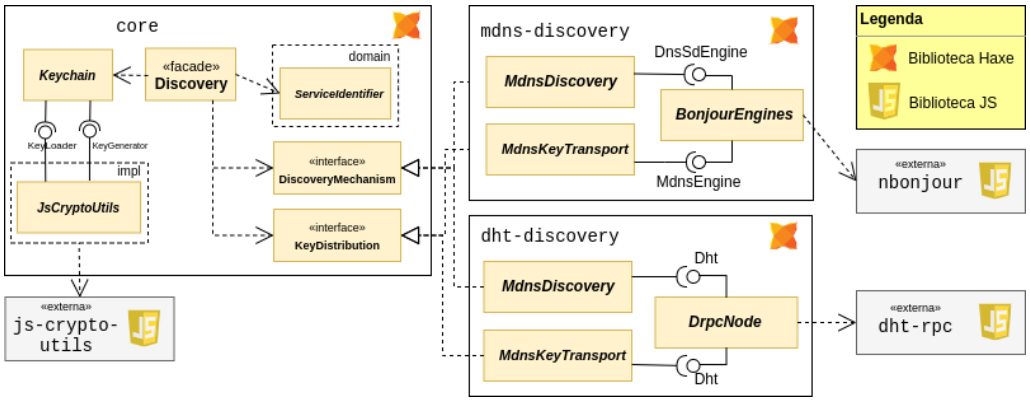

Figura 3. Principais módulos da implementação.

No *core* são implementadas as funções de identidade e autenticação dos serviços, enquanto os mecanismos de descoberta e distribuição de chaves representam e distribuem as chaves criptográficas e descrições de serviços. Para tanto, o *core* assina e verifica as descrições de serviços, seguindo uma ordem padronizada e um modelo que são independentes das representações utilizadas pelos mecanismos. A arquitetura da solução é *peer-to-peer*, então clientes e provedores compartilham o mesmo código, executando a publicação e assinatura, ou descoberta e verificação, de acordo com o papel adotado.

4.6. Avaliação

As funcionalidades da solução de identidade e autenticação proposta foram descritas ao longo deste trabalho. Mas, para analisar a sua viabilidade e da solução de descoberta como um todo, foi conduzido um experimento em um ambiente de redes virtuais, utilizando o emulador CORE[13]. O foco do experimento estava em analisar como a integração de múltiplos mecanismos de descoberta e distribuição de chaves afeta o processo de descoberta. Duas métricas foram analisadas para isso: o tempo de resposta das operações de publicação, busca e localização, e a proporção de serviços encontrados (taxa de sucesso) na busca e localização[14]. A topologia de rede adotada (Figura 4a) buscou simu-

[12] haxe.org.

[13] *Common Open Research Emulator* (CORE), disponível em coreemu.github.io/core/.

[14] Busca corresponde à descoberta de serviços que atendam a uma determinada consulta, enquanto a localização objetiva encontrar a descrição de um serviço, a partir de seu identificador.

lar um conjunto de redes locais conectadas através da Internet. Três principais fatores foram considerados: os números de *hosts* (clientes e provedores), de redes locais e de serviços, resultando em 36 *design points*. A topologia da rede era escalada de acordo com os dois primeiros fatores. Para cada *design point* foram definidas 10 replicações, com as operações a serem executadas (publicação, localização e busca) distribuídas aleatoriamente entre os *hosts*. Dois tipos de chaves foram utilizadas: RSA (com 2048 bits) e curvas elípticas (curva P-256), distribuídas aleatoriamente e em proporções iguais entre os *hosts*. Quatro combinações de mecanismos de descoberta e distribuição de chaves (tratamentos) foram analisados: mDNS, DHT, mDNS+DHT e *mdns-raw* (o mDNS sem as capacidades de identidade, autenticação, etc.). O *mdns-raw* provê um *baseline* para a busca local. Não há, entretanto, um SDP padronizado equivalente que sirva de comparação para os mecanismos baseados na DHT. Cada configuração do experimento foi executada com cada tratamento, i.e. formando blocos de execução pareada dos testes. Para análise dos resultados, a média das métricas em cada execução foi utilizada como unidade de análise. Os efeitos dos fatores foram analisados a partir de modelos de regressão linear *Ordinary Least Squares* (OLS), e os tratamentos comparados através de testes estatísticos[15] paramétricos (ANOVA e TukeyHSD) e não paramétricos (testes de Friedman e de Wilcoxon).

Os resultados obtidos[16] (Figuras 4b e 4c) apontam que, embora a combinação entre os mecanismos heterogêneos (de descoberta e de distribuição de chaves) possa não alcançar métricas ótimas nos cenários analisados, i.e. próximas do valor do melhor mecanismo, também essa combinação não prejudicou os resultados, em relação aos mecanismos com resultados inferiores. O tempo de resposta, por exemplo, tendeu ao mecanismo mais lento na publicação e busca, mas na localização o tempo se manteve entre os tempos do mDNS e DHT. Já a taxa de sucesso, se aproximou do melhor valor. Esses resultados apontam que, embora os mecanismos ainda precisem ser otimizados, a solução proposta é viável, especialmente considerando outros benefícios, como flexibilidade e segurança.

(a) Topologia com M *hosts* em N redes (K=M/N *hosts* em cada rede).

(b) Tempo de resposta x número de *hosts*.

(c) Taxa de sucesso x número de redes.

Figura 4. Topologia de rede e resultados do experimento.

[15]Foi utilizado nível de significância (α) de 0.05. Os testes estatísticos foram calculados a partir das bibliotecas SciPy (scipy.org) e Statsmodels (statsmodels.org).

[16]Por uma questão de espaço, não detalhamos os valores obtidos no experimento. Entretanto, os gráficos incluídos permitem visualizar as tendências das métricas, o que para a análise realizada seria o suficiente.

5. Discussão

A solução para identidade de serviços apresentada neste trabalho provê a base para integrar a descoberta de serviços através de ambientes e SDPs heterogêneos, de forma descentralizada, e com suporte a serviços móveis e multi-dispositivo. Com isso, surgem possibilidades para novos tipos de aplicações, capazes de operar em ambientes dinâmicos e de forma autônoma, como em sistemas ubíquos ou aplicações da IoT. Esta solução também contribui para tornar esses sistemas mais seguros, ao oferecer uma identidade que pode ser verificada a partir de primitivas criptográficas, de modo seguro, descentralizado e transparente para aplicações. Nenhuma das soluções multi-protocolo e multi-ambiente encontradas na literatura (vide Seção 3) oferece essas mesmas capacidades. Entretanto, permanecem ainda limitações e desafios, que podem ser explorados por trabalhos futuros.

Uma limitação na solução apresentada está na ausência de um mecanismo para delegar permissões sobre um serviço. Como a solução identifica provedores a partir de chaves criptográficas, para oferecer serviços através de múltiplos dispositivos, as chaves precisam ser transferidas entre esses dispositivos, o que aumenta as possibilidades de ataques. Certificados poderiam evitar essas transferências, ao delegar autoridade sobre uma identidade para outra chave. Organizações poderiam também utilizar essa função para distribuir responsabilidades. Por sua vez, em ambientes como IoT, em que dispositivos podem não ter recursos computacionais para se autenticar de forma segura [Abreu et al. 2021], seria possível repassar funções para dispositivos mais robustos. Outros desafios relacionados ao gerenciamento de chaves, incluem a definição de meios para: revogar chaves comprometidas e para recuperar identidades caso uma chave seja perdida.

Um outro desafio, conhecido como *Triângulo de Zooko*, consiste em prover nomes que sejam, ao mesmo tempo, "compreensíveis por humanos", "descentralizados" e "seguros" [Wachs et al. 2014b]. A identidade de serviços projetada priorizou os dois últimos requisitos, pois ela não precisaria ser manipulada diretamente por humanos. Entretanto, para prover nomes compreensíveis, a solução proposta pode ser combinada com abordagens como: *petnames*[17], que associam nomes a IDs criptográficos, mas tem validade apenas local; ou *blockchains* como Namecoin[18], que permite um registro global descentralizado de nomes, mas com um custo para manutenção deles [Wachs et al. 2014b].

Oferecer privacidade na descoberta de serviços pode ser também desafiador, especialmente em soluções multi-protocolo e multi-ambiente. A solução de descoberta proposta não requer qualquer informação privativa dos usuários. Ainda assim, a localização de servidores e o tráfego de mensagens podem ser monitorados. É possível definir mecanismos de descoberta e distribuição de chaves, e utilizar protocolos de comunicação que ofereçam privacidade, por exemplo, através de TOR[19]. No entanto, podem ser necessárias novas pesquisas para oferecer privacidade através de ambientes e SDPs heterogêneos.

Um desafio de longo alcance está em alcançar consenso e padronização. A solução proposta ajuda a lidar com essa questão e promover padronização em etapas, através de seus pontos de flexibilidade. Pois permite a integração com padrões emergentes. Por exemplo, podem ser desenvolvidos mecanismos de distribuição de chaves que utilizem DIDs[20] para representar identificadores e descrições de serviços de forma padronizada.

[18] namecoin.org.
[19] torproject.org.
[20] w3.org/TR/did-core.

6. Conclusão

Este trabalho apresentou uma solução para identificar e autenticar serviços de forma descentralizada, a partir do conceito de *self-certifying names*. Essa solução permite integrar serviços descobertos através de ambientes e SDPs heterogêneos, contribuindo também para autonomia, mobilidade, segurança e o suporte a serviços oferecidos por múltiplos dispositivos. Para tanto, a solução, como descrito em detalhes neste artigo, se baseia em três componentes: um *schema* de identidade, baseado no princípio dos *self-certifying names*; um *schema* de autenticação, definido a partir do identificador do serviço; e os mecanismos de distribuição de chaves, que permitem adaptar a forma de compartilhamento de chaves criptográficas, de acordo com o ambiente de execução e o tipo da aplicação. Trabalhos futuros podem estender essa solução para tratar alguns dos desafios discutidos, como privacidade na descoberta de serviços ou gerenciamento de identidades multi-dispositivo e multi-usuário. Uma outra possibilidade seria explorar técnicas de análise de segurança, e.g. [Neto et al. 2021], para verificar a segurança do *schema* de autenticação proposto.

Agradecimentos

Este trabalho foi apoiado pelo Grupo de Pesquisa em Artefatos Físicos de Interação (PAIRG) da Universidade Federal do Rio Grande do Norte (UFRN), e parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) sob processo no. 156239/2019-1. Também agradecemos pelos recursos do Laboratório de Computação Física e Fisiológica do PAIRG (PAIRG L2PC) da UFRN.

Referências

Abreu, V., Santin, A., Viegas, E., Vicentini, C., e Silva, M. (2021). "Gestão de identidade e acesso para dispositivos IoT na smart grid". In "Anais do XXI SBSeg" (p. 1–14).

Achir, M., Abdelli, A., e Mokdad, L. (2020). "A taxonomy of service discovery approaches in IoT". In "Proc. of the WINCOM 2020."

Ahmed, R., e Boutaba, R. (2011). "A survey of distributed search techniques in large scale distributed systems". *IEEE COMST*, v. *13*, n. 2, 150–167.

Ahmed, R., Boutaba, R., Cuervo, F., Iraqi, Y., Li, T., e ... Ziembicki, J. (2005). "Service naming in large-scale and multi-domain networks". *IEEE COMST*, v. *7*, n. 1-4, 38–54.

Almenárez, F., Marín-López, A., Campo, C., e García, C. (2004). "PTM: A pervasive trust management model for dynamic open environments". In "PSPT 2004."

Battaglia, F., e Lo Bello, L. (2018). "A novel JXTA-based architecture for implementing heterogenous Networks of Things". *Computer Communications*, v. *116*, 35–62.

Benet, J. (2019). "IPFS - Content Addressed, Versioned, P2P File System (DRAFT 3)."

Bernstein, D. J. (2006). "Curve25519: New diffie-hellman speed records". In "Proc. of PKC 2006" (Vol. 3958, p. 207–228). Springer.

Brizolara, P. L. S., e Miranda, L. C. d. (2018). "Sustentabilidade de Foto-memórias na Era Digital: Desafios e Oportunidades para a Computação". In "SEMISH 2018."

Cheshire, S., e Krochmal, M. (2013). "DNS-Based Service Discovery." RFC 6763.

Cheshire, S., e Steinberg, D. H. (2005). "Zero Configuration Networking". O'Reilly.

Del Campo, J., Pegueroles, J., e Soriano, M. (2006). "Providing security services in a multiprotocol service discovery system for ubiquitous networks". In "ARES 2006."

Finney, H., Donnerhacke, L., Callas, J., Thayer, R. L., e Shaw, D. (2007). "OpenPGP message format." RFC 4880.

Flores, C., Grace, P., e Blair, G. (2011). "SeDiM: A middleware framework for interoperable service discovery in heterogeneous networks". *ACM TAAS*, v. *6*, n. 1.

Frank, K., Suraci, V., e Mitic, J. (2008). "Personalizable service discovery in pervasive systems". In "Proc. of ICNS 2008" (p. 182–187). doi: 10.1109/ICNS.2008.21

Jones, M., e Sakimura, N. (2015). "JSON web key (JWK) thumbprint." RFC 7638.

Khatibi, E., e Sharifi, M. (2021). "Resource discovery mechanisms in pure unstructured peer-to-peer systems: A comprehensive survey". *Peer-to-Peer Netw Appl*, v. *14*, n. 2.

Lardies, F., Rafael, P., Fernandes, J., Zhang, W., Hansen, K., e Kool, P. (2009). "Deploying Pervasive Web Services over a P2P Overlay". In "WET ICE 2009" (p. 240–245).

Mazières, D., e Kaashoek, M. F. (1998). "Escaping the evils of centralized control with self-certifying pathnames". In "Proc. of the 8th ACM SIGOPS" (p. 118–125). ACM.

Meshkova, E., Riihijärvi, J., Petrova, M., e Mähönen, P. (2008). "A survey on resource discovery mechanisms, peer-to-peer and service discovery frameworks". *Computer Networks*, v. *52*, n. 11, 2097–2128. doi: 10.1016/j.comnet.2008.03.006

Mohammed, F., Ali, A., Al-Ghamdi, A., Alsolami, F., Shamsuddin, S., e Eassa, F. (2020). "Cloud computing services: Taxonomy of discovery approaches and extraction solutions". *SYMMETRY-BASEL*, v. *12*, n. 8.

Neto, A. M., Cunha, Í., e Oliveira, L. (2021). "Uma extensão de framework de análise de protocolos de composibilidade universal para acordo de chaves com autenticação baseado em identidade". In "Anais do XXI SBSeg" (p. 141–154).

Pantazoglou, M., Tsalgatidou, A., e Athanasopoulos, G. (2006). "Discovering web services and JXTA peer-to-peer services in a unified manner". In "ICSOC 2006" (p. 104+).

Pourghebleh, B., Hayyolalam, V., e Anvigh, A. (2020). "Service discovery in the Internet of Things: Review of current trends and research challenges". *Wireless Networks*.

Raverdy, P., Issarny, V., Chibout, R., e Chapelle, A. (2006). "A Multi-Protocol Approach to Service Discovery and Access in Pervasive Environments". In "MobiQuitous'06."

Rivest, R. L., e Lampson, B. (1996). "SDSI - a simple distributed security infrastructure."

Rodrigues, P., Réveillère, L., Bromberg, Y.-D., e Négru, D. (2011). "Scalable and interoperable service discovery for future internet". In "Proc. of M-MPAC 2011."

Siebert, J., Cao, J., Zhou, Y., Wang, M., e Raychoudhury, V. (2007). "Universal adaptor: A novel approach to supporting multi-protocol service discovery in pervasive computing". *Lecture Notes in Computer Science*, v. *4808 LNCS*, 683–693.

Stallings, W. (2014). "Computer Security Concepts". In "Cryptography and Network Security: Principles and Practice" (Sixth ed., p. 7–26). Pearson Education.

Sundramoorthy, V., Hartel, P., e Scholten, J. (2009). "A Taxonomy of Service Discovery Systems". In "Context-Aware Computing and Self-Managing Systems" (p. 43–77).

TorProject. (2017). "Tor Rendezvous Specification - Version 3" (Tech. Rep.).

Wachs, M., Schanzenbach, M., e Grothoff, C. (2014a). "A Censorship-Resistant, Privacy-Enhancing and Fully Decentralized Name System". In "CANS 2014" (p. 127–142).

Wachs, M., Schanzenbach, M., e Grothoff, C. (2014b). "On the Feasibility of a Censorship Resistant Decentralized Name System". In "FPS 2013" (p. 19–30). Springer.

Zarrin, J., Aguiar, R., e Barraca, J. (2018). "Resource discovery for distributed computing systems: A comprehensive survey". *J Parallel Distrib Comput*, v. *113*, 127–166.