Pós-Graduação em Ciência da Computação

Richardson Bruno da Silva Andrade

Reforçando o Controle de Acesso Contra Ataques de Personificação e de Replicação em um Ambiente de Internet das Coisas

Universidade Federal de Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife
2020

Richardson Bruno da Silva Andrade

Reforçando o Controle de Acesso Contra Ataques de Personificação e de Replicação em um Ambiente de Internet das Coisas

Dissertação apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciências da Computação.

Área de concentração: Redes de Computadores.

Orientador: Prof. Dr. José Augusto Suruagy Monteiro.

Recife

2020

Catalogação na fonte
Bibliotecária Monick Raquel Silvestre da S. Portes, CRB4-1217

A553r Andrade, Richardson Bruno da Silva
Reforçando o controle de acesso contra ataques de personificação e de replicação em um ambiente de internet das coisas / Richardson Bruno da Silva Andrade. – 2020.
238 f.: il., fig., tab.

Orientador: José Augusto Suruagy Monteiro.
Dissertação (Mestrado) – Universidade Federal de Pernambuco. CIn, Ciência da Computação, Recife, 2020.
Inclui referências e apêndices.

1. Redes de computadores. 2. Internet das coisas. I. Monteiro, José Augusto Suruagy (orientador). II. Título.

004.6 CDD (23. ed.) UFPE - CCEN 2020 - 119

Richardson Bruno da Silva Andrade

Reforçando o Controle de Acesso Contra Ataques de Personificação e de Replicação em um Ambiente de Internet das Coisas

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciências da Computação da Universidade Federal de Pernambuco, como requisito parcial para a obtenção do título de Mestre em Ciências da Computação.

Aprovada em: 13/02/2020.

BANCA EXAMINADORA

_____

Prof^{o}. Dr^{o}. Paulo André Silva Gonçalves
Centro de Informática / UFPE

_____

Prof^{a}. Dr^{a}. Michelle Silva Wangham
Mestrado em Computação Aplicada / UNIVALI

_____

Prof^{o}. Dr^{o}. José Augusto Suruagy Monteiro
Centro de Informática / UFPE
(Orientador)

À querida vó Maria de Lima Silva (*in memoriam*), eterna saudades.

AGRADECIMENTOS

Gostaria de agradecer a todas as pessoas que diretamente ou indiretamente contribuíram nessa jornada. Gostaria de agradecer a Deus por ter me dado oportunidade e a responsabilidade de assumir o desenvolvimento dessa dissertação de mestrado, além de possibilitar ânimo, determinação e fé para desenvolver esse projeto na minha vida.

Em segundo lugar quero agradecer aos meus pais, tios, e irmã pelo amor incondicional, cuidado zelo e apoiadores nos dias de estudo e de pesquisa. Quero agradecer também à minha namorada pelo carinho, apoio e o compartilhamento de seu conhecimento.

Em terceiro lugar, quero agradecer ao orientador pelos questionamentos, pelas críticas acerca do objeto de investigação, nas sugestões de melhoria na formulação do problema, na elaboração das hipóteses e no delineamento dos objetivos da pesquisa, pela parceria nas correções e nas dicas de metodologia de pesquisa científica para a dissertação. Além disso, sobretudo, de como expor o conhecimento adquirido de maneira lógica, clara, objetiva e didática durante o desenvolvimento da pesquisa.

Em último, àqueles que estiveram presentes no meu dia a dia, como os integrantes do grupo de pesquisa, pelas indagações e na troca de conhecimento; do pessoal do laboratório de pesquisa, pela empatia, colaboração e hospitalidade; dos professores das disciplinas da pós, pela atenção, aos anseios das dúvidas; da portaria, pela liberação das catracas e na entrega de correspondência; dos funcionários terceirizados, pela limpeza nas instalações físicas; do pessoal do suporte e da oficina do CIN pelo atendimento aos chamados de suporte para configuração e disponibilização de recursos de informática e no suporte tecnológico e digital.

RESUMO

Internet of Things (IoT) está se disseminando nas casas devido ao baixo custo do hardware e ao aumento da capacidade de processamento embarcado, dando maior conforto e conveniência por meio de uma variedade de dispositivos de automação residencial. Entretanto, sem os devidos mecanismos de segurança, esses dispositivos podem ser controlados, sem o consentimento do dono do dispositivo, por meio de aplicações de terceiros instaladas em *smartphones*, *tablets* ou assistentes virtuais. O ACE-OAuth age no controle de acesso ao recurso sobre aplicações de terceiros, no seu fluxo de código de autorização. No entanto, o ACE-OAuth não garante que a origem das requisições do cliente seja verificada e a identidade do cliente seja confirmada pelo servidor de autorização, comprometendo a confiança na comunicação entre as partes. Soluções como chaves públicas e privadas e tunelamento têm um alto custo computacional em termos de maior consumo de banda e na latência em redes privadas. A adoção de soluções mais simples tais como a inclusão do método desafio-resposta e a introdução de *claims* para informar e verificar a identidade das aplicações, nesta proposta, permitiu mitigar as ameaças e ataques de personificação e de replicação durante a fase de autenticação e autorização. Os resultados da avaliação de segurança demonstraram que as contramedidas propostas reduziram a atuação das ameaças em torno de 30% pontos percentuais e diminuiram os riscos de ataque em cerca de 11% para um atacante amador e 16% para um perfil de atacante especialista. Desse modo, a proposta obteve sucesso em reforçar a segurança do controle de acesso no ambiente de IoT de modo eficiente e eficaz.

Palavras-chave: Ambientes Restritos. Autenticação. Autorização. Controle de Acesso. Internet das Coisas. OAuth.

ABSTRACT

Internet of Things (IoT) is spreading in homes due to hardware low costs and the increase of the capacity of embedded processing, giving more comfort and convenience by means of a variety of automation home devices. However, without appropriate security mechanisms, these devices can be controlled, without their owner's consent by means of third-party apps installed in smartphones, tablets, or virtual assistants. ACE-OAuth acts in the resources access control over third-party apps in IoT environments by the use of an authorization code grant flow. However, ACE-OAuth does not guarantee that client's request source is verified and that client's identity is confirmed by the authorization server, compromising a trusted communication between parties. Solutions such as the use of public/private keys and tunneling have a high computational cost in terms of latency and bandwidth consumption in private networks. The adoption of simpler solutions such as the inclusion of a challenge-response method and the introduction of claims to inform and to assert the apps' identity, in this proposal, it was able to mitigate personification and replay attacks during ACE-OAuth's authentication and authorization phase. The results of the security evaluation demonstrated that the proposed countermeasures decreased threats by around 30 percentage points and reduced the risks of attacks around 11% for an amateur attacker and 16% for an expert attacker when compared with the original flow's countermeasures. Thus, the proposal reinforced security efficiently and effectively in IoT access control environments.

Keywords: Constrained Environments. Authentication. Authorization. Access Control. Internet of Things. OAuth.

LISTA DE CÓDIGOS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ACE	*Authentication and Authoriazation for Constrained Environment*
API	*Application Programming Interface*
ARP	*Address Resolution Protocol*
CBOR	*Concise Binary Object Representation*
CoAP	*Constraint Aplication Protocol*
COSE	CBOR *Object Signing and Encryption*
CWT	CBOR *Web Token*
DNS	*Domain Name System*
DTLS	*Datagram Transport Layer Security*
FIDO	*Fast IDentity Online*
HTML	*Hypertext Markup Language*
HTTP	*Hyper Text Transfer Protocol*
HTTPS	*Hyper Text Transfer Protocol Secure*
IANA	*Internet Assigned Numbers Authority*
IEEE	*Institute of Electrical and Electronics Engineers*
IETF	*Internet Engineering Task Force*
IoT	*Internet of Things*
JSON	JavaScript Object Notation
JWE	*JSON Web Encryption*
JWS	*JSON Web Signature*
JWT	JSON Web *Token*
LAN	*Local Area Network*
OAuth	*Open Authorization*
OIDC	OpenID Connect
PKI	*Public Key Infrastructure*
POP	*Proof-of-Possession*
REST	*Representational State Transfer*
RFC	*Request for Comments*
RFID	*Radio Frequency Identification*
SOAP	*Simple Object Access Protocol*
SSL	*Socket Security Layer*
TCP	*Transmission Control Protocol*
TLS	*Transport Layer Security*
U2F	*Universal Second Factor*
UDP	*User Datagram Protocol*
UFA	*Universal Authentication Framework*
URI	*Uniform Resource Identifier*
URL	*Uniform Resource Locator*
Wi-Fi	*Wireless-Fidelity*

WoT	*Web of Things*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

“Conhecimento é poder. A informação é libertadora. A educação é a premissa de progresso, em toda sociedade, em toda família.”

Kofi Annan, diplomata ganês.

Neste capítulo são apresentados os conceitos básicos sobre controle de acesso e os processos de autenticação e de autorização e como eles atuam na defesa contra aplicações não autorizadas em ambiente de IoT. Na motivação é enfatizada a evolução e a importância da aplicação do controle de acesso em ambientes de IoT. Na problematização, são apontadas as principais causas dos ataques de personificação e replicação em ambientes de IoT. No objetivo, são definidas a solução proposta e a questão da pesquisa a ser validada. E, por último, é mostrado o procedimento metodológico adotado neste trabalho.

1.1 CONTROLE DE ACESSO, AUTENTICAÇÃO E AUTORIZAÇÃO

O controle de acesso é um alicerce para a segurança geral de qualquer sistema de comunicação (BELTRAN; SKARMETA, 2018), já que o sistema de comunicação concede ou proíbe o acesso aos seus recursos ou objetos baseado em vários critérios. Além disso, o controle de acesso cria um nível de confiança aos usuários legítimos (BERTIN et al., 2019).

É sabido que, por exemplo, a fim de estabelecer o controle de acesso numa residência, é fundamental estabelecer a verificação de identificação, processo pelo qual uma informação em torno de uma pessoa é reunida e usada para oferecer algum nível de confiança (MADSEN, 2016).

Tipicamente, para realizar a verificação de identificação, um porteiro poderia solicitar alguma credencial, como um documento oficial, e conferir se o documento apresentado é da pessoa que a porta e apresenta antes de liberar a entrada na residência. Dessa maneira, a credencial também é algo que o usuário é (*fingerprint*, padrão de retina, de voz ou da face), tem (bracelete, *smarthphone*, *token*) ou sabe (identificador, *password*, ou PIN), (MADSEN, 2016) e, que ele alega ser,

verdadeiramente, de sua posse no processo de autenticação ou de autorização (SHIREY, 2007a).

Nesse sentido, por meio do uso de uma credencial, o usuário consegue provar quem ele diz ser. Isso é conhecido como processo de autenticação (BERTIN et al., 2019), Figura 1.

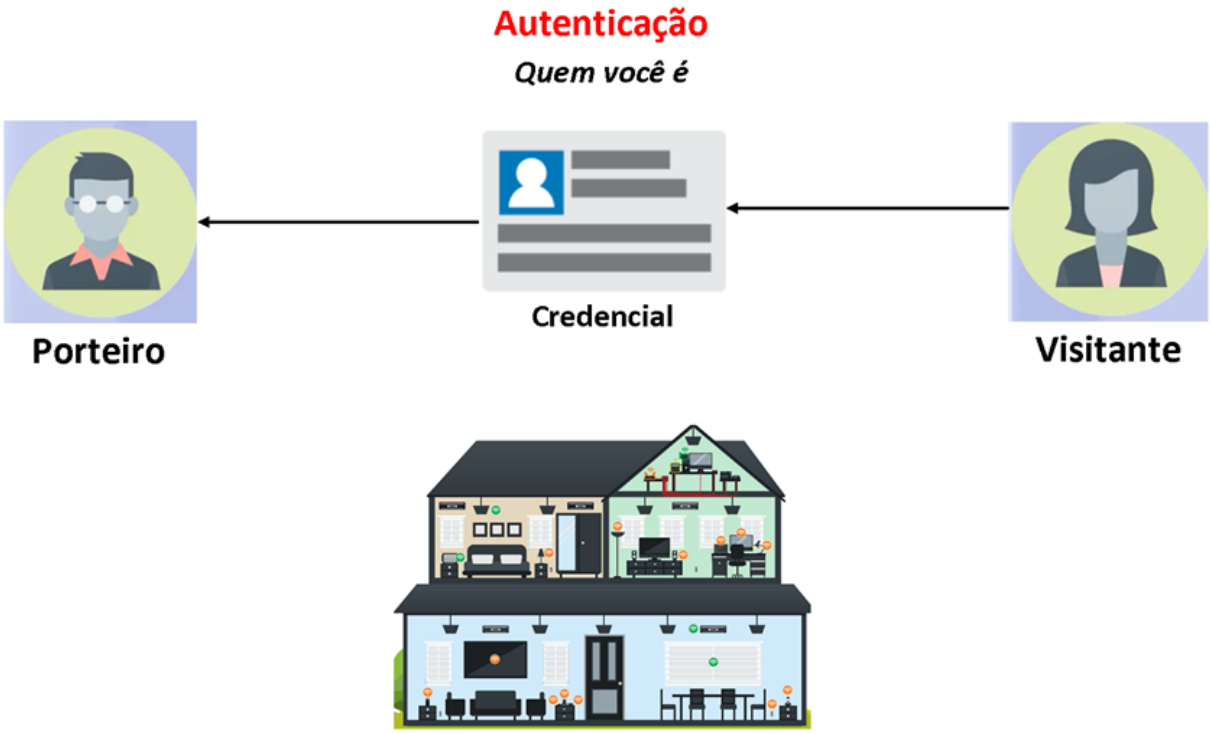

Figura 1 – Autenticação.

Por outro lado, o processo de autorização surge, por exemplo, quando o morador deixa instruções específicas sobre o que o visitante pode mexer ou não na moradia. Dessa forma, a autorização verifica quem, uma vez identificado, tem permissão para fazer algo (ABOMHARA; KØIEN, 2014), Figura 2.

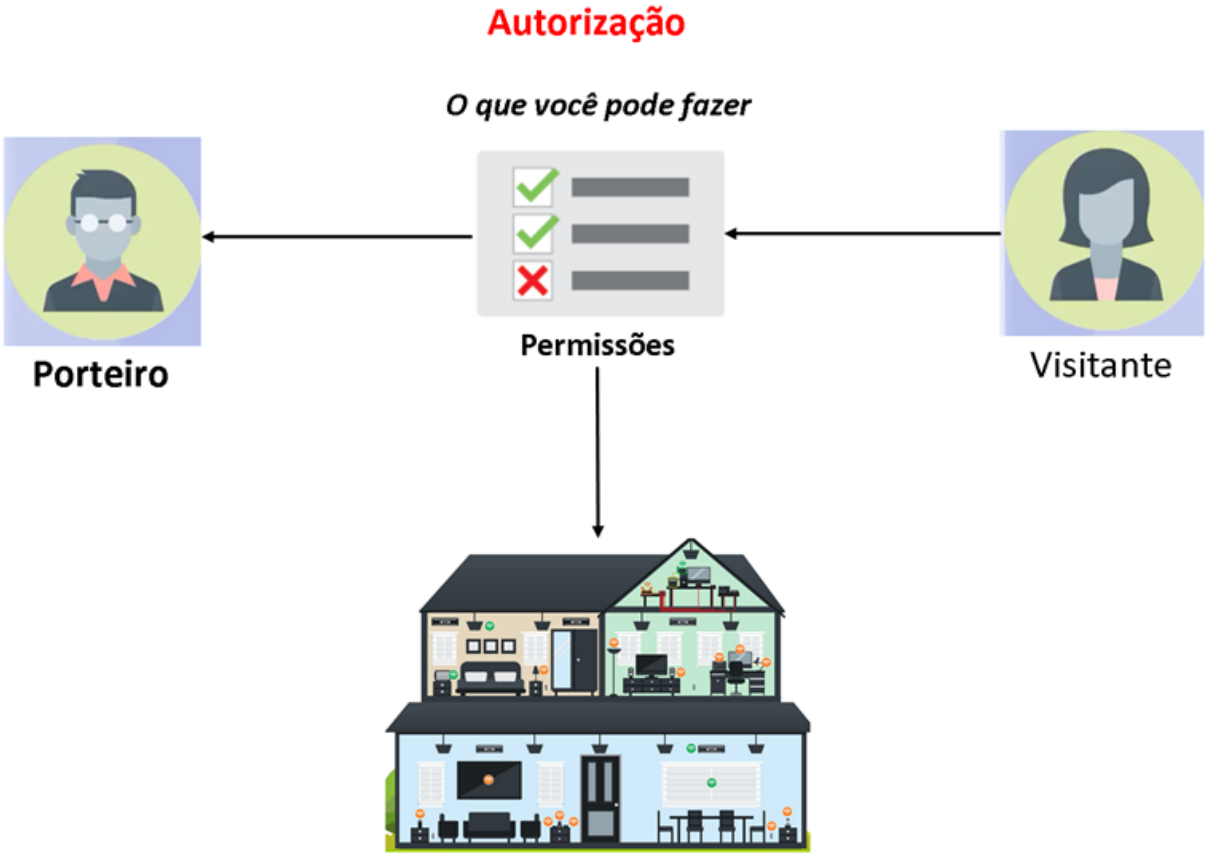

Figura 2 – Autorização.

Em um ambiente com Internet das Coisas (*Internet of Things,* IoT), como de automação residencial, aplicações de terceiros são utilizadas para acessar recursos e acionar as funcionalidades dos dispositivos de IoT. Estas aplicações consistem em *softwares* que não foram desenvolvidas seguindo as exigências de desenvolvimento pelo mesmo fornecedor oficial do dispositivo, do serviço, do sistema ou da plataforma (MIDRACK, 2019). A imposição do controle de acesso permite restringir o modo de atuação das aplicações de terceiros em ambiente de IoT (BERTIN et al., 2019).

De modo geral, entretanto, muitas pessoas podem se sentir desconfortáveis sobre compartilhar os recursos com uma terceira parte e por isso vão querer confidencialidade no acesso aos recursos (SCULLY; MCDONALD; MCEVOY, 2018). Além disso, as pessoas vão querer uma garantia de que seus recursos sejam compartilhados a partir de seu consentimento (SCULLY; MCDONALD; MCEVOY, 2018).

Por essa razão, a imposição do controle de acesso permite restringir o modo de atuação das aplicações de terceiros em ambiente de IoT (BERTIN et al., 2019).

Os controles de acesso tradicionais a aplicações de terceiros, por exemplo, em sistemas Web, utilizam processos de autenticação e autorização baseados no uso de credenciais, no entanto, não foram projetados para oferecer serviços de confidencialidade ou de integridade a fim de proteger contra a exposição de dados sensíveis (HARDT, 2012).

Dessa maneira, o uso inseguro e ineficaz da credencial no controle de acesso a aplicações de terceiros abre um leque de ameaças e dá margem a possíveis ataques de personificação e ataques de replicação em ambientes de IoT (DEEP; ZHENG; HAMEY, 2019). Em casas inteligentes, esses tipos de ataques são classificados como nível de risco um, já que possuem um custo/esforço baixo para o ataque assim como um custo/esforço baixo em relação à segurança (TSCHOFENIG; BACCELLI, 2019) e, dessa maneira, podem comprometer com maior facilidade a segurança dos objetos ou dos recursos que precisam ser protegidos.

1.2 MOTIVAÇÃO

O modo de realizar o controle de acesso aos recursos na Internet, especificamente em IoT, é uma constante preocupação de segurança nas aplicações, nos sistemas e nos serviços (BERTIN et al., 2019).

Tipicamente, na arquitetura cliente-servidor, por exemplo, o servidor provê arquivos, documentos, multimídia e serviços e os clientes que consomem esses recursos. Por conseguinte, nesses ambientes descentralizados, no lado cliente, aplicações de terceiros podem realizar o acesso aos recursos de forma ineficaz e insegura, ao expor suas as credenciais a cada solicitação (LI, 2017) (LODDERSTEDT; MCGLOIN; HUNT, 2013).

Dessa maneira, como não há garantia de que as aplicações de terceiros utilizem mecanismos suficientes para proteger as credenciais, os sistemas que detêm os recursos precisam realizar um controle de acesso mais rígido ou realizar atribuição de um nível de privilégio mínimo. Por exemplo, algumas aplicações clientes de terceiros podem conseguir acesso irrestrito aos recursos protegidos do usuário ou poderiam, voluntariamente ou não, expor outras informações sigilosas, como as credenciais, para partes mal-intencionadas (HARDT, 2012).

Esses problemas apresentados devido ao uso inseguro e ineficaz no controle de acesso de aplicações de terceiros estão presentes no ambiente de automação residencial em IoT. Nesse ambiente, as aplicações de terceiros são instaladas nos

dispositivos (ex. *smartphones*, *tablets*, ou *hubs* inteligentes) que realizam acesso aos dados gerados pelos sensores ou controle sobre as funcionalidades dos dispositivos de IoT (EL-HAJJ et al., 2019).

Considerado isso, algumas soluções descentralizadas para Web e móvel possibilitam reduzir os riscos associados aos problemas gerados pelo controle de acesso nas aplicações de terceiros, entre elas o OAuth 2.0[1], por meio da concessão de autorização baseada em permissão sobre atributos (HARDT, 2012). No entanto, o OAuth 2.0 foi projetado para oferecer autorização contando com alguma solução pré-existente de autenticação.

No domínio de IoT, uma das estratégias para desenvolver, no domínio de IoT, uma das estratégias para desenvolver soluções, consiste em reusar padrões tradicionais na Web e móveis. Contudo, para a IoT abraçar a autorização descentralizada, são necessários novos modelos adaptados para atender a natureza dos dispositivos de IoT (BELTRAN; SKARMETA, 2018). Além disso, como o ambiente de IoT possui restrições de memória, processamento e energia, as soluções tradicionais nem sempre se adequam perfeitamente às necessidades de um ambiente de IoT (ECHEVERRÍA; KLINEDINST; SEITZ, 2019).

Dessa forma, o modelo adaptado para ambientes de IoT, o *framework de Autenticação e Autorização para Ambientes Restritos por meio de Autenticação Aberta (Authentication and Authorization Environments by Open Authentication,* ACE-OAuth)[2] foi desenvolvido. O ACE-OAuth funciona como uma extensão do OAuth 2.0, se preocupando com as restrições de recursos impostos pelo ambiente de IoT (SEITZ et al., 2020). Entretanto, o ACE-OAuth também herda as mesmas ameaças causadas pelo uso inseguro e ineficaz das credenciais no OAuth 2.0 (LODDERSTEDT; MCGLOIN; HUNT, 2013) e dessa maneira o ACE-OAuth está suscetível aos ataques de personificação e de replicação.

1.3 PROBLEMATIZAÇÃO

Segundo, o relatório “Tentativas de Aquisição Hostil” (AKAMAI, 2020), entre 2017 e 2019, um em cada cinco tentativas, no total de 85 bilhões de ataques, ganharam acesso não autorizado a contas ou recursos de usuários. Esses acessos

[1] *The* OAuth 2.0 *Authorization Framework*: https://tools.ietf.org/html/rfc6749
[2] *Authentication and Authorization for Constrained Environments* (ACE) *using the* OAuth 2.0 *Framework* (ACE-OAuth): https://datatracker.ietf.org/doc/draft-ietf-ace-oauth-authz/

não autorizados ocorrem, por exemplo, porque não há taxa limite para tentativas de autenticação, o que permite que atacantes executem palpites de combinação a fim de ganhar acesso. Nessa ocasião, é preciso identificar se os clientes estão usando credenciais de forma ilegítima (ECHEVERRÍA; KLINEDINST; SEITZ, 2019).

Ainda assim, neste mesmo relatório, constata-se que o acesso não autorizado vem ocorrendo por meio da integração com APIs baseadas em estilos arquiteturais de cunho distribuído, como REST ou SOAP (AKAMAI, 2020). E com isso os reflexos do problema do acesso não autorizado decorrente da integração de APIs estendem ao domínio de IoT, já que grande parte dos sistemas ou aplicações IoT faz uso desses estilos arquiteturais mencionados, na construção e disponibilização de APIs para plataformas de IoT.

No ambiente de IoT, o ACE-OAuth é uma das possíveis soluções para controle de acesso aos recursos de IoT sobre aplicações de terceiros. Para estabelecer e gerenciar uma comunicação segura, o ACE-OAuth estabelece requisitos como confidencialidade, proteção de integridade e autenticação mútua sobre a comunicação entre os participantes (BELTRAN; SKARMETA, 2018). Entretanto, o ACE-OAuth não põe qualquer controle eficaz sobre a comunicação entre aplicação cliente e a entidade de autorização, o que em tese deveria ser protegida a *priori, já que* há a emissão das credenciais confidenciais nessa etapa (BELTRAN; SKARMETA, 2018). Soluções encontradas na literatura, baseadas no ACE-OAuth, também não trataram a proteção das credenciais na comunicação entre aplicação cliente e a entidade de autorização (SCIANCALEPORE et al., 2017) (ARAGON et al., 2018), (FREMANTLE; AZIZ, 2018), (SEITZ et al., 2020).

Outras possíveis limitações e desvantagens também decorrem da aplicação de técnicas, práticas, processos métodos, restrições de domínio e soluções antigas no uso da credencial, o que torna todos esses mecanismos inseguros no controle de acesso em ambiente de IoT, como por exemplo:

- No lado cliente, a aplicação de terceiros é responsável pela criação, atualização e uso da credencial;
- A má prática de armazenar a credencial no dispositivo do cliente para ações futuras;
- Uma mesma credencial é utilizada em diferentes *endpoints* do fluxo. Esses *endpoints* são interfaces de comunicação entre as partes que permitem a troca de dados;

- Nem sempre é possível garantir o tráfego da credencial em canal seguro;
- Métodos de autorização alguma vezes usam uma camada com criptografia ou assinatura digital sobre a mensagem; em vez disso, usa-se codificação hexadecimal na tentativa de confundir ou dificultar, embora seja inseguro;
- A utilização de mecanismos tradicionais de segurança exige grande consumo de recursos computacionais e de armazenamento;
- A existência de ameaças e de ataques de personificação e de replicação.

Dessa forma, a hipótese definida para ser respondida na investigação deste trabalho é: “a solução proposta elaborada reduz o risco de ataques de personificação e de replicação em ambiente de IoT do que o fluxo original no ACE-OAuth?”.

1.4 OBJETIVOS

O objetivo deste trabalho é reforçar a segurança do fluxo de concessão do código de autorização no ACE-OAuth, já que há chance de acontecer uma exposição da credencial na comunicação entre as partes em um ambiente de IoT. Diante disso, a finalidade do objetivo deste trabalho é mitigar, principalmente, os ataques de personificação e replicação oriundos da exposição das credenciais nas trocas de mensagens. Os objetivos específicos da proposta são:

- Representar a identificação do cliente feita nas solicitações por meio de um conjunto de atributos afirmativos para ser apresentado pela aplicação cliente à autoridade de autorização;
- Economizar o custo computacional do uso da prova de posse que é apresentada pelo cliente ao servidor de recurso para acessar o recurso;
- Reduzir o atraso nas solicitações de credenciais e de recursos pelo cliente, em consequência da aplicação de mecanismos de segurança, que influenciam o tempo de resposta durante as trocas de mensagens entre as partes;
- Diminuir o tamanho das mensagens por meio do protocolo de transferência e codificação serializada dos dados para suavizar as trocas de mensagens;
- Identificar qual combinação dos cenários da solução proposta possui um custo aceitável, mas preservando requisitos mínimos de segurança.

1.5 SOLUÇÃO

Como forma de melhorar o controle de acesso em ambiente de IoT, este trabalho propõe o reforço na segurança no fluxo de concessão código de autorização o qual é utilizado no controle de acesso do ACE-OAuth. Para isso, a solução proposta focou nas fragilidades de confirmação de origem e de identificação das aplicações as quais ocasionam ameaças e ataques de personificação e replicação em ambientes de IoT. Entretanto, como contramedidas às ameaças e aos ataques, a solução reforçou o fluxo original ao adotar dois outros fluxos: o fluxo de desafio-resposta e fluxo de veracidade por meio da estrutura de identificação ID Token.

A solução proposta neste trabalho pode contribuir com o amadurecimento da implementação do ACE-OAuth, destacando a importância de proteger as solicitações do fluxo de comunicação código de autorização, o qual é comumente usado para aplicações Web, mas também pode ser usado em aplicações para *smartphones* e *tablets* em ambientes em IoT (SEITZ et al., 2020).

1.6 METODOLOGIA

Nesta seção, é classificada a metodologia a ser utilizada na pesquisa e é apresentada uma síntese dos procedimentos metodológicos utilizados para o desenvolvimento deste trabalho.

1.6.1 Metodologia da Pesquisa

O método adotado nesta pesquisa foi o método indutivo. O método indutivo visa criar afirmações generalizadas sobre as sentenças particulares utilizadas nas conclusões sobre o fenômeno investigado (LAKATOS, 2003). Dessa forma, este trabalho determinou um conjunto limitado de premissas para ser experimentadas em um ambiente controlado a fim de obter resultados verdadeiros e, consequentemente, desenvolver uma proposta boa e próxima a fim de atingir grande parte dos casos observados da problemática.

Além disso, caracterizou-se a metodologia científica adotada neste trabalho, em três classes: o da natureza, o da abordagem de trabalho e o dos objetivos (GIL, 2002).

Primeiro, acerca da natureza, este trabalho correspondeu a uma pesquisa aplicada, pois introduziu o procedimento racional e sistematizado, esquematizando a partir dos seguintes elementos do projeto de pesquisa: levantamento da bibliografia, delimitação e formulação do problema, delineamento dos objetivos, solução proposta, levantamento e teste de hipóteses e, análise e interpretação de dados. Segundo, em relação à abordagem de trabalho, este trabalho é de caráter quantitativo, pois busca investigar o impacto dos fatores adotados nos experimentos sobre o fenômeno investigado por meio de uma avaliação de desempenho e confirmar a hipótese principal por meio de uma avaliação de segurança. Por fim, em relação aos objetivos, este trabalho consistiu em uma pesquisa exploratória, porque visou prospectar o problema e formular novas hipóteses ou amadurecer hipóteses já existentes em tema ainda pouco investigado, na forma de pesquisa bibliográfica e mais fortemente com uma abordagem prática.

1.6.2 Procedimentos Metodológicos

Nesta subseção é apresentada uma síntese metodológica utilizada no desenvolvimento deste trabalho.

Revisão bibliográfica. Na revisão bibliográfica foram pesquisados trabalhos científicos, livros, artigos e especificações dos protocolos relacionados com o tema deste trabalho que cobrem os tópicos de automação residencial, ACE-OAuth e segurança, a fim de servir como base teórica para as etapas seguintes e contextualização em relação à problematização do objeto de investigação e o desenvolvimento de perfil de comunicação para o *framework* ACE-OAuth.

A revisão bibliográfica foi realizada usando serviço *online* de bases de fontes bibliográficas como o Google Scholar (GS)[3], a base de Elsevier[4] e na do IEEE[5], assim como nas especificações do IETF[6]. Essas fontes bibliográficas possuem alta qualidade e relevâncias nos assuntos cobertos e grande número de referências. Todos os resultados das pesquisas foram organizados de acordo com relação ao tema.

As publicações para serem selecionadas deveriam se encontrar no intervalo

[3] Google Scholar https://scholar.google.com.br/scholar
[4] Elsevier *Journals*: https://www.journals.elsevier.com/
[5] IEEE Xplore: https://ieeexplore.ieee.org/Xplore/home.jsp
[6] IETF *Document Search*: https://datatracker.ietf.org/doc/search

entre 2015 e 2019 e serem relacionadas aos aspectos de automação residencial, OAuth 2.0, e ACE-OAuth. Ademais, outras publicações mais antigas foram adotadas, porém, somente publicações que contivessem referências aos protocolos ou às abordagens mencionados pelo ACE-OAuth.

Os termos usados na pesquisa bibliográfica foram: “*Home Automation*”, “*Smart Home*”, “*Authorization*’ “OAuth”, “ACE-OAuth”, “IoT”, “*Security*” e “*third-party apps*”. De modo geral, uma das possíveis combinações de *queries* utilizada na busca da literatura foi “*Smart Home*” AND “*Authorization*” AND ‘IoT” AND “*Security*”.

Entendimento dos conceitos. Deu-se ênfase aos termos-chave e aos aspectos que estruturam o *framework* ACE-OAuth e especificações afins. Esses conceitos foram fundamentais para a elaboração da proposta. Além disso, outras definições foram investigadas para que pudessem ser adicionadas à proposta como tipos de solução baseado na estrutura ID Token para autenticação.

Identificação dos problemas. Foi realizada a análise dos tipos de ameaças e ataques, identificação de aspectos não cobertos ou não solucionados nos trabalhos relacionados sobre o ambiente de IoT e o conhecimento dos novos desafios encontrados e identificação dos problemas em abertos sobre autenticação e autorização na automação residencial.

Projeto da solução proposta. Foram identificados quais problemas deveriam ser atacados, a definição dos fluxos de comunicação a serem adotados, a escolha dos protocolos de comunicação e o mecanismo de segurança.

Implementação. Foi desenvolvido um protótipo como prova de conceito da solução proposta.

Avaliação de Desempenho. Foram realizados experimentos em duas plataformas separadas como local e em nuvem com e sem estabelecimento de canal de segurança a fim de identificar qual desses fatores impacta na solução proposta. Além disso, foram considerados nos experimentos a combinação dos cenários, o tamanho das mensagens e o tempo levado para consumir os serviços.

Avaliação de Segurança. Foi aplicada uma avaliação do modelo de ameaça de sobre modelos de comunicação do fluxo código de autorização com reforço contida na proposta e do mesmo fluxo, original, sem reforço. Além disso, foram realizadas a identificação e análise de riscos baseada em árvore de ataques para ter estimativas dos ataques mais fáceis de serem realizados pelo atacante e desenvolver medidas ou mecanismos de defesa melhores.

1.7 ESTRUTURA DO TEXTO

A estrutura desse trabalho está descrita da seguinte maneira: o capítulo 2 deste documento apresenta os fundamentos teóricos como IoT, princípios de segurança, a diferenciação entre autenticação, autorização e controle de acesso OAuth e ACE-OAuth. O capítulo 3 apresenta os trabalhos relacionados ao objeto de investigação deste trabalho. O capítulo 4 apresenta uma visão geral da solução proposta, o perfil de comunicação e segurança adotado, o modelo de comunicação, as considerações de segurança e as medidas contra os ataques de personificação e replicação e avaliação de segurança sobre a proposta por meio de uma avaliação de não conformidade a ameaças sobre modelo de comunicação e a identificação das vulnerabilidades e avaliação de riscos. O capítulo 5 apresenta a avaliação de desempenho sobre o sistema investigado, os serviços, as métricas e os parâmetros adotados na avaliação, a elaboração do projeto de experimentos, a técnica de avalição e os cenários utilizados e, por fim, a análise e a avaliação dos resultados. Por fim, no capítulo 6 é apresentada a discussão e a conclusão, assim como as contribuições deste trabalho e propostas para trabalhos futuros.

2 Fundamentação Teórica

“Agarre-se a seus sonhos, pois, se eles morrerem, a vida será como um pássaro de asa quebrada, incapaz de voar”.

Autor da citação, algum feito incrível do sujeito.

Neste capítulo é apresentada uma visão geral dos conceitos básicos e gerais de aspectos como Internet das Coisas (*Internet of Things*, IoT), dispositivos, automação residencial, princípios da segurança da informação, noções de controle de acesso, criptografia, OAuth 2.0, ACE-OAuth e gerenciamento de identidade.

2.1 INTERNET DAS COISAS

A IoT pode ser compreendida como uma infraestrutura de rede global, configurável, escalável e com a capacidade de expansão dinâmica baseada sobre a padronização e os protocolos de comunicação interoperáveis.

Diante disso, o principal objetivo da IoT é a interconexão de várias redes heterogêneas que permitem colecionar dados, compartilhar, analisar e gerenciar recursos. Para Lin et al (2017), a IoT é a uma arquitetura horizontal que deve integrar camadas de comunicação de diversas aplicações.

Entretanto, a origem da IoT surge a partir do desenvolvimento de Identificação por Rádio Frequência (*Radio Frequency Identification*, RFID) que consiste de um identificador universal o qual oferece uma identidade única para cada objeto físico, cunhada pelos membros do centro Auto-ID do Instituto de Tecnologia de Massachusetts (*Massachusetts Institute of Technology*, MIT).

Vale destacar que, segundo Borgia (2014), uma coisa (*thing*) de IoT é algo que se move no tempo e no espaço e que pode ser identificado unicamente facilmente legível, reconhecível, localizável, endereçável e controlável via Internet. Desse modo, a IoT torna-se ubíqua e onipresente, já que permite as pessoas estarem conectadas em qualquer lugar, em qualquer com coisa, em qualquer tempo, em qualquer rede e usando qualquer serviço.

2.1.1 Dispositivos de IoT

Um dispositivo de IoT é objeto físico ou virtual com capacidades reduzidas ou com escassez de recursos como processamento memória, energia e comunicação. Esses dispositivos realizam periodicamente o sensoriamento do ambiente e envia os dados coletados para o controlador central como um *hub*.

Além do mais, muitos desses dispositivos de IoT se comunicam por conexão sem fio, oferecendo aos habitantes o controle remoto sobre as funcionalidades do dispositivo, por exemplo, em um ambiente residencial (SCHIEFER, 2015). Por exemplo, travas da porta, lâmpadas inteligentes e termostatos, necessitam serem acessados todos os dias pelos usuários usando seus dispositivos como *smartphones*, *touch-panels* e *tablets* (TSCHOFENIG, 2016), Figura 3.

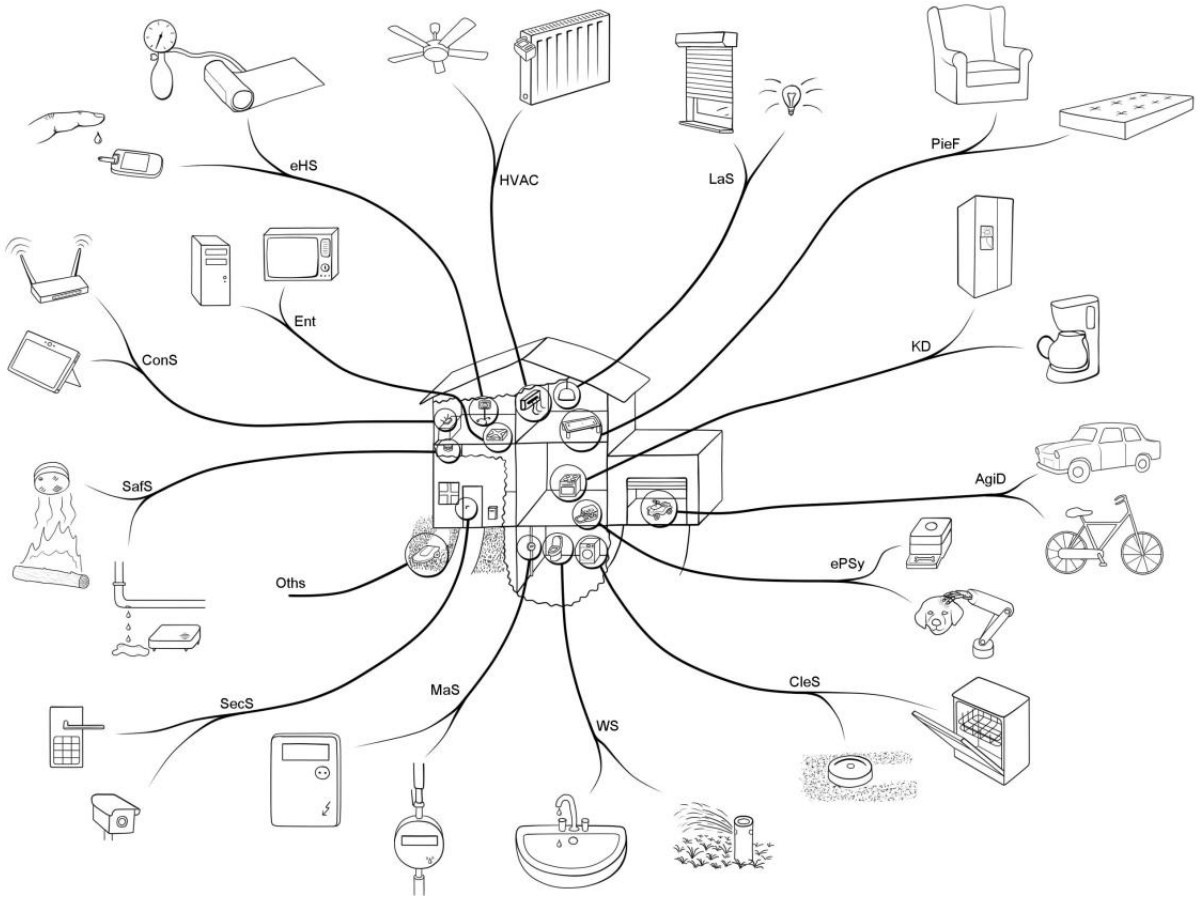

Figura 3 – Representação da categorização dos dispositivos de automação residencial.

Fonte: adaptado de (SCHIEFER, 2015).

Diante da variedade de dispositivos para uso da automação residencial, Schiefer (2015) realizou uma categorização dos dispositivos de automação

residencial, Tabela 3.

Tabela 3 – Uma maneira de categorizar os dispositivos da casa inteligente. Fonte: Adaptado de (SCHIEFER, 2015).

Categoria	Funções dos Dispositivos	Exemplo
Sistemas de Controle	Controlam a residência	Estação ou sistemas, *tablet* ou *smartphone*
Sistemas de Segurança	Protegem o patrimônio de seus recursos	Sistemas de travas, câmeras de segurança
Sistemas de Risco de Vida	Detectam e evitam ameaças em relação à condição física e à vida	Detecção de gás, incêndio, terremoto
Sistemas e-saúde	Inspeção ou assistência médica	Medidor de glicose, aplicador de insulina, medidor pressão arterial, sensor de oxigênio
Sensores e medidores	Medidores de água e de eletricidade	Medidores de água e de eletricidade, sensores e atuadores discretos
Aquecedores, ventilação e ar-condicionado	Regulam a temperatura ou ventilação do ar	Termostato e ventilador
Luz e sombra	Emitem ou reduzem à luz	Lâmpadas, coberturas solar e cortinas
Cozinha	Usados na cozinha	Geladeira, refrigerador, cafeteira
Sistemas de água	Usados no jardim	Torneira, cano do banheiro, banheiro, chuveiro e irrigador de gramado
Sistemas de limpeza	Lavam louça, roupas ou retiram poeira	Lavador de roupa, lava-louças e robôs aspiradores de pó
Sistemas de e-pet	Para localização e bem-estar dos animais de estimação	Colar para localizar, robô para acariciar ou sistema de alimentação automatizado

Categoria	Funções dos Dispositivos	Exemplo
Entretenimento	Para diversão e lazer	Sistemas de áudio, televisão, consoles de jogos, robô de brinquedos
Peças de mobiliário	Para sentar e dormir	Mesas, armários e massageadores de colchão e sofá
Dispositivos ágeis	Para transporte de pessoas	Carros, bicicletas e gadgets de bicicleta

Já outros dispositivos de IoT ajudam melhor entender a rotina do usuário, isso por conta da opção extra por meio do suporte a assistentes virtuais com inteligência artificial embarcada como Google *Assistent*[7], Alexa[8] e Siri (HILT et al., 2019). Esses assistentes virtuais permitem controlar os dispositivos IoT por voz o que dispensa a necessidade de um dispositivo de controle como o celular (HILT et al., 2019). Há três gerações de casas inteligentes (LI, 2016):

- 1ª geração - tecnologia *wireless* e servidor *proxy* com abordagem de automação residencial.
- 2ª geração - Inteligência artificial nos dispositivos que controlam os eletrodomésticos.
- 3ª geração - robô amigo que pode interagir com seres humanos.

Modo de Controle. Segundo Tschofenig (2016), o acesso pode acontecer ou indiretamente via infraestrutura da nuvem ou diretamente através do uso de tecnologias ondas de rádio de curto alcance. No entanto, para Ali & Awad (2018) os sistemas que controlam as atividades da automação residencial podem ser divididos:

- **Local**. O sistema utiliza um controlador sob medida para realizar a automação. Ex.: um painel instalado fisicamente.
- **Remoto**. O sistema controlado remotamente por meio de conexão com a Internet. Ex.: aplicação instalada no *smartphone*.

[7] Google *Home*: https://store.google.com/?srp=/product/google_home
[8] Alexa *Smart Home*: https://amzn.to/3bIR2kt

2.1.2 Automação Residencial ou Casa Inteligente

Como pode ser visto na Figura 4, dentre dos diversos domínios no universo de IoT, encontra-se o domínio de automação residencial ou de casa inteligente.

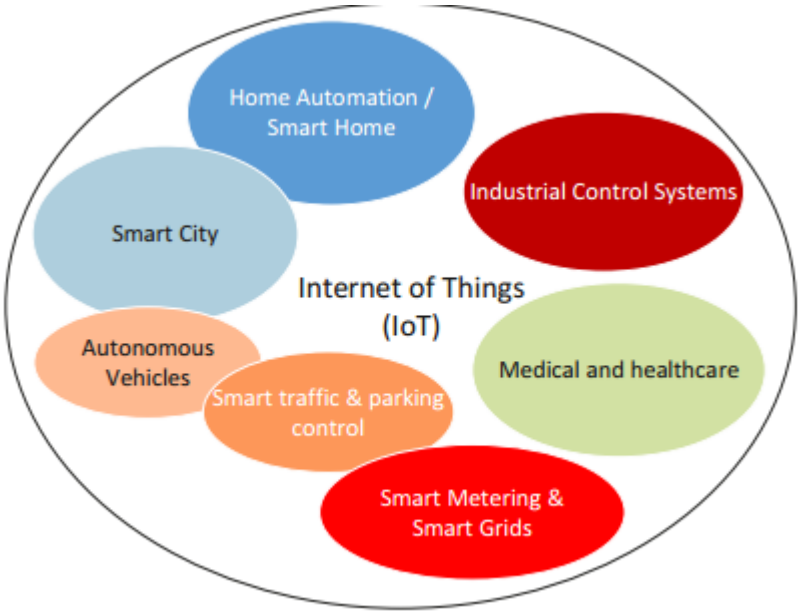

Figura 4 – Domínios de aplicação de IoT.
Fonte: (ANGRISHI, 2017).

O termo casa inteligente pode significar uma residência privada quem contém vários dispositivos inteligentes de IoT que automatizam diversas tarefas repetitivas do cotidiano.

Segundo Stojkoska & Trivodalie (2017), uma visão holística da casa inteligente é compreendida como um sistema de gerenciamento que é composto pelos dispositivos, *hubs*, nuvem e aplicações de terceiros.

Estrutura. Cada casa possui uma Rede de Sensores Móvel (*Wireless, Sensor Networking*, WSN) e os dados sensoriados nessa rede por meio do dispositivo são encaminhados para uma estação central, como por exemplo, concentrador (*hub*) da casa. O *hub* pode ser qualquer dispositivo (o mais comum é roteador ou o assistente virtual, mas poderia ser também *desktop*, *tablet*, *notebook*, *smartphone*). Comumente, os dispositivos centrais têm uma maior capacidade de processamento memória e consumo de energia. Outro componente considerado que faz parte da automação residencial é a nuvem a qual trata do armazenamento do grande volume de dados gerados e do processamento complexo.

Plataformas de automação de IoT. Para atender alta demanda de solicitações dos dispositivos de IoT é necessária a estrutura a comunicação em de plataformas de automação de IoT. De acordo com Hilt et al. (2019) há três tipos de servidor para cada tipo de plataformas de automação de IoT :

- **Local**: o servidor instalado e executado em dispositivos *standalone* como Raspberry PI, Arduino, computador *desktop* ou *notebook* e como comunicação via Wi-Fi ou Ethernet.
- **Baseado em cloud**: serviços que eram disponibilizados mais próximos do usuário são migrados para infraestrutura em nuvem, adicionando capacidade, escalabilidade, flexibilidade e integração com outros serviços na *Web*.
- **Assistentes virtuais**: corresponde aos mais populares assistentes virtuais como Amazon Lexa, Google Assistant e Apple Siri que estão instalados num dispositivo que está fisicamente posicionado no interior da residência e atua como concentrador da comunicação entre os dispositivos em seu entorno e faz conexão externa com a nuvem.

2.2 PRINCÍPIOS DE SEGURANÇA DA INFORMAÇÃO

Nesta seção, são cobertos termos segurança da informação, a diferença entre segurança e privacidade, serviços de segurança e modelos de ataque.

2.2.1 Segurança

Segundo Previtali (2016), a segurança pode ser compreendida de dois modos diferentes:

- **Segurança (*security*)**: É a condição de o sistema ser protegido do acesso não intencionado ou não autorizado, mudança ou destruição.
- **Segurança (*safety*)**: É a condição onde o sistema opera sem causar inaceitável risco de contusão física ou dano para saúde da pessoa, diretamente ou indiretamente como um resultado de dano para propriedade ou para o ambiente.

2.2.2 Privacidade

Segundo Previtali (2016), a privacidade diz respeito ao direito do indivíduo de controlar ou influenciar o uso de informações a seu respeito. Essas informações podem ser coletadas e armazenadas por quem oferece uma solução de uso do indivíduo e podem ser compartilhadas com terceiros.

Outro termo que está intimamente ligado com a privacidade é a confidencialidade de dados. O serviço de confidencialidade significa que o dado não é exposto para entidades do sistema a menos que eles tenham sido autorizados para ter acesso ao dado.

2.2.3 Definições de Segurança

Segundo Coelho et al. (2014) alguns conceitos e termos que contribuem na aplicação de controle de segurança são:

- **Ativo**: Qualquer coisa que tenha valor. Ex.: dispositivos, banco de dados, dados de pessoas, processos e serviços.
- **Ameaça**: Qualquer evento que explore vulnerabilidades ou causa potencial de um incidente indesejado que pode resultar em dano. Ex.: deixar a chave da casa no vizinho.
- **Vulnerabilidade**: Qualquer fraqueza que possa ser explorada e comprometer a segurança de sistema ou informações. Ex.: configurações incorretas funcionais ou de segurança geram vulnerabilidades.
- **Risco**: Combinação da chance da ameaça se concretizar dado um evento e de suas consequências para a pessoa, o serviço e a organização.
- **Ataque**: É um ato deliberado de tentar de se desviar dos controles de segurança com objetivo de explorar as vulnerabilidades. Há dois modelos de ataques: o ataque passivo e o ataque ativo.
    - O ataque **passivo** faz o uso de escutas e monitoramento de transmissões com intuito de obter informações que estão sendo transmitidas. Ex.: A escuta de uma conversa telefônica.
    - O ataque **ativo** envolve a modificação dos dados, criação de informações falsificadas ou negação de serviço.

2.2.4 Serviços de Segurança

Os principais serviços de segurança ou objetivos de segurança são autenticidade, confidencialidade, integridade, disponibilidade, não-repúdio e controle de acesso.

Autenticidade. Implica ser genuíno, capaz de ser verificado e confiável (STALLINGS, 2014). Verifica o usuário é quem diz ser e que tem sua origem de uma fonte confiável possibilitando comunicação segura. O objetivo da autenticidade é confirmar outra parte quem alega ser realmente autêntica. Ex.: assinatura digital.

Confidencialidade. É uma propriedade onde o agente autorizado tem acesso e divulgação a informação, impondo por meio de meios de proteção a privacidade (STALLINGS, 2014). A confidencialidade tem objetivo de proteger os dados transmitidos contra ataques passivos e dos acessos não autorizados (COELHO; ARAÚJO; BEZERRA, 2014). Ex. a mensagem é encapsulada com criptografia ou transmitida por estabelecimento do canal de segurança.

Integridade. Prevenção contra a modificação ou destruição imprópria de informação (STALLINGS, 2014). Desse modo, a integridade visa combater ataques ativos por meio de alterações ou remoções não autorizadas da informação. Ex: uma sequência aleatória é gerada e aplicada ao conteúdo da mensagem a ser enviado. O receptor compara a sequência gerada pelo emissor com resultado do cálculo sobre o conteúdo recebido. Além disso, a integridade também é um pré-requisito para os outros serviços. Ex.: Se a integridade do controle de acesso do sistema operacional for violada, também será a violada a confidencialidade de seus serviços.

Disponibilidade. Assegurar o acesso e o uso rápido e confiável da informação (STALLINGS, 2014). A disponibilidade determina quais os recursos são aptos para o acesso das entidades autorizadas, sempre que solicitados, a fim de proteger contra perdas e degradações (COELHO; ARAÚJO; BEZERRA, 2014). Ex. servidor de aplicações críticas, que deve está ativo permanentemente, deve reagir por evento conhecido e ser operado por usuário previamente reconhecido pelo sistema.

Não-Repúdio. Também conhecido como irretratabilidade, impede que o emissor ou o receptor neguem uma mensagem transmitida (STALLINGS, 2014). Quando dada mensagem é enviada, o destino pode provar que esta foi realmente enviada por determinada origem e vice-versa (COELHO; ARAÚJO; BEZERRA, 2014). O não-repúdio não permite que uma das partes da comunicação refute a declaração de ser verdadeiramente o responsável pelo envio dos dados (SAYANA; JOSHI, 2016).

Controle de acesso. É a capacidade de limitar e dominar o acesso aos sistemas e aplicações por meio dos *links* de comunicação (STALLINGS, 2014). Para realizar p controle de acesso, cada indivíduo precisa ser identificado (autenticação) de modo que os direitos de acessos possam ser ajustados ao perfil do indivíduo (autorização) na tentativa realizar o acesso. Dessa forma, o controle de acesso possibilita conceder ou revogar o acesso sobre algo ou sobre determinada ação. Uma descrição mais detalhada se encontra na seção (2.5).

2.2.5 Modelos de Ataque

Os modelos de ataques compreendidos na análise e avaliação de segurança são (COELHO; ARAÚJO; BEZERRA, 2014):

- **Interrupção**: Quando um ativo é destruído ou torna-se indisponível caracterizando um ataque contra disponibilidade. Ex.: negação de serviço ou destruição de um disco rígido;
- **Interceptação**: Quando um ativo é acessado por uma parte não autorizada, caracterizando um ataque contra confidencialidade. Ex.: cópia não autorizada de arquivo ou programas;
- **Modificação**: Quando um ativo é acessado por uma parte não autorizada e ainda é alterado, caracterizando um ataque contra integridade. Ex.: alterar os valores do arquivo de dados confidencial e com acesso restrito;
- **Fabricação**: Quando um ativo é acessado por uma parte não autorizada e insere informação falsa, caracterizando um ataque contra autenticidade. Ex.: adição de registros em um arquivo.

2.3 AUTENTICAÇÃO

A autenticação é o processo de verificar uma identidade é alegada por ou para uma entidade do sistema (STALLINGS, 2014). O processo de identificação consiste em duas etapas a etapa de identificação e de verificação. A etapa de identificação ocorre no momento em que o sujeito apresenta um identificador ao sistema de controle. Já na etapa de verificação, confirma-se o vínculo entre a identidade e o identificador. Os donos das credenciais terão que passar através de um processo de registro, onde os donos das credenciais atribuem ou são atribuídos por um único identificador próprio junto com algum segredo. As credenciais usualmente se destinam a ser usadas mais de uma vez (Ex.: de uso das credenciais são um distintivo de policial, uma carteira de motorista de automóvel e um passaporte nacional). De todo modo, Rischer & Sanso (2017) identificaram quatro tipos de paradigmas envolvendo as credenciais, que são credenciais iguais, credenciais diferentes, credencial universal e credencial especial.

Por outro lado, de acordo com Ali et al. (2017), a autenticação realiza a verificação sobre a comunicação entre partes, em que uma das partes reivindica ser verdadeiramente o autor da mensagem enviada. Dessa forma, a autenticação consiste na apresentação da credencial pelo seu dono e a verificação da credencial por outra parte ou autoridade.

Além do mais, no contexto de comunicação de rede, a autenticação descreve a verificação da identidade de uma parte envolvida na comunicação como, por exemplo, no processo de acesso no serviço na Web em que o cliente envia a sua credencial em texto claro ou criptografado ou usando um protocolo de autenticação para realizar um acesso remoto no servidor o qual verifica a identidade do cliente (GERBER, 2018).

De qualquer forma, a autenticação oferece diversas técnicas aprimoradas por meio da credencial que o usuário pode usar como desafio-resposta, chaves compartilhadas, centro de distribuição de chaves, certificados e autenticação de *host* (RESCORLA, 2005)

2.4 AUTORIZAÇÃO

Segundo Rescorla (2005) a autorização é o processo pelo qual um sujeito determina quando uma parte autenticada tem permissão para acessar um recurso particular ou serviço. Contudo, pode haver múltiplos escopos ou níveis de acesso sobre o recurso, então, é necessário limitar o acesso de uma parte envolvida para esse recurso (GERBER, 2018).

Embora os termos autenticação e autorização sejam confundidos, a autenticação identifica simplesmente uma parte, enquanto que autorização define quando uma parte pode realizar uma determinada ação, Figura 5.

Figura 5 – Diferença entre autorização e controle de autenticação.
Fonte: (LEVIN, 2019).

2.5 CONTROLE DE ACESSO

O controle de acesso também é um serviço de segurança. O controle de acesso também limita e controla o acesso lógico/físico aos ativos de uma organização por meio dos processos de identificação, autenticação e autorização (COELHO; ARAÚJO; BEZERRA, 2014). Os objetivos do controle de acesso são de verificar se uma entidade que solicita o acesso a um recurso tem os direitos necessários e proteger os recursos protegidos contra acessos não autorizados.

O problema de controle de acesso dos serviços ou das aplicações tem sido extensivamente tratado na literatura. Vários trabalhos se concentraram em como implementar diferentes estratégias de controle de acesso. Para a discussão da proposta deste trabalho é foi dado ênfase no controle de acesso baseado em

atributos (*Attribute-based Access Control*, ABAC).

Controle de acesso baseado em atributos. O controle de acesso concede ou revoga as requisições baseadas sobre atributos do sujeito ou dos recursos e condição do ambiente. Por meio do ABAC, a mudança nos atributos ocorre dinamicamente, tal como os atributos de tempo e de localização podem ser usado na tomada de decisão do controle de acesso (BERTIN et al., 2019). Propostas como OAuth 2.0 e ACE-OAuth se baseiam nesse modelo de controle de acesso baseado em atributos. A seção (2.7) detalha os aspectos do OAuth 2.0 e a seção (2.8) detalha os aspectos do ACE-OAuth 2.0.

Controles de Acesso sobre Aplicações de Terceiros. De qualquer modo, em IoT, o controle de acesso aos recursos atua sobre as aplicações de terceiros. Desse jeito, de acordo com Stojkoska & Trivodalie (2017), deve haver uma preocupação e atenção na aplicação de segurança nas aplicações de terceiros assim como ocorre nos dispositivos, *hub* e *cloud* como mostrado na Figura 6. Nesse sentido, El-Hajj et al. (2019) investigaram os controles de acesso utilizados pelas plataformas de soluções de IoT que podem ser aplicados sobre as aplicações de terceiros, no Apêndice E.

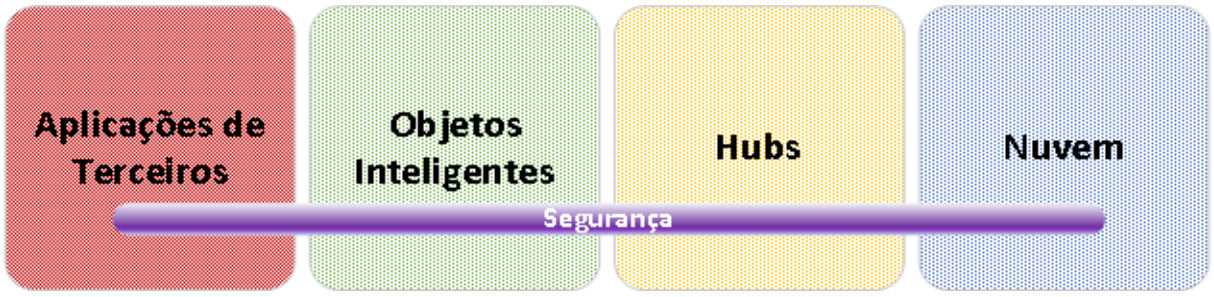

Figura 6 – Segurança dos componentes de IoT.
Fonte: (STOJKOSKA; TRIVODALIE, 2017).

Entretanto, mecanismos de controle de acesso na Web puderam ser estendidos para (*Web of Things*, WoT) mas que ainda a aplicação de terceiros utilizam métodos de autenticação ultrapassados, baseados em nome do usuário e/ou segredo, em protocolos de transferência de dados (HTTP, CoAP, XAMPP, MQTT) como Autenticação Básica (*Basic Autenticação*), Chave da API (API *Key*) e Sumarização (*Digest*) e Autorização do Portador (*Bearer Authorization*)

(BOTYRIUTE, 2018).

Autenticação básica. Recomenda-se raramente devido às vulnerabilidades de segurança, já que o remetente põe a credencial dentro do cabeçalho da requisição, a qual pode ser visualizada ao analisar o conteúdo da requisição, Figura 7. Além do que, mesmo a credencial é codificada/decodificada em hexadecimal (Base64[9]), não garante nenhuma os dados são protegidos, pois basta utilizar um decodificador[10] na base hexadecimal para obter o texto original. Por outro lado, a autorização básica não requer gerenciamento de sessão, armazenamento de dados temporários, preenchimento de formulário e não precisa de *handshakes* ou protocolo de resposta complexo.

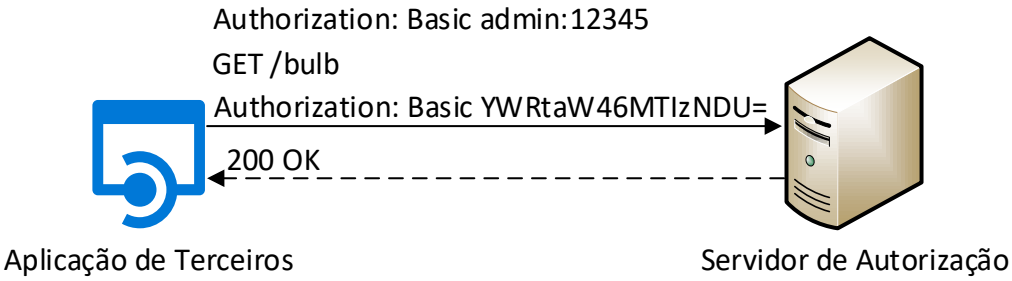

Figura 7 – Exemplo de autenticação básica.

Chave da API. Resolve os problemas de autorização básica HTTP. A chave de API gera um único valor que é atribuído no primeiro acesso para cada usuário a partir da combinação de alguns dados (ex.: hardware, IP) e em outras vezes, a chave de API é gerada aleatoriamente pelo servidor, Figura 8. A chave de API é enviada comumente como parte da URL da solicitação, mas pode aparecer em outros lugares, como no corpo da mesma. Em relação à segurança, a falta de máscara para essa chave, facilita na identificação e captura da credencial.

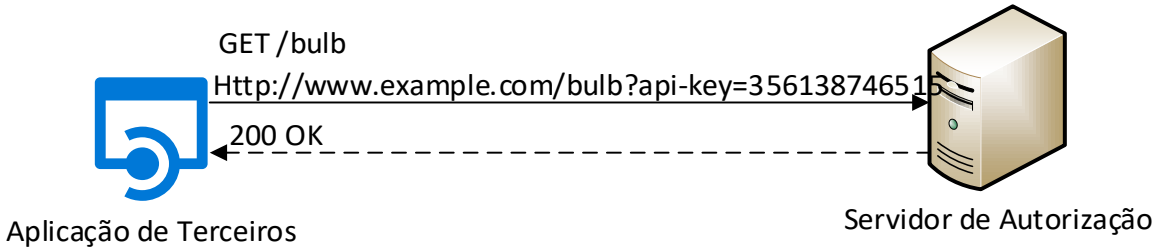

Figura 8 – Exemplo de chave de API.

[9] Base64: https://en.wikipedia.org/wiki/Base64.
[10] Decodificado na base hexadecimal: https://www.base64decode.org/.

Portador do token. Usa o esquema de autenticação que envolve a utilização de *tokens*. Autorização do portador concede o acesso para o portador deste *token* Figura 9. Além disso, a autorização do portador permite acesso aos recursos. Para que autorização do portador ocorra é necessário que o *token* seja enviado no cabeçalho da mensagem quando se realiza requisição de acesso os recursos. Além do mais, a autorização do portador deve ser a partir do estabelecimento do canal de segurança (HTTPS).

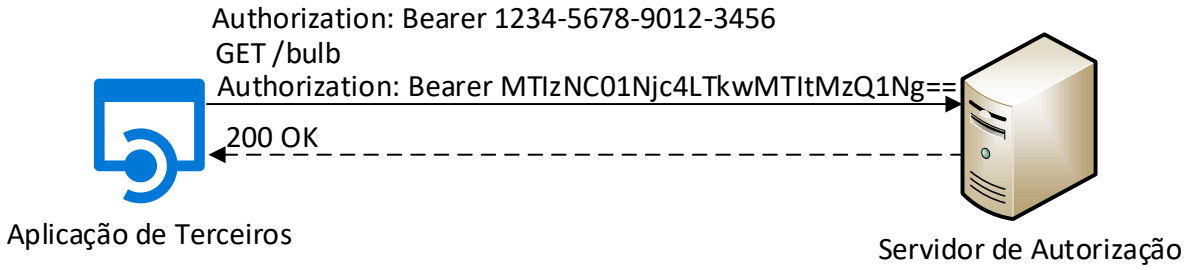

Figura 9 – Exemplo de autorização do portador.

Sumarização. É uma técnica de compactação de mensagem que usa um algoritmo de resumo sobre a mensagem a fim de representar a origem particular da requisição, Figura 10. O *digest* usa codificação na base 64. De qualquer forma, mesmo usando a sumarização, a mensagem é transmitida no formato em texto claro, o que também torna inseguro. Além de que, os dados sumarizados por determinados algoritmos de sumarização já podem ser quebrados num intervalo de tempo reduzido[11]. Para que a transmissão seja protegida é necessário estabelecer um canal de segurança (HTTPS).

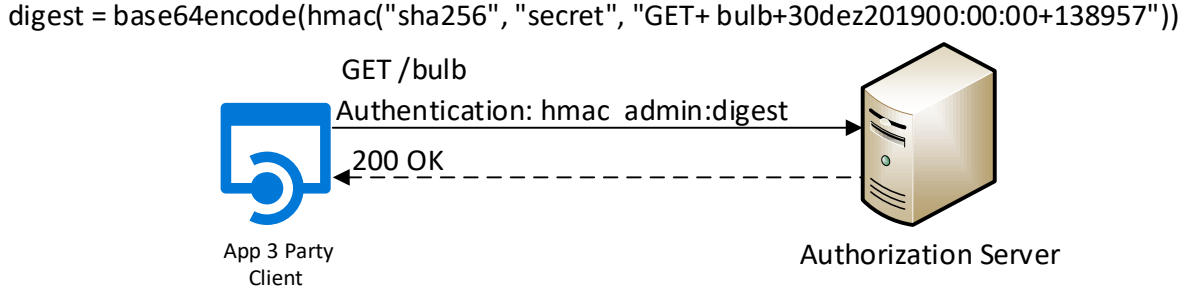

Figura 10 – Exemplo de sumarização.

[11] *Hash Cracker*: https://hashkiller.io/

O resumo das quatro técnicas utilizadas pelas aplicações é apresentado na Tabela 1:

Tabela 1 – Comparação entre os controles de acesso adotados pelas aplicações
Fonte: próprio autor.

Características	Autorização Básica	Chave API	Autorização do Portador	*Digest*
Encodificação Base 64	✗	✗	✓	✓
Tipo de estrutura da credencial	Nome do usuário e segredo	chave	*token*	Nome do usuário, segredo, tempo ou outro dado
Credencial gerada aleatoriamente	✗	✓	✗	✓
Criptografado	✗	✗	✗	✗

2.6 CRIPTOGRAFIA

Operações e métodos de criptografias reforçam a proteção de aplicações e sistemas. Por isso, é imprescindível compreender alguns termos centrais referentes aos aspectos de criptografia na segurança, tais como a diferenciação entre criptografia e descriptografia, entre criptografia simétrica e assimétrica, entre texto claro e texto cifrado, além de termos comuns como cifra, chave criptográfica e *hash*.

- **Texto claro**: Uma mensagem legível e clara;
- **Cifra**: Um algoritmo que executa criptografia;
- **Texto cifrado**: O resultado final do processo de criptografia;
- **Criptografia**: O processo de conversão de dados, como mensagens ou informações em texto claro para texto cifrado;
- **Descriptografia**: O processo de conversão de dados, como mensagens ou informações de texto cifrado para texto claro.
- **Chave criptográfica**: Uma cadeia de bits, senha ou similar que é usada pela criptografia para criptografar o texto simples;
- ***Hash***: Uma entidade criptografada que codifica os dados de maneira que não possa ser descriptografada ou recuperada.

Em geral há três tipos principais de mecanismos de criptografia: a criptografia simétrica e criptografia assimétrica e funções de *hash* criptográficas (EUROPOL, 2019).

Criptografia Simétrica. Usa-se uma mesma chave para encriptar um texto não criptografado pelo remetente e para decriptar o texto cifrado pelo destinatário. Para isso, a mesma chave secreta é compartilhada ou conhecida pelas ambas às partes como mostra a Figura 11.

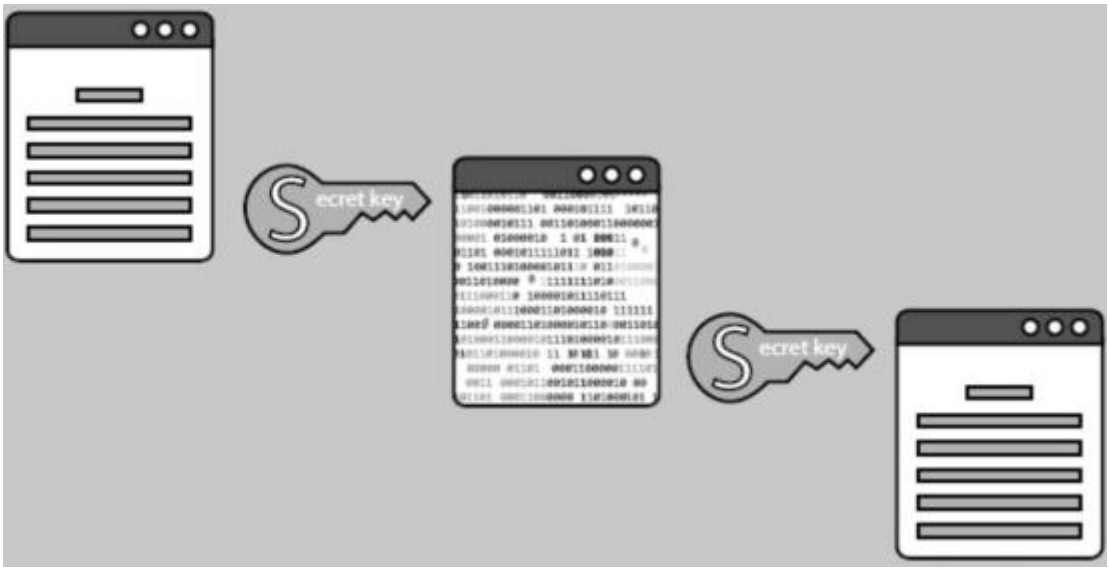

Figura 11 – Criptografia simétrica.
Fonte: (EUROPOL, 2019).

Criptografia assimétrica. Conhecida como criptografia da chave pública é baseada na combinação de chave pública e privada. O remetente usa a chave pública do destinatário para encriptar o texto não cifrado e apenas o destinatário pode decriptar o texto cifrado usando sua chave privada associada à sua chave pública, Figura 12. Por meio do uso da criptografia assimétrica, a chave pública garante a autenticidade e a chave privada assegura a integridade e confidencialidade.

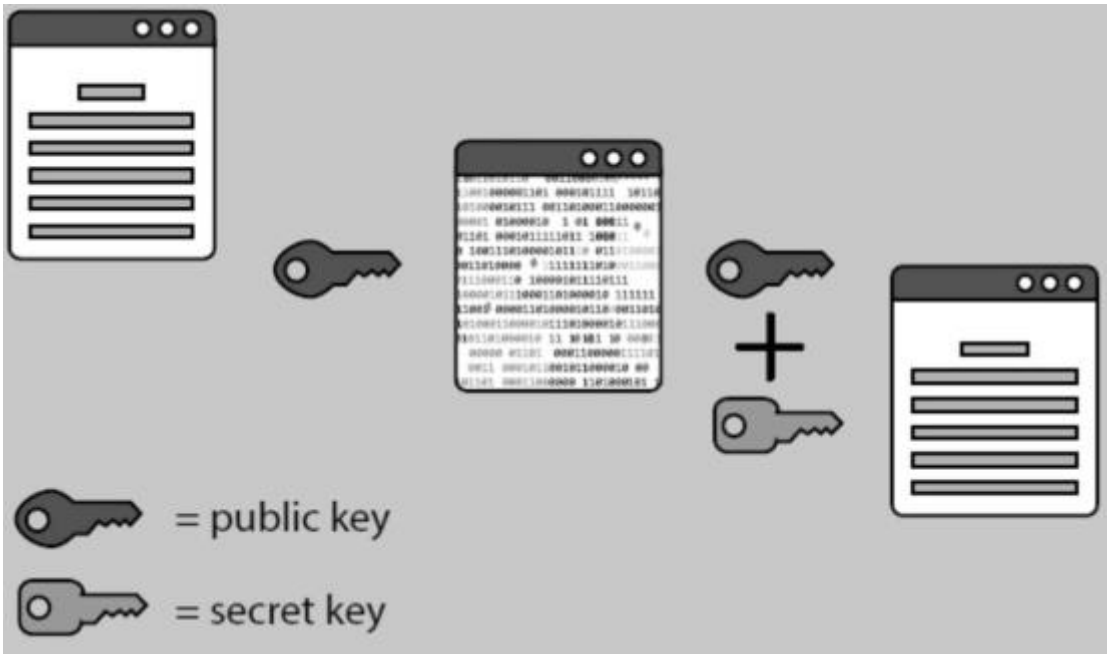

Figura 12 – Criptografia assimétrica.

Fonte: (EUROPOL, 2019).

Função *hash*[12]. Conhecida como sumarização (*digest)* da mensagem ou encriptação de um caminho, é uma função que aplica um algoritmo sobre os dados para produzir um determinado resultado a partir de mecanismos de uso de chave *hash* para autenticação de mensagens (*Keyed-Hashing for Message Authentication)* HMAC[13]. O uso da função *hash* é muito comum para checar a integridade do conteúdo e também auxilia na verificação da origem de solicitação.

2.7 OPEN AUTHORIZATION 2.0

Nesta seção, será mostrado o *framework* de autorização OAuth 2.0. Inicialmente será apresentada a definição, algumas características e as partes envolvidas que participam do processo de autenticação e autorização assim como os tipos de fluxo de comunicação utilizado na proposta.

2.7.1 Definição

De acordo com (HARDT, 2012), *Open Authentication* (OAuth) 2.0, é um *framework* que permite que aplicações ou serviços de terceiros obtenham acesso limitado a um serviço, em nome do proprietário do recurso o qual concede autorização de acesso ao serviço requisitante sobre o recurso protegido.

Dentre as características do OAuth 2.0 encontram-se:

- **Utilidade**: Embora exija mais esforço para implementar com segurança, o OAuth 2.0 é bastante utilizável;
- **Flexibilidade**: Considera também como clientes as aplicações móveis, além das aplicações *Web*;
- **Desacoplamento das responsabilidades**: Melhor separação das tarefas e dissociação entre a manipulação de solicitações de recursos e a autorização do usuário;
- **Centrado em torno dos *tokens* do portador**: Os *tokens* de portador são fáceis de implementar e integrar, mas não excelentes para fins segurança

[12] https://tools.ietf.org/html/rfc4949#page-140
[13] https://www.rfc-editor.org/rfc/rfc2104.html

porque os *tokens* do portador podem ser facilmente copiados ou sequestrados;

- **Dependência de transferência segura**: A maioria das defesas de segurança é delegada ao canal de segurança e, no OAuth 2.0, o canal de segurança serve para proteger os *tokens* do portador.

De todo modo, para compreender o OAuth 2.0, a analogia apresentada por (SAKIMURA, 2017) mostra como o OAuth 2.0 é utilizado para proteger recursos valiosos tal como o metrô para passageiros, Figura 13. Identifica-se que o metrô de passageiros é protegido por meio das catracas e para conseguir atravessar as catracas é preciso ir às máquinas de bilhetes para pegar os *tickets* ou *tokens*.

Figura 13 – Metrô como um recurso protegido.
Fonte: (SAKIMURA, 2017).

Na Figura 14, de maneira análoga, as máquinas de bilhetes são os servidores de autorização, as catracas protetoras de recursos são os servidores de recursos e os tickets são os *token*s de acesso na Figura 14. Um *token* de acesso pode ser válido por uma pessoa ou válido por um período de tempo.

Figura 14 – Analogia ao esquema OAuth 2.0.
Fonte: (SAKIMURA, 2017).

Portanto, como mostrado na Figura 13, Figura 14 e Figura 15, para acessar o metrô, é necessário solicitar um bilhete às máquinas de bilhetes, pegar o bilhete e apresentá-lo à catraca para ter acesso ao metrô como mostrado no fluxo da Figura 15. Na Figura 15, é estabelecida a relação entre os atores do OAuth 2. a qual identifica o vendedor de bilhetes como o servidor de autorização, os passageiros como os clientes, os donos do recurso como o proprietário da companhia do metrô, os *tokens* como *tickets* ou bilhetes e as catracas são como o servidor de recurso e o metrô é o próprio recurso.

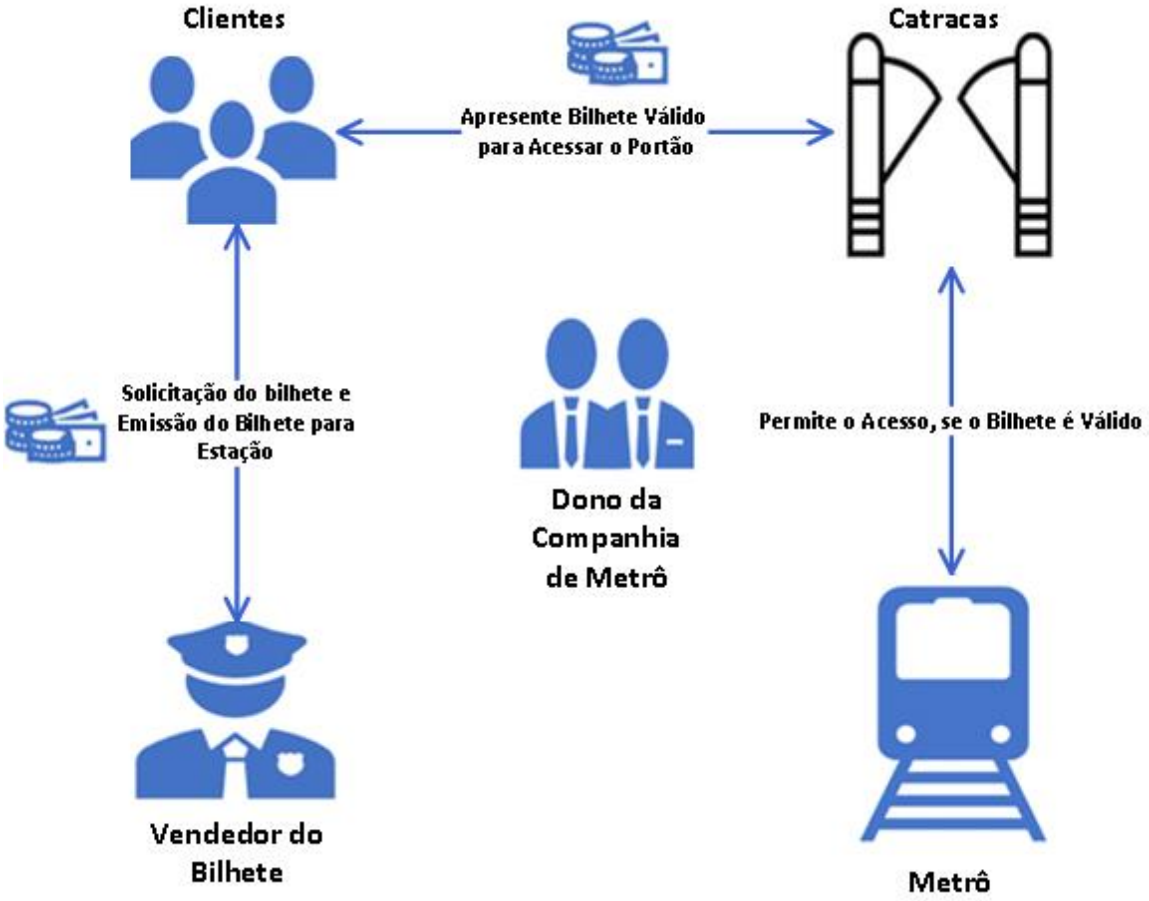

Figura 15 – Analogia do OAuth 2.0 com o metrô.
Fonte: (SHARMA, 2019).

Em suma, o acesso é gerenciado pelo servidor de autorização e no consentimento explícito do dono do recurso. Para conseguir acesso ao recurso, o cliente submete a requisição de acesso ao recurso para o servidor de autorização. O servidor de autorização recebe a requisição, após obter o consentimento do dono do recurso, gera uma concessão de acesso (ex.: bilhete) o qual pode ser usado pelo o cliente para provar ao servidor do recurso que o esse cliente tem privilégios em relação ao escopo e indica que credencial de acesso pode realizar alguma operação sobre o recuso como “recupera” ou “atualizar”, e a duração de utilização sobre recurso.

2.7.2 Atores

Os atores principais que participam do processo do OAuth 2.0 são o Cliente (*Client*, C), Servidor de Autorização (*Authorization Server*, AS), Servidor de Recurso (*Resource Server*, RS) e o Dono do Recurso (*Resource Owner*, RO). A Figura 16

mostra as classes de atores do OAuth 2.0.

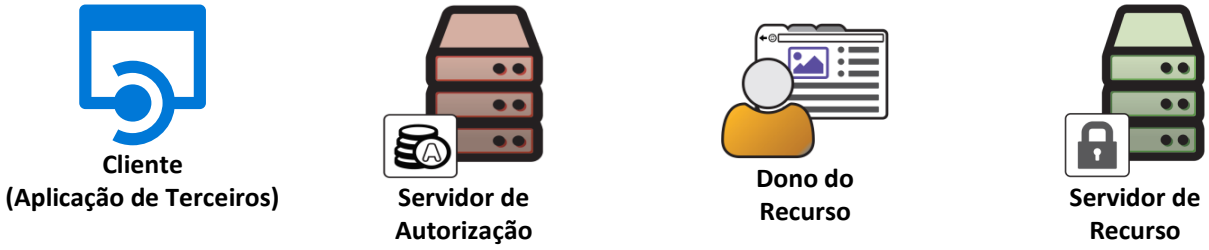

Figura 16 – Atores do OAuth 2.0.

Cliente. Consiste de um dispositivo ou aplicação que tem a intenção de consumir algum recurso. Um cliente faz requisições sobre uma API de serviços. Essas requisições consistem de ações sobre recursos protegidos se passando pelo do dono do recurso e que deve está autorizado pelo mesmo e autorização deve ser validada pelos servidores de autorização e de recurso (KUMAR, 2019). O cliente pode ser classificado de acordo com a capacidade de estabelecer conexão segura com servidor de autorização: **confidencial** quando são capazes de manter confidencialidade de suas credenciais; **públicos** quando são incapazes de manter confidencialidade.

Servidor de Autorização. É responsável pela autenticação e autorização do cliente.

Servidor de Recurso. Gerencia o acesso aos recursos disponibilizado, uma vez que é concedida a permissão pelo servidor de autorização.

Dono do Recurso. É o proprietário de um recurso e com direito a conceder acesso a ele.

2.7.3 Tokens

O token é uma credencial necessária para proteger os recursos. O *token* consiste de uma estrutura de dados que representa a autorização ou proibição, por exemplo, no uso de um bilhete ou uma licença, tem validade de uso. De qualquer modo, o *token* pode possuir vários formatos e vários métodos de utilização como, por exemplo, *token* de acesso (*Access Token*, AT)[14] (*Refresh Token*, RT)[15] e chaves

[14] *Access Token*: https://tools.ietf.org/html/rfc6819#section-3.2
[15] *Refresh Token*: https://tools.ietf.org/html/rfc6819#section-3.3

criptográficas como Prova de Posse (***P**roof-**o**f-**P**ossession*, PoP) dos *tokens*.

***Token* de Acesso**. Representa a credencial necessária para acessar os recursos. Tipicamente, o *token* de acesso é uma estrutura de dados emitida pelo servidor de autorização para o cliente e consumida pelo servidor de recursos. Além disso, o *token* de acesso representa uma autorização de duração curta para uso (em minutos ou horas).Além do mais, o *token* de acesso pode ser renovado através do uso de um *token* de *refresh* (LODDERSTEDT; MCGLOIN; HUNT, 2013).

Token* de *refresh. Representa uma autorização de longa duração porque é renovada para que o cliente possa acessar os recursos. O *token* de *refresh* é trocado apenas entre o cliente e o servidor de autorização e nunca são enviados ao servidor de recursos. Os clientes usam este tipo de *token* para obter novos *tokens* de acesso quando o *token* de acesso atual se tornar inválido ou expirar, ou para obter *tokens* de acesso adicionais com escopo idêntico ou aproximado usados para requisições no servidor de recursos (LODDERSTEDT; MCGLOIN; HUNT, 2013).

Prova de Posse (PoP) dos *Token*s. PoP vincula uma chave criptográfica a um token de acesso a fim do que o cliente possa ter sua liberação de acesso permitida. Para isso, o cliente apresenta o conteúdo criptográfico para o servidor de recurso. Por sua vez, o servidor de recurso verificar o conteúdo criptográfico. De posse da chave criptográfica, servidor de recurso descriptografa o conteúdo e depois valida o token. Se o *token* for válido, o cliente poderá ter acesso o recurso. Dessa maneira, a PoP faz com que se melhore a segurança no uso do *token* de acesso, pois reduz a chance de espionagem e interceptação da credencial de acesso do cliente durante a transmissão (BELTRAN; SKARMETA, 2017). As solicitações de PoP são geradas pelo servidor de autorização para o cliente demonstrar que tem posse de um segredo para o servidor de recursos o qual é conhecido exclusivamente entre o servidor de autorização e o servidor de recurso. A chave PoP pode usar criptografia simétrica ou assimétrica sobre o *token* de acesso.

2.7.4 Tipos de Fluxo

Na Figura 17 é demonstrada a interação realizada entre o servidor de autorização, o servidor de recurso, o cliente e o dono do recurso concebido por

(HARDT, 2012). As etapas do fluxo[16] que o protocolo do *framework* OAuth 2.0 realiza são apresentadas a seguir e como exemplo de um caso de uso do OAuth 2.0, as ilustrações se baseiam na situação em cliente quer acessar um serviço de armazenamento de arquivos em nuvem para realiza um backup dos arquivos:

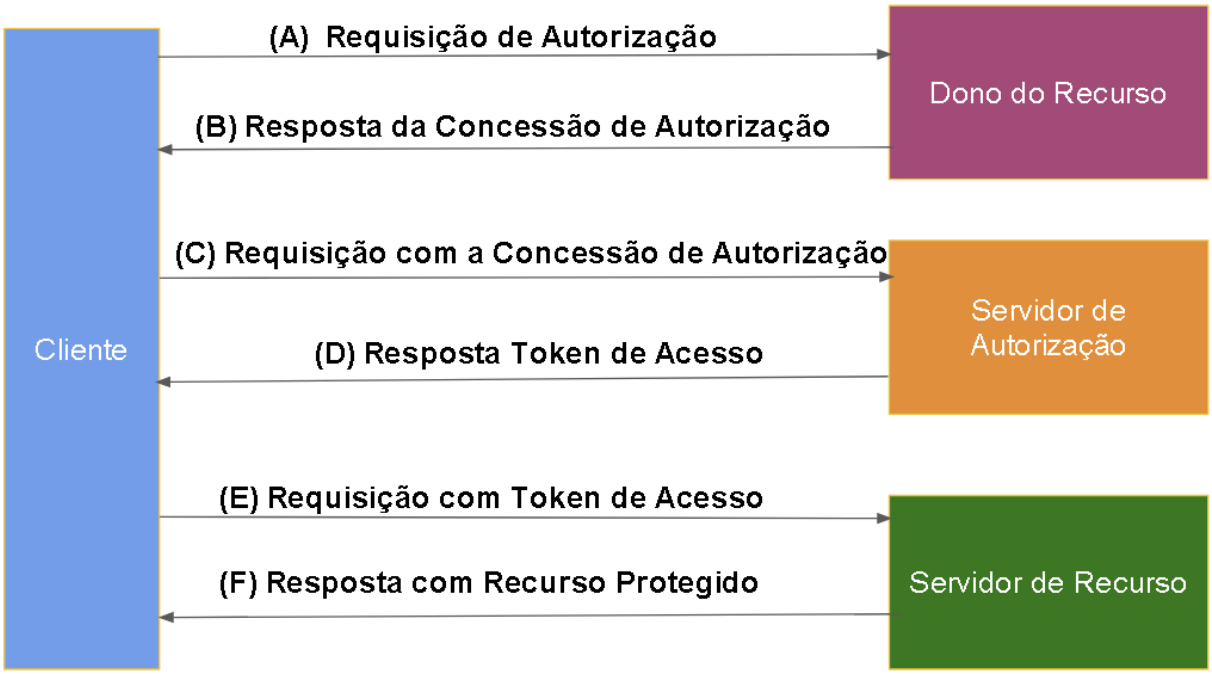

Figura 17 – Fluxo do protocolo OAuth 2.0.
Fonte: (HARDT, 2012).

Requisição de Autorização. A aplicação cliente solicita o privilégio de acesso para acessar os recursos do usuário (dono do recurso), Figura 18. O cliente solicita autorização do proprietário do recurso. A solicitação de autorização pode ser feita diretamente ao proprietário do recurso ou, de preferência, indiretamente por meio do servidor de autorização como intermediário.

[16] *Protocol Flow*: https://tools.ietf.org/html/rfc6749#section-1.2

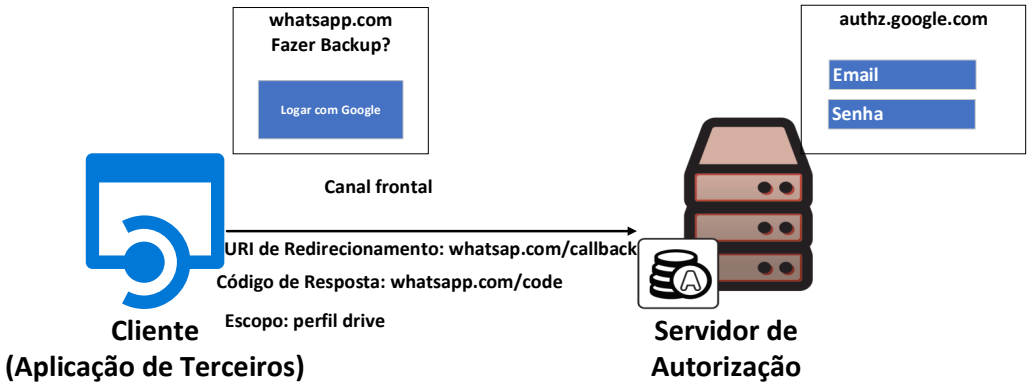

Figura 18 – Requisição de autorização.

Resposta da Concessão de Autorização. O dono do recurso escolhe em quais escopos concede privilégio de acesso sobre os recursos, Figura 19. O cliente recebe uma concessão de autorização, que é uma credencial representando a autorização do proprietário do recurso. O tipo de concessão de autorização depende do método utilizado pelo cliente para solicitar autorização e dos tipos suportados pelo servidor de autorização, Figura 20.

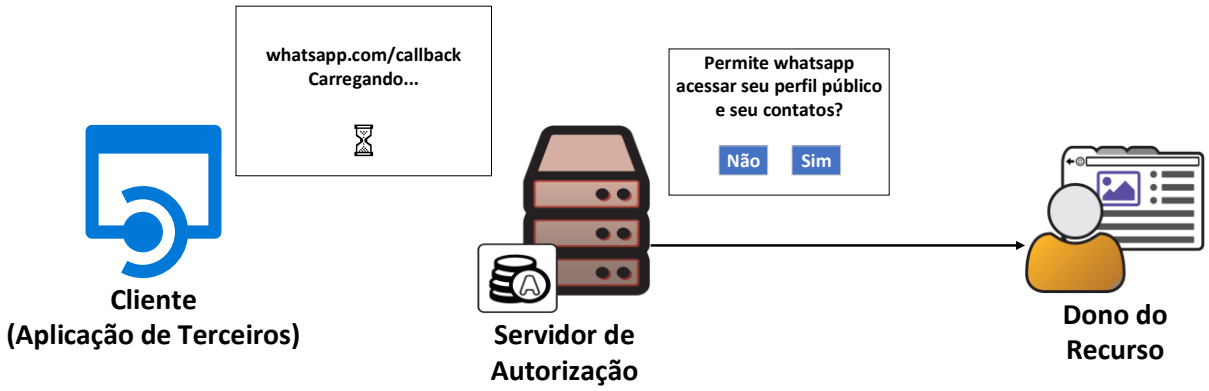

Figura 19 – Consentimento do dono do recurso.

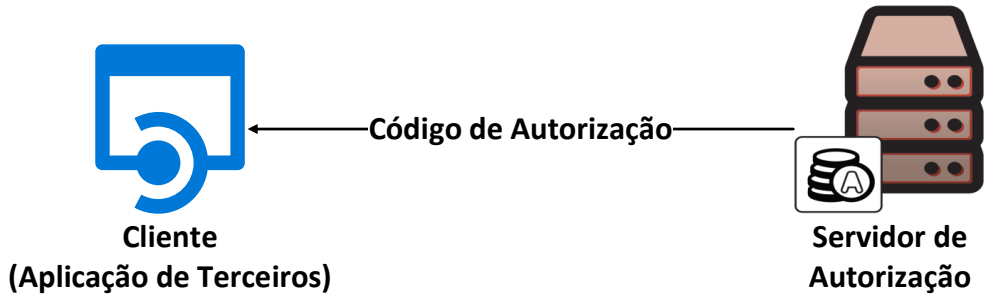

Figura 20 – Resposta da concessão de autorização.

Requisição com a Concessão de Autorização. O cliente sabendo de qual escopo tem privilégio de acesso ao recurso, pode requisitar um *token* de acesso válido para o servidor de autorização. O cliente solicita um *token* de acesso, autenticando-se com o servidor de autorização e apresentando a concessão de autorização.

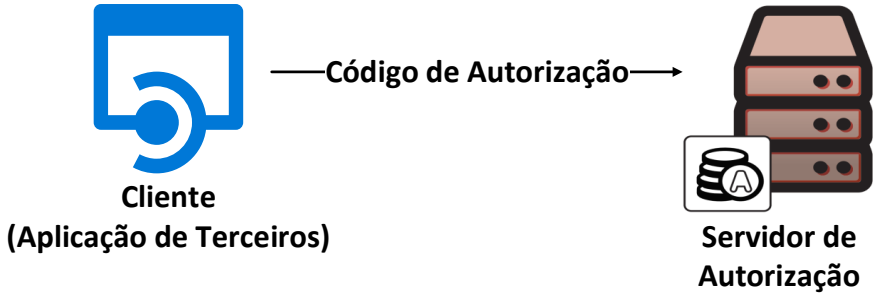

Figura 21 – Requisição com a concessão de autorização.

Resposta *Token* de Acesso. Se o servidor de autorização validar a requisição pelo cliente, o *token* de acesso será emitido. O servidor de autorização autentica a credencial apresentada pelo cliente e valida à concessão de autorização dada pelo dono do recurso. Se o consentimento for fornecido pelo dono do recurso, o servidor de autorização emite um *token* de acesso, Figura 22.

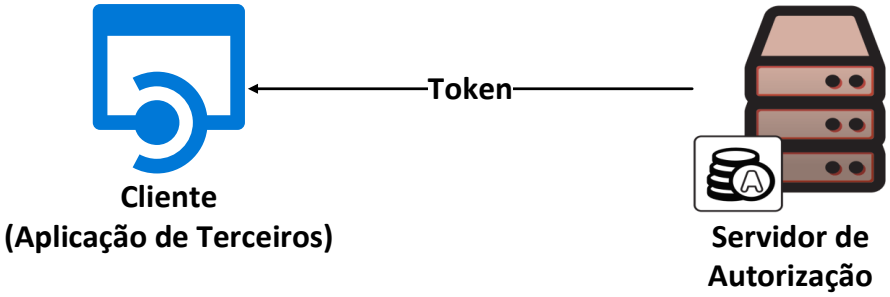

Figura 22 – Resposta do *token* de acesso.

Requisição com *Token* de Acesso. O cliente pode solicitar os recursos do dono sobre o servidor de recursos usando *token* de acesso para autenticação. Por suave, o cliente solicita o recurso protegido do recurso servidor e se autentica apresentando o *token* de acesso, Figura 23.

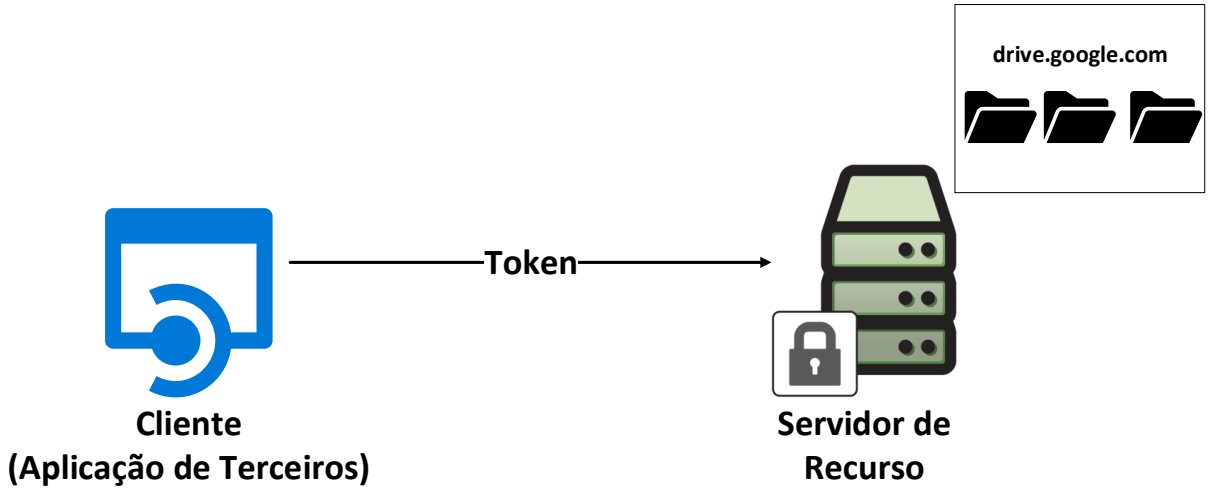

Figura 23 – Requisição com token de acesso.

Resposta com Recurso Protegido. Se o *token* de acesso for válido, o servidor de recursos fornece os recursos, Figura 24.

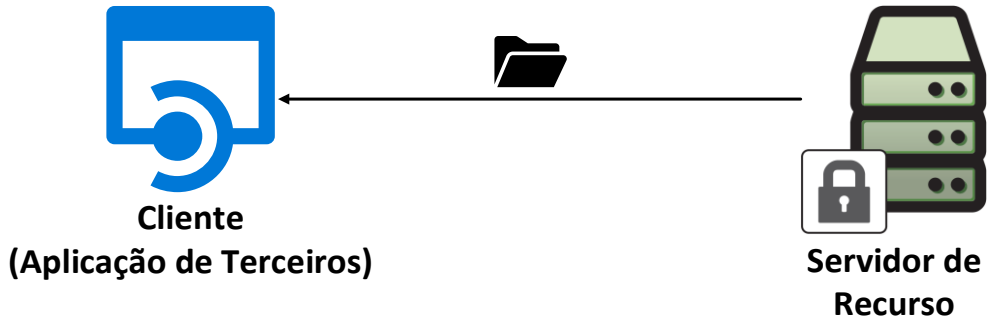

Figura 24 – Resposta com recurso protegido.

A especificação do *framework* OAuth 2.0[17] define quatro de tipos de fluxos (código de autorização, implícito, credenciais do cliente e credenciais do dono do recurso). Na Figura 25, é apresentado o fluxo genérico de concessão de autorização contém seguintes etapas:

[17] OAuth 2.0 *Framework*: https://tools.ietf.org/html/rfc6749

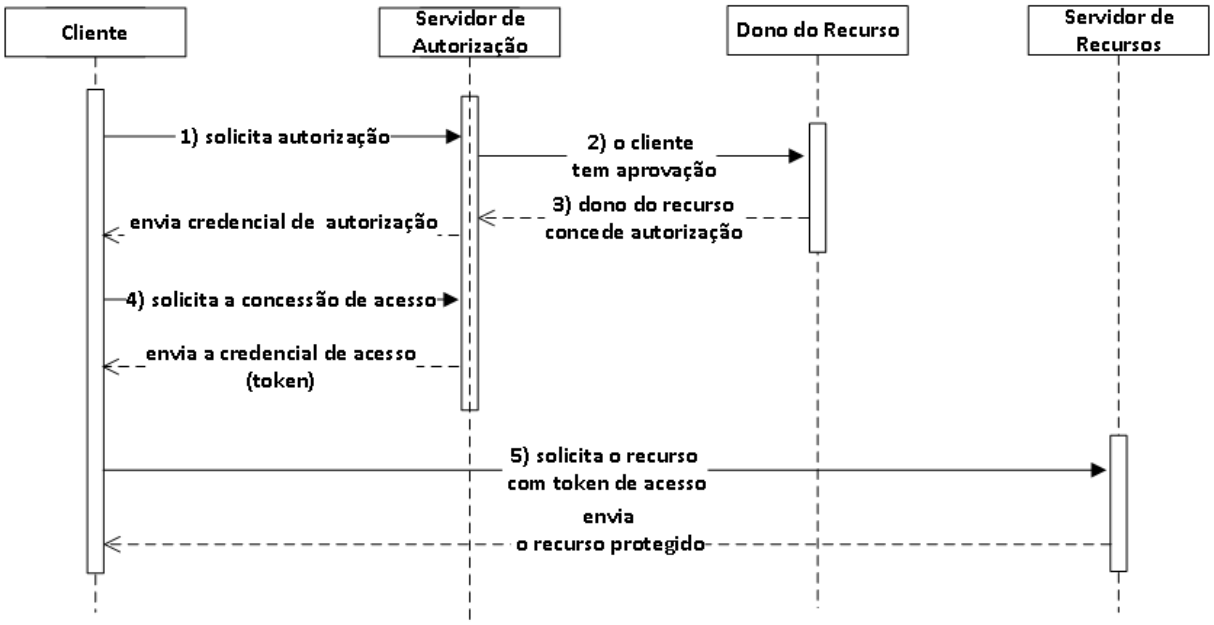

Figura 25 - Fluxo de concessão genérico de autorização do OAuth 2.0.

1) O cliente envia uma requisição para o servidor de autorização, solicitando uma credencial de autorização;
2) O servidor de autorização verifica o cliente. Se o cliente é legítimo, o servidor de autorização solicita ao dono do recurso o consentimento a para aprovar ou negar o pedido do cliente. Se o cliente tem aprovação para receber a credencial de autorização;
3) Se dono do recurso concede autorização, o servidor de autorização o emite para o cliente a credencial de autorização; Caso contrário, o servidor de autorização rejeita a solicitação do cliente e não emite a credencial de autorização para o cliente.
4) O cliente envia a requisição solicitando a credencial de acesso para o servidor de autorização, colocando na requisição a credencial de autorização. Se a credencial de autorização for válida, o servidor de autorização emite a credencial de autorização de acesso (*token*) para o cliente;
5) O cliente recebe o *token* e usa como credencial de acesso para solicitar os recursos para o servidor de recurso;
6) O servidor de recurso, responsável pelo controle de acesso aos recursos, recebe a requisição do cliente e verifica se a credencial de acesso repassada pelo cliente é legítima. Se a credencial de acesso for confirmada, o servidor de recurso envia o recurso protegido.

Tipicamente é imprescindível conhecer os tipos de aplicações clientes[18] definidas a fim de entender a razão do desenvolvimento de cada tipo de fluxo (HARDT, 2012). As aplicações podem ser caracterizadas como: aplicação *web*, aplicação baseada em agente do usuário (*user-agent*) e aplicação nativa.

Aplicação *web*. Refere-se a um cliente confidencial em execução em um servidor da web. Os proprietários de recursos acessam o cliente por meio de uma interface de usuário HTML renderizada em um agente do usuário no dispositivo usado pelo proprietário do recurso. As credenciais do cliente, bem como qualquer *token* de acesso emitido para o cliente, são armazenadas no servidor da Web e não precisam ser acessíveis ao proprietário do recurso.

Aplicação baseada no agente do usuário. Refere-se a um cliente público no qual o seu código é baixado de um servidor da *Web* e é executado dentro de um agente do usuário (ex.: navegador da *Web*) no dispositivo usado pelo proprietário do recurso. Os dados e as credenciais do protocolo são facilmente acessíveis (e frequentemente visíveis) ao proprietário do recurso.

Aplicação nativa. Refere-se um cliente público instalado e executado no dispositivo usado pelo proprietário do recurso. Os dados e as credenciais do protocolo estão acessíveis ao proprietário do recurso. Supõe-se que qualquer credencial de autenticação de cliente incluída na aplicação possa ser extraída. Por outro lado, credenciais emitidas dinamicamente, como *tokens* de acesso ou *tokens* de atualização, podem receber um nível aceitável de proteção. Essas credenciais são protegidas de servidores mal-intencionados com os quais o aplicativo pode interagir. Em algumas plataformas, essas credenciais podem estar protegidas de outras aplicações que residem no mesmo dispositivo.

De qualquer maneira, na investigação deste trabalho, destacam-se o fluxo de concessão código de autorização (*authorization code flow*)[19], fluxo de veracidade (*assert flow*) (CAMPBELL et al., 2015) e fluxo de concessão baseado na prova de chave para troca de código (*Proof Key for Code Change*, PKCE) (SAKIMURA;

[18] *The* OAuth 2.0 *Authorization Framework:* https://tools.ietf.org/html/rfc6749#page-15
[19] *Authorization Code Grant* :https://tools.ietf.org/html/rfc6749#section-4.1

BRADLEY; AGARWAL, 2015). Os demais fluxos encontrados na investigação se encontram no Apêndice D .

Fluxo da concessão de código de autorização É utilizado por clientes confidenciais. Este tipo de fluxo é comum em aplicações *backend, isto é sem interação com usuário*. Além de que, é um modo para realizar autenticação do cliente.

Nesse tipo a concessão o cliente deve ser capaz de interagir com o agente do usuário do dono do recurso e de receber requisições de entrada do servidor de autorização. A Figura 26 adaptada que representa o fluxo da concessão de código de autorização.

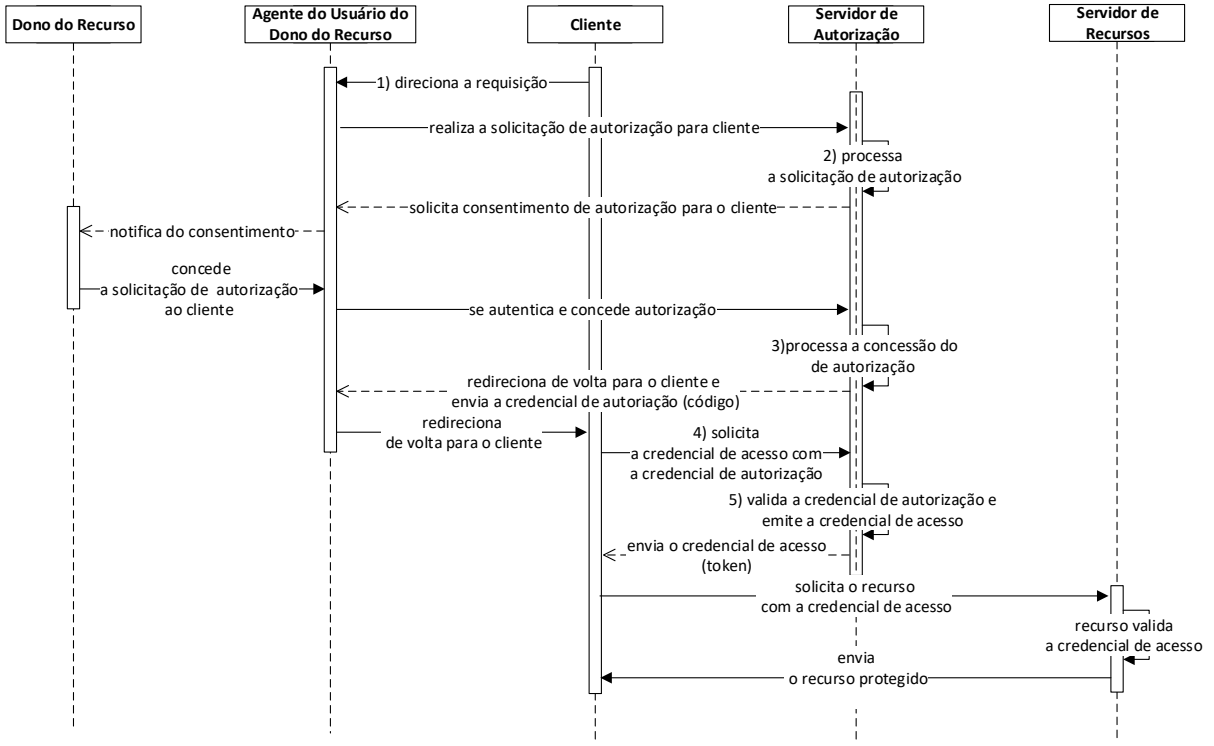

Figura 26 – Fluxo de concessão código de autorização.
Fonte: adaptado de (HARDT, 2012).

Na Figura 26 as etapas realizadas no fluxo de concessão código de autorização são:

1) O cliente inicia o fluxo pelo direcionamento para o agente do usuário do dono do recurso para o *endpoint* de autorização. O cliente inclui o identificador do cliente, escopo requisitado, estado local e um

redirecionamento URI[20], uma sequência compacta de caracteres que identifica um recurso abstrato ou físico, para o qual o servidor de autorização enviará de volta para o agente do usuário uma vez que o acesso seja concedido.

2) O servidor de autorização autentica o dono do recurso (via agente do usuário) e o dono do recurso concede autorização ao cliente.
3) Ao assumir o dono do recurso conceda autorização, o servidor da autorização redireciona o agente do usuário de volta para o cliente usando o URI de redirecionamento oferecido na requisição ou durante o registro do cliente, na etapa (2). A URI de redirecionamento inclui a credencial de autorização (código de autorização) e estado local oferecido pelo cliente recentemente.
4) O cliente solicita a credencial de acesso (*token* de acesso) para o *endpoint* do servidor de autorização, incluindo o código de autorização recebido na etapa (3) e o URI de redirecionamento.
5) O servidor de autorização autentica o cliente, valida a credencial de autorização e assegura que a URI de redirecionamento recebido é igual com a URI de redirecionamento usado pelo o cliente no passo (3). Se a comparação ocorrer com sucesso, o servidor de autorização emite a credencial de acesso (*token* de acesso) e, opcionalmente, emite credencial de renovação (*token* de *refresh)* e a(s) envia como resposta ao cliente.

Fluxo de concessão de veracidade. Pode ser aplicado na circunstância em que o cliente confidencial e o servidor de autorização possuem um vínculo de confiança com nível mas elevado, o que evita o consentimento de concessão por porta do dono do recurso para que o cliente tenha autorização para acessar ao o recurso. Nesse caso, a concessão de autorização é representada por meio um conjunto de atributos verdadeiros (*claims*) a fim de estabelecer uma relação de confiança com a parte confiante, emitido por uma autoridade confiável (RISCHER; SANSO, 2017).

Soluções de gerenciamento de identidade, no Apêndice G , como OpenID Connect[21], uma camada de autenticação sobre o OAuth 2.0, o qual emite e confirma

[20] *Uniform Resource Identifier* (URI): Generic Syntax https://tools.ietf.org/html/rfc3986
[21] Especificação do OpenID Connect: https://openid.net/connect/

a identidade do sujeito por meio do conjunto de atributos afirmativos, *claims*, que representa como, quando e onde a autenticação ocorreu.

Os *claims* formam a estrutura ID Token. No corpo da mensagem do ID Token está contido os atributos Autoridade Emissora (*Issuer Authorithy*), Público-Alvo (*Audience*), Data de Emissão (*Issue Date*) e Data de Expiração (*Expired Date*). Na Figura 27, há também um número de *claims* opcionais que podem ajudar parte confiável a validar o ID Token tempo de autenticação (*Authentication Time*) e *nonce*. Além disso, o ID Token pode também contar com atributos *claims* adicionais como nome e endereço de e-mail. Entretanto para desenvolvimento deste trabalho a formatação dos claims da estrutura ID Token foi seguindo padrão CWT.

Figura 27 – *Claims* do ID Token.
Fonte: (ORACLE, 2017).

- **Sujeito**: Atribuído para um usuário pelo provedor de Identidade;
- **Autoridade emissora**: Provedor de identidade que emite o *token*;
- **Público-Alvo**: Identifica a parte confiante a quem pode usar este o *token*;
- **Data de Emissão**: Data e tempo que o *token* foi emitido;
- **Data de Expiração**: Data e tempo que o *token* foi expirado;
- **Tempo de Autenticação**: Mostra o tempo que aplicação cliente foi autenticada;
- ***Nonce***: Valores que podem ser usados para mitigar ataques de replicação.

De todo modo, no fluxo de concessão de veracidade, a parte confiável, o servidor de autorização, emite um conteúdo no formato confiável ao cliente, a parte confiante. A Figura 28 representa o fluxo da concessão de veracidade.

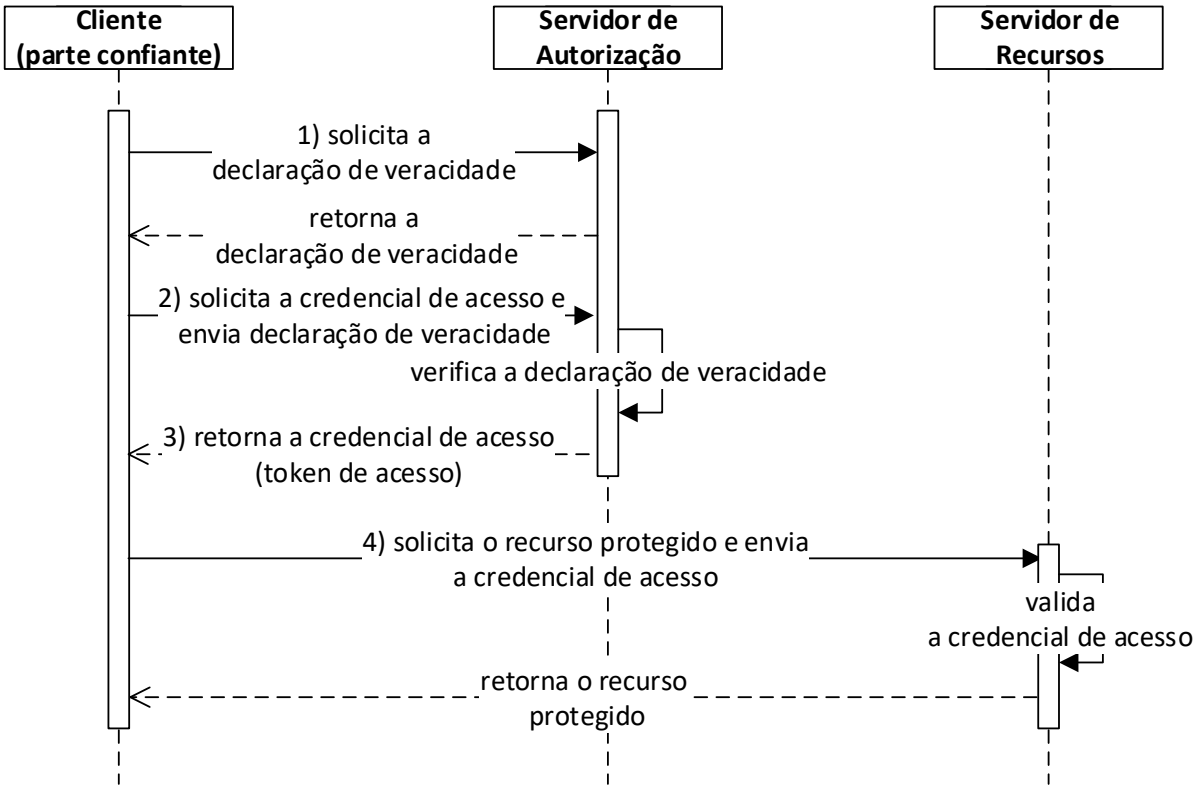

Figura 28 – Fluxo de concessão de veracidade.
Fonte: adaptado de (RICHER; SANSO, 2017).

Na Figura 28, as etapas realizadas para o tipo de concessão de veracidade são descritas a seguir:

1) O cliente solicita a declaração de veracidade para autoridade confiável, o servidor de autorização. Se o cliente é legítimo, a parte confiável retorna a declaração de veracidade.
2) Em posse da declaração de veracidade o cliente e solicita o *token* de acesso ao servidor de autorização. Por sua vez, o servidor de autorização verifica a declaração apresentado pelo cliente.
3) Se a declaração de veracidade é reconhecida pelo servidor de autorização, o servidor de autorização emite o *token* de acesso para o cliente.
4) O cliente com posse do *token* de acesso solicita o recurso protegido ao servidor de recursos. Se o *token* de acesso for válido, o servidor de recurso retorna o recurso protegido.

Fluxo de concessão de prova de chave para troca de código. É aplicado para os casos mitigação de ataques na etapa de solicitação de autorização no OAuth 2.0 (SAKIMURA; BRADLEY; AGARWAL, 2015). No PKCE, A espera um mensagem nova de B, primeiro A envia a B um desafio e exige que a mensagem subsequente resposta recebida de B contenha o valor do desafio correto (STALLINGS, 2014). Nessa troca de código, na solicitação de autorização, o cliente gera um desafio e o envia para o servidor de autorização. Esse desafio é armazenado pelo servidor de autorização até próxima solicitação, a requisição do *token*. Quando o cliente envia a requisição do *token* para o servidor de autorização, o cliente inclui o código verificador que gerou o desafio. Como o servidor de autorização recebeu e armazenou o desafio na requisição de autorização, o servidor autorização recupera desafio armazenado e gera o desafio baseado no código verificador recebido na requisição do *token*. Por sua vez, o servidor de autorização compara com o desafio recebido na requisição de autorização. Caso o desafio for gerado, na requisição do *token*, for igual ao desafio recebido, na requisição de autorização, o servidor de autorização aprova para a emissão do credencial de acesso (*token*). Senão a solicitação requisição de *token* é rejeitada. A Figura 29 representa o fluxo da concessão de PKCE.

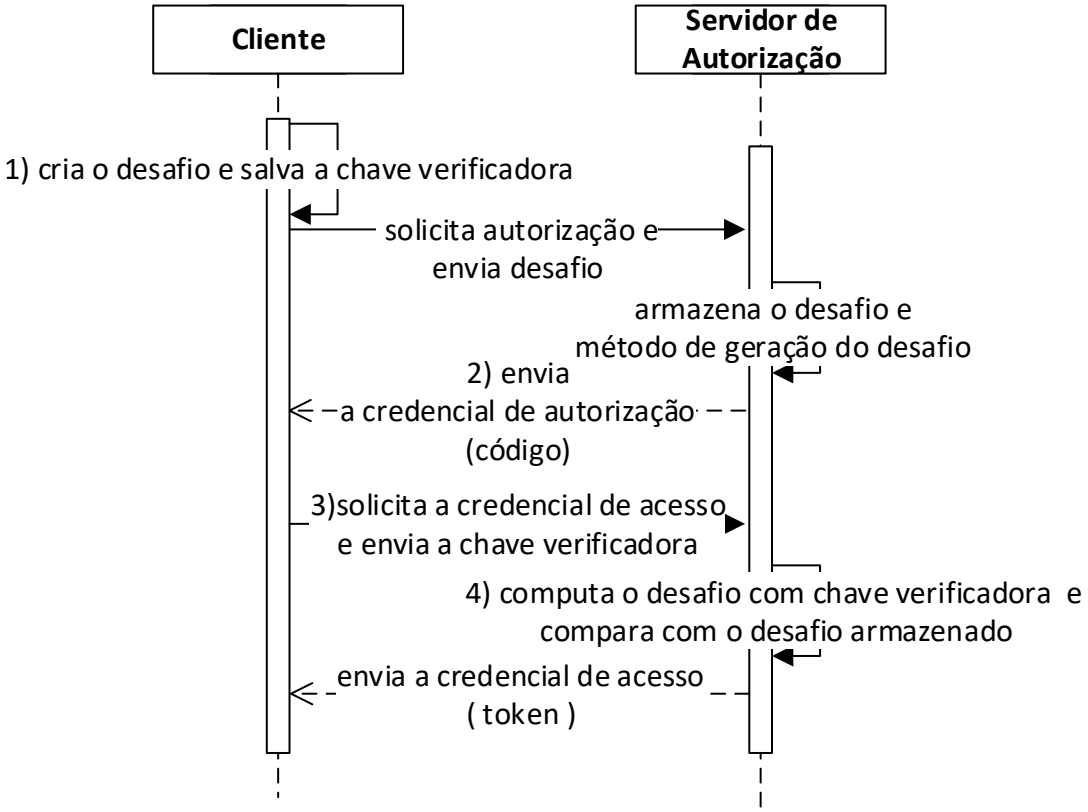

Figura 29 – Fluxo de concessão desafio-resposta.
Fonte: de (RICHER; SANSO, 2017).

As etapas realizadas para o fluxo de concessão de PKCE são:

1) O cliente cria e salva código verificador (“`code_verifier`”). Em seguida o cliente computa o código de desafio (“`code_challenge`”) baseado no (“`code_verifier`”) utilizando um método *hash* (”`code_challenge_method`”). Depois disso o cliente envia o código verificador (“`code_verifier`”) com os parâmetros para solicitação de autorização para o servidor de autorização.
2) O servidor de autorização envia a resposta para o cliente, mas armazena o código de desafio (“`code_challenge`“) e o método do código do desafio (“`code_challenge_method`“). Eles são associados com o código de autorização que é emitido na resposta de autorização pelo servidor de autorização.
3) Quando o cliente recebe o código de autorização, ele faz uma requisição para emissão da credencial de acesso (*token* de acesso) e inclui como parâmetro o código verificador (“`code_verifier`“).

4) O servidor gerar novamente o código de desafio (“`code_challenge`“), baseado no código verificador (“`code_verifier`“) enviado na requisição do cliente e depois checa se o valor computado é igual ao valor original do código de desafio, salvo anteriormente.

Diante dos fluxos apresentados e ao propósito da aplicação, pode-se ser definido qual ou quais tipos de fluxo de concessão pode ser utilizados no modelo de autorização, Figura 30. Para o desenvolvimento desta proposta utilizou-se o fluxo de concessão código de autorização em conjunto com o fluxo de concessão de asserção/veracidade e o fluxo de concessão desafio-resposta também conhecido como PKCE.

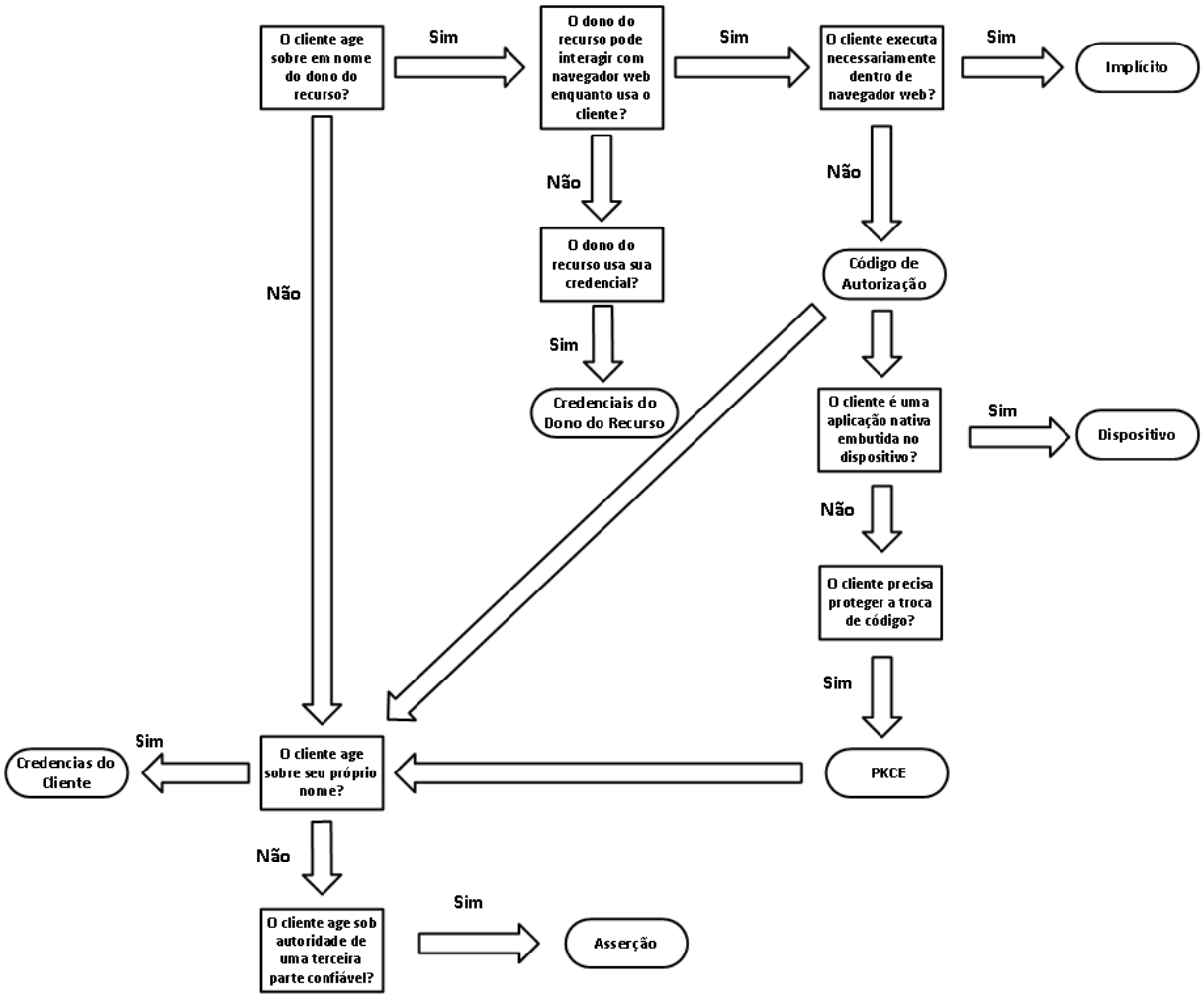

Figura 30 – Tipos de fluxo de concessão de autorização.

2.8 ACE-OAUTH

Em desenvolvimento pelo grupo de trabalho ACE[22] da IETF, o *framework* ACE-OAuth[23] é uma extensão da padronização de delegação de autorização do OAuth 2.0 para IoT (SEITZ et al., 2020). O ACE estabelece como o cliente deve utilizar prova de posse no controle de acesso pelo servidor de recurso em ambiente de IoT (BELTRAN; SKARMETA, 2017), Figura 31.

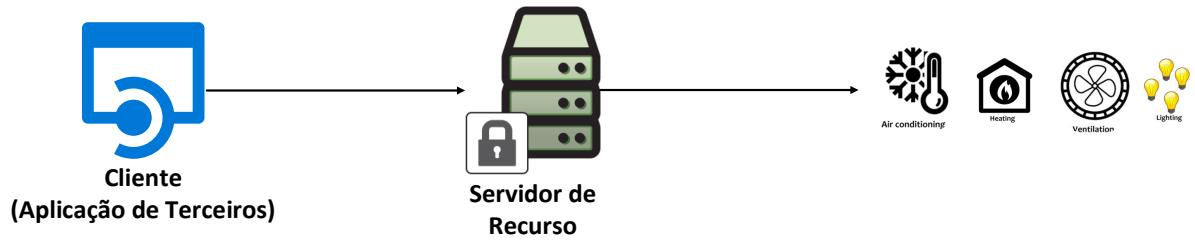

Figura 31 – Controle de acesso aos recursos em ambiente de automação residencial.

Contudo, para ter os benefícios do uso da PoP, ambos o cliente e o servidor de recurso têm que pré-estabelecer uma relação de confiança com servidor de autorização sobre a arquitetura OAuth 2.0. Para isso, ambos ,o cliente e o servidor de recursos, podem compartilhar uma chave secreta com produzida pelo servidor de autorização e reconhecida pelo servidor de recurso para garantir a comunicação segura, Figura 32.

[22] *Authentication and Authorization for Constrained Environments* (ACE): https://datatracker.ietf.org/wg/ace/charter/
[23] *Authentication and Authorization for Constrained Environments* (ACE) *using the* OAuth 2.0 Framework (ACE-OAuth): https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-30

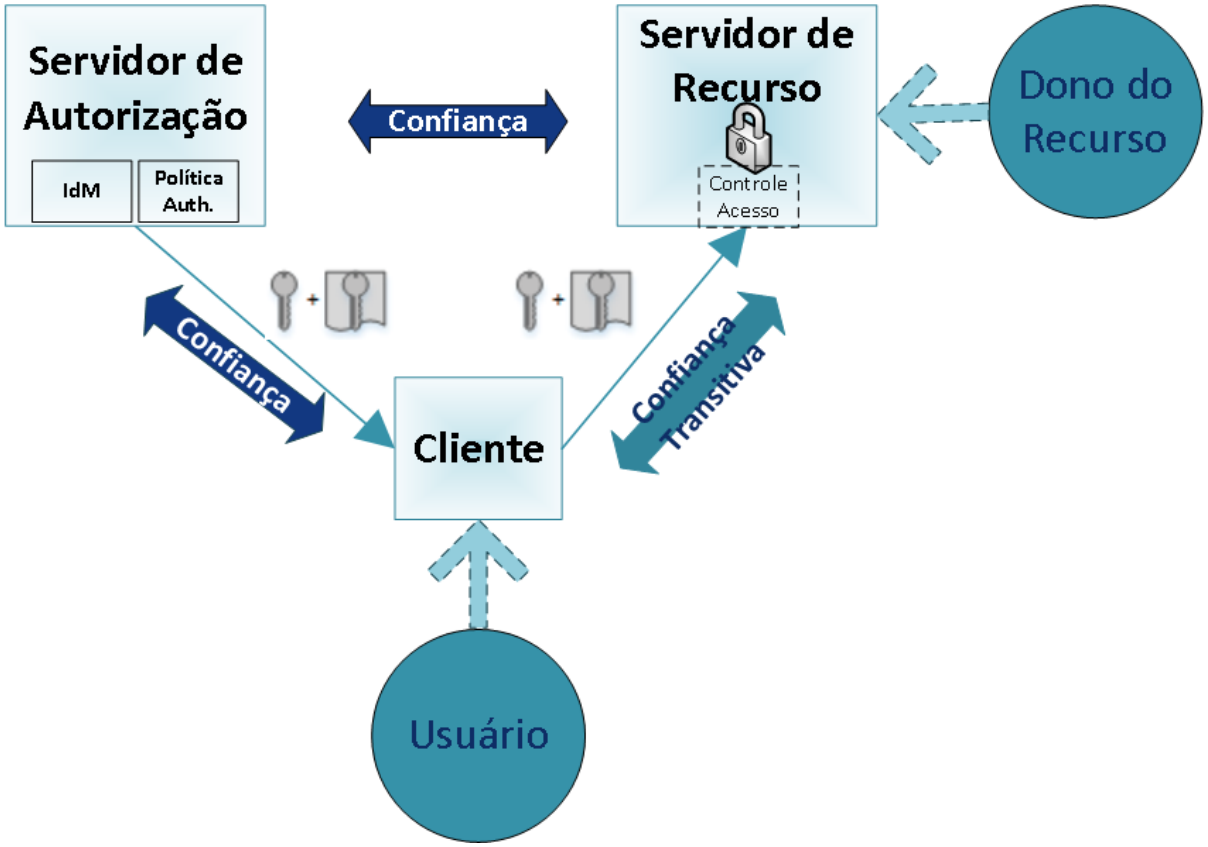

Figura 32 – Arquitetura para delegação de autorização em IoT.
Fonte: (BELTRAN; SKARMETA, 2017).

Dessa forma, na Figura 32, a aplicação PoP no *token* faz com melhore a segurança no uso da credencial de acesso (*token* de acesso), pois reduz a chance de interceptação da credencial de autorização do cliente durante a transmissão (BELTRAN; SKARMETA, 2017), e desse modo, evita o uso inadequado do portador do *token* (*token bearer)* [24] que concede autorização de acesso para o portador do *token* sem se preocupar com a autenticidade e a confidencialidade. Por conta do uso do *token* do portador, o *token* de acesso pode ser copiado ou sequestrado facilmente e, consequentemente, pode ser modificado ou adulterado, originando ataques de personificação e replicação.

2.8.1 Framework ACE-OAuth

O *framework* ACE-OAuth é constituído pelos seguintes aspectos, conforme Seitz et al. (2020):

[24] *The* OAuth 2.0 *Authorization Framework: Bearer Token Usage*: https://tools.ietf.org/html/rfc6750

Provisão de Credencial. Para IoT, não se assume que o cliente e o servidor de recurso são partes de uma infraestrutura de chave em comum, então o servidor de autorização é responsável por prover credenciais ou informações associadas para autenticação mútua entre o cliente e o servidor de recurso. As credenciais ou informações associadas que vincula o cliente e servidor de recurso são parte do conteúdo do *token* de acesso (*access token*).

Prova de Posse. O *framework* ACE-OAuth, já implementa prova de posse para *token*s de acesso por padrão. Dessa forma, o portador do *token* pode provar ser o portador da chave que está vinculado ao *token* (SEITZ et al., 2020). A vinculação é realizada por meio de um atributo *claim* ("`cnf`") que indica uma chave é usada para prova de posse.

Perfis do ACE. O cliente ou servidor de recurso podem ser limitados a suportar alguns tipos de encodificação ou alguns protocolos. No framework ACE-OAuth é esperado que o servidor de autorização gerencie a escolha do perfil que seja compatível para realizar o vínculo entre o cliente e um servidor de recursos (SEITZ et al., 2020).Para isso o servidor de autorização informa o cliente do perfil selecionado usando o parâmetro ("`ace_profile`") no *token* de resposta. Os tipos de perfil do ACE[25] podem ser assumido são CoAP/DTLS[26], OSCOAP [27], Publish-Subscribe[28] MQTT[29], IPsec[30], (*Joining OSCOAP multicast groups*)[31] e (*Security for Low-Latency Group Communication*)[32]. Na proposta deste trabalho foi selecionado o perfil CoAP/DTLS, pois o CoAP possui uma melhor custo para ambientes restritos, atua sobre o UDP, o que facilita nas trocas de mensagens e estabelecimento de canal de segurança que gera menor impacto, além disso é uma recomendação da própria especificação ACE-OAuth.

---

[25] ACE *Profile Roadmap*: https://trac.ietf.org/trac/ace
[26] *Datagram Transport Layer Security* (DTLS) *Profile for Authentication and Authorization for Constrained Environments* (ACE): https://tools.ietf.org/html/draft-ietf-ace-dtls-authorize-10
[27] OSCORE *profile of the Authentication and Authorization for Constrained Environments Framework* : https://tools.ietf.org/html/draft-seitz-ace-oscoap-profile-06
[28] CoAP Pub-Sub *Profile for Authentication and Authorization for Constrained Environments* (ACE)https://tools.ietf.org/html/draft-palombini-ace-coap-pubsub-profile-06
[29] MQTT-TLS *profile of* ACE: https://tools.ietf.org/html/draft-sengul-ace-mqtt-tls-profile-04
[30] IPsec *profile of* ACE: https://tools.ietf.org/html/draft-aragon-ace-ipsec-profile-01
[31] *Key Management for* OSCORE *Groups in* ACE: https://tools.ietf.org/html/draft-ietf-ace-key-groupcomm-oscore-06
[32] *Security for Low-Latency Group Communication*: https://tools.ietf.org/html/draft-tschofenig-ace-group-communication-security-00

Perfil CoAP/DTLS. O cliente pode usar a chave de criptografia simétrica ou assimétrica para obter segurança em nível de transporte com servidor de recursos. Para isso antes de estabelecer a comunicação entre o cliente e o servidor é realizado *handshake* do DTLS para acertar detalhes de configuração no estabelecimento do canal de segurança na comunicação.

2.8.2 Blocos

A especificação do framework ACE tem como arcabouço quatros blocos dentre eles, o *framework* OAuth 2.0, Protocolo de Aplicação Restrita (*Constrained Application Protocol,* CoAP)[33], Representação do Objeto Binário Conciso (*Concise Binary Object Representation*, CBOR)[34], Criptografia e Assinatura do Objeto CBOR e (CBOR *Object Signing and Encryption*, COSE)[35], Figura 33:

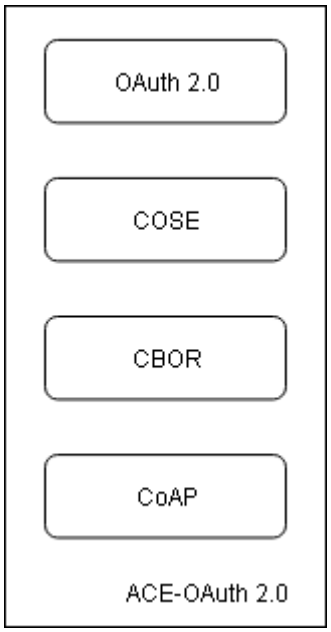

Figura 33 – Componentes recomendados pelo *framework* ACE-OAuth.

CoAP. É um protocolo leve para transferência *Web* e que atua na camada de aplicação, especificamente projetado para ambientes com limitações de recursos. Esse protocolo tipicamente roda sobre UDP o que reduz o *overhead* nas trocas de

[33] *The Constrained Application Protocol* (CoAP): https://tools.ietf.org/html/rfc7252
[34] *Concise Binary Object Representation* (CBOR): https://tools.ietf.org/html/rfc7049
[35] CBOR Object Signing and Encryption (COSE): https://tools.ietf.org/html/rfc8152

mensagens (SHELBY; HARTKE; BORMANN, 2014)

O CoAP é um protocolo que atua na camada de aplicação com finalidade de evitar sobrecarga na rede e economia na utilização dos recursos. O CoAP é uma boa solução para dispositivos que possuem limitação de recursos (memória, consumo de energia, processamento).

O CoAP tem um menor impacto no processamento, na memória, na latência e na comunicação dos dispositivos, já que atua sobre o UDP. Nesse como se sabe, o UDP não se preocupa em estabelecer uma orientação à conexão, realizar o controle de fluxo, estabelecer uma transmissão confiável e em entregar pacotes ordenados, como ocorre com TCP. O protocolo CoAP possui as características de camada de mensagem, integração e compatibilidade, modo de transmissão, função observador e segurança (WESTPHALL, 2018).

Camada de mensagem. Adiciona uma camada de mensagem para detecção e retransmissão dos pacotes a fim de tratar o problema de perda de pacotes, não solucionado pelo UDP, Figura 34.

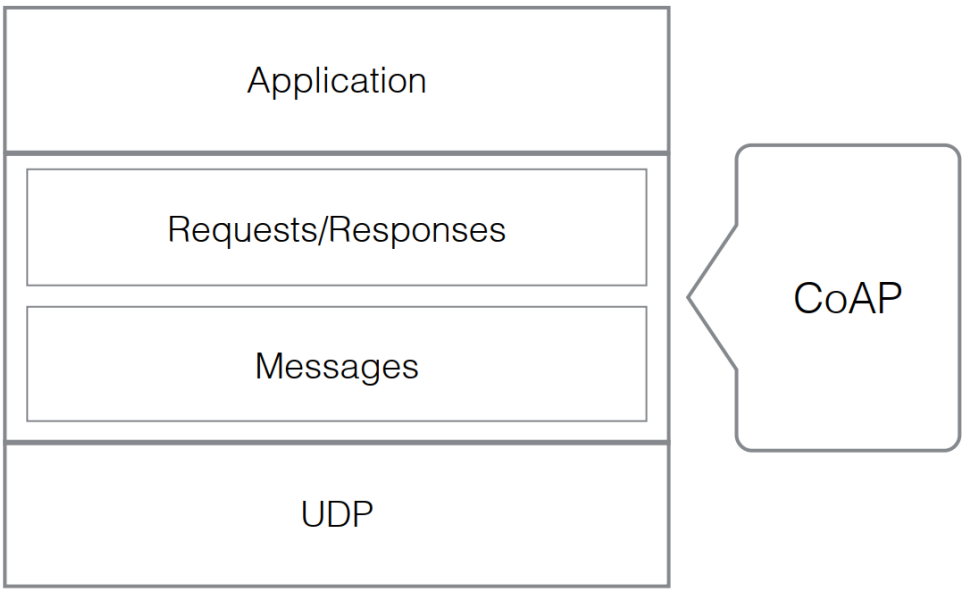

Figura 34 – Camadas do protocolo CoAP.
Fonte: (PIRETTI, 2013).

Integração e compatibilidade. Usa o mesmo conjunto de operações básicas (ex. GET, POST, PUT, DELETE) contidas no HTTP. Possibilita a conversão de mensagens HTTP para CoAP ou vice-versa, quando se passa por algum *proxy*. E

também, pode usar modelo de arquitetura Transferência de Estado Representacional (*Representational State Transfer,* REST)[36], Figura 35.

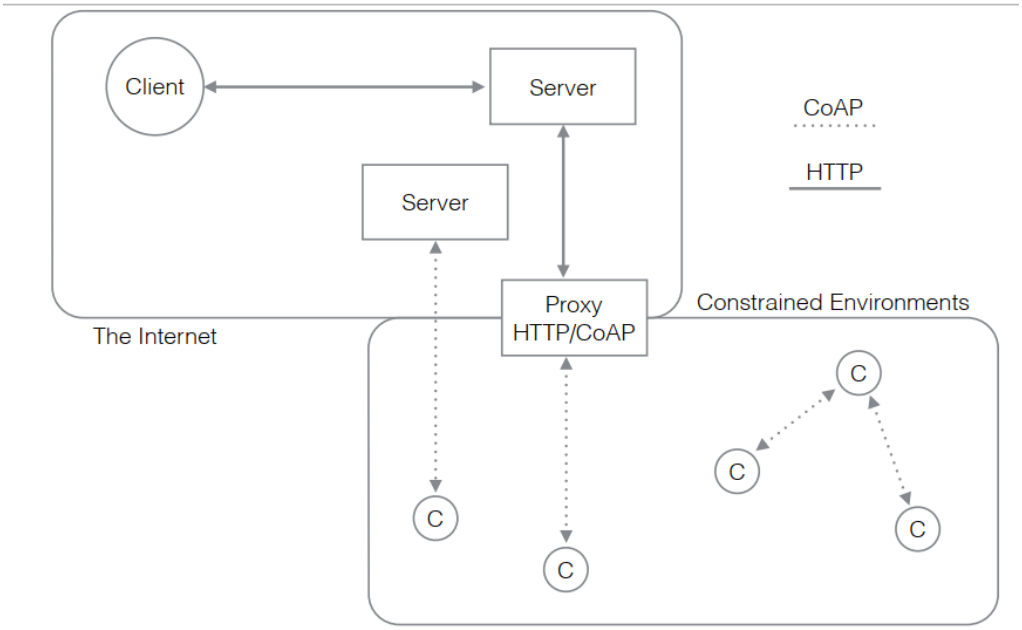

Figura 35 – Arquitetura REST CoRE[37].

Fonte:(PIRETTI, 2013).

Campos no cabeçalho. O CoAP possui os seguintes campos no cabeçalho: *token*, tamanho do *token*, versão, tipo de mensagem, código da operação, id da mensagem, opções e carga útil, Figura 36.

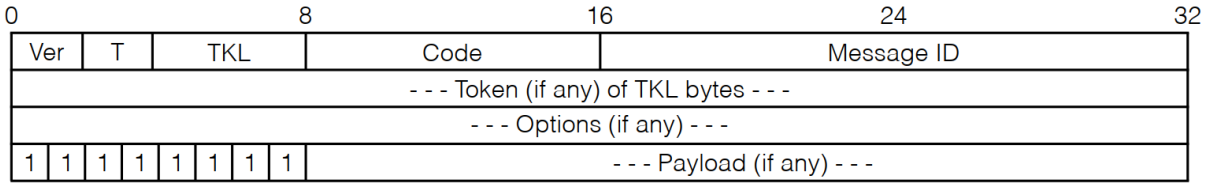

Figura 36 – Formato da mensagem CoAP.

Fonte: (PIRETTI, 2013).

Modo de transmissão. A sua finalidade é de não perder as solicitações, onde se verificado se é o último bloco foi recebido.

Função de observador. Contribui para não haver perdas de requisições e nem

[36] *Architectural Styles and the Design of Network-based Software Architectures:* https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[37] Constrained RESTful Environments (CoRE) Link Format: https://tools.ietf.org/html/rfc6690

sobrecarga dos dispositivos.

Segurança. Uso de DTLS para obter segurança nas trocas de mensagens.

De qualquer maneira, as mensagens CoAP são trocadas de forma assíncrona entre os *endpoint*s. Além disso, as mensagens CoAP podem chegar fora de ordem, podem aparecer duplicadas ou podem aparecer sem aviso prévio. Além do mais o CoAP define tipos de mensagens para camada de mensagem, que são:

- **Confirmável** (***Con****firmable*, CON): Indica que os dados carregados tem que ser reconhecidos a partir do receptor, oferecendo funcionalidade confiável.
- **Não-confirmável** (***Non****-Confirmable*, NON): Carregam dados que não requerem reconhecimento.
- **Reconhecimento** (***Ack****noledgement*, ACK): Reconhece mensagens de solicitação CON.
- **Reiniciar** (***R****eset*, RST): Erros de sinais que ocorrem na recepção da mensagem CON.

No entanto, vale mencionar também que as mensagens podem ser transmitidas de forma confiável e não confiável.

Mensagens transmitidas de forma confiável. A transmissão confiável de uma mensagem é iniciada marcando a mensagem como confirmável no cabeçalho CoAP. Uma mensagem de reconhecimento é enviada ao se dar uma resposta à solicitação recebida ou a mensagem de reconhecimento é enviada quando **a mensagem for rejeitada** enviando uma mensagem de *reset* como resposta e ignorando a mensagem recebida. De todo modo, uma resposta é conhecida como *piggy-backed* quando pega carona mensagem ACK já pronta, ao invés de enviar uma mensagem para ser estruturada, por conta disso a mensagem é disponibilizada rapidamente e. Se a requisição precisa de mais tempo para ser processada e para prevenir retransmissões do cliente deve responder com uma mensagem de reconhecimento, Figura 37.

Mensagens transmitidas de forma não-confiável. Uma mensagem não confirmável sempre traz uma resposta e não deve estar vazia.

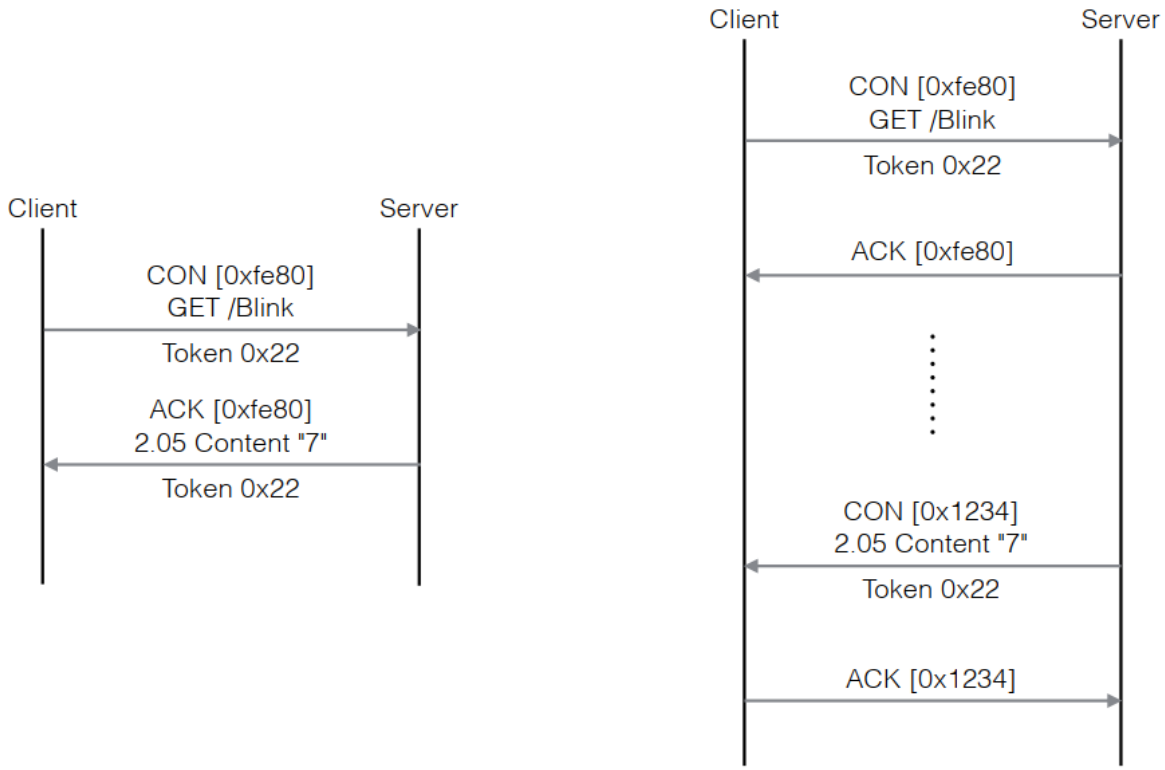

Figura 37 – Resposta *piggy-back* e separado em CoAP.

Fonte: (PIRETTI, 2013).

DTLS. Para obter uma configuração parecida com o HTTP usando o Segurança na Camada de Transporte (*Transporte Layer Security*, TLS), o CoAP pode ser combinado com a Camada de Transporte Segura baseado em Datagrama (*Datagram Transporte Layer Security*, DTLS) na troca de mensagens. A estruturação do datagrama utilizado no DTLS tem algumas similaridades que são herdados do próprio TLS (RESCORLA; MODADUGU, 2012), Figura 38. Assim como o CoAP, O DTLS tem um menor impacto no processamento e na memória nos dispositivos e na latência na comunicação, já que atua sobre o UDP.

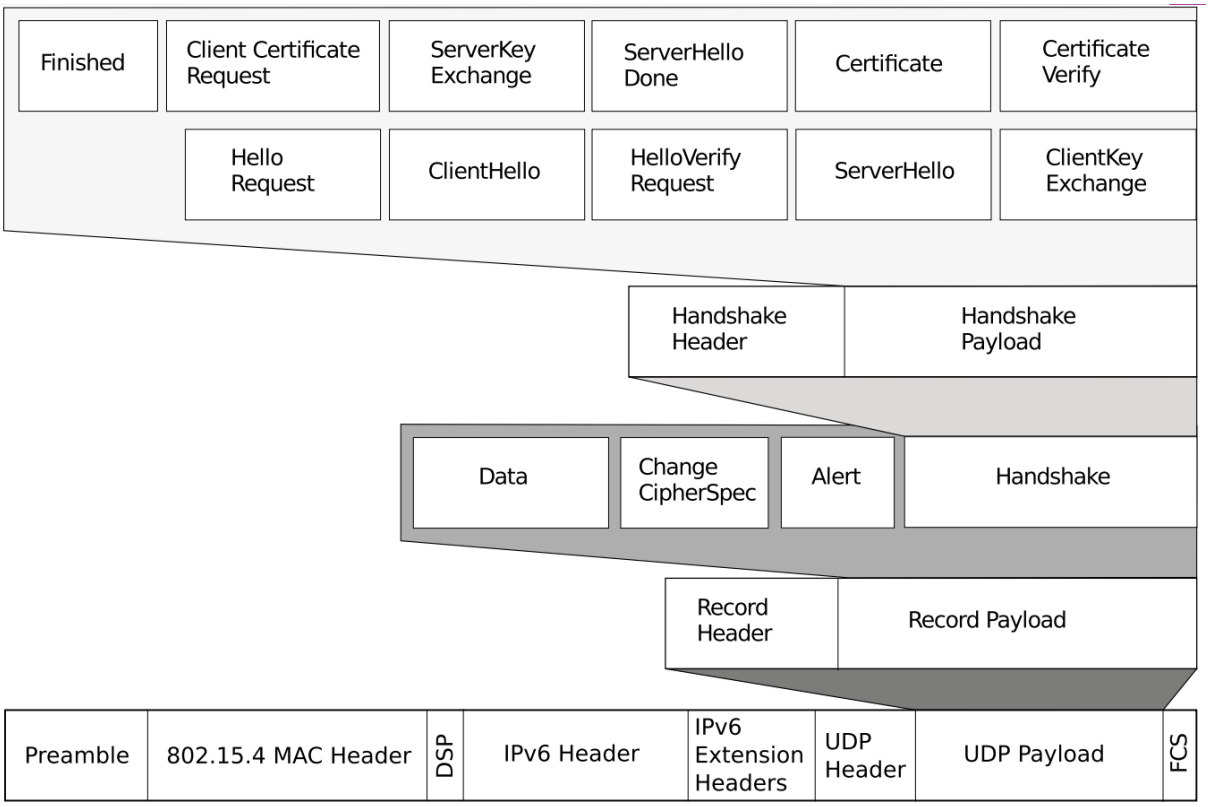

Figura 38 – Estrutura do datagrama seguro com DTLS.
Fonte: (RAZA et al., 2013)

CBOR. É uma especificação de estrutura de dados compacta para ser usada na serialização e deserialização das mensagens no processo de comunicação entre duas partes. A ideia é criar um decodificador e codificador em ambas às extremidades da comunicação onde os dados trocados sejam serializados.

Segundo Bormann & Hoffma (2013), o CBOR é uma encodificação binária projetada para redução de código e tamanho da mensagem. Além disso, o CBOR pode ser usado para codificar *tokens* autocontidos e também para codificar a carga útil em protocolos de mensagens.

O objetivo do CBOR é reduzir o tamanho da mensagem e reduz o poder de processamento. Essa redução no consumo de recurso é um bom sinal para ambiente com recurso restrito em IoT como mostra Código 1. Além disso, substitui o uso da notação do objeto Javascript (*JavaScript Object Notation*, JSON)[38], um formato de dados que é popular nas aplicações nativas e *Web*, mas é considerado ineficiente para alguns sistemas de IoT que usa tecnologia de rádio transmissor de

[38] *The JavaScript Object Notation* (JSON) *Data Interchange Format*: https://tools.ietf.org/html/rfc8259

baixa potência.

```
1 a4                                    # map(4)
2       01                              # unsigned(1) (=AS)
3       78 1c                           # text(28)
4          636f6170733a2f2f61732e657861
5          6d706c652e636f6d2f746f6b656e  # "coaps://as.example.com/token"
6       05                              # unsigned(5) (=audience)
7       76                              # text(22)
8          636f6170733a2f2f72732e657861
9          6d706c652e636f6d              # "coaps://rs.example.com"
10      09                              # unsigned(9) (=scope)
11      66                              # text(6)
12         7254656d7043                 # "rTempC"
13      18 27                           # unsigned(39) (=cnonce)
14      45                              # bytes(5)
15         e0a156bb3f                   #
```

Código 1 – Comparação entre CBOR e JSON.

No Código 1, o CBOR pode usar outros tipos de dados como inteiro para representar a chave do par chave/valor, diferentemente do JSON que especifica no formato dos dados por meio do tipo de dados conjunto de caracteres (*string*) que impacta na codificação da mensagem.

COSE. A necessidade para adquirir habilidades sobre serviços de segurança básica em aplicações *Web* e móveis originou a elaboração de um conjunto de especificação, que formam a família JOSE desenvolvidas pelo grupo de trabalho da IETF, que especifica como processar a criptografia, assinaturas, operações Código de Autenticação de Mensagem (*Message Authentication Code*, MAC) e como codificar as chaves usando o formato de dados JSON.

No entanto, houve um aumento intensivo na busca de ter serviços de segurança que se moldassem ao ambiente restrito de IoT. Com isso, se originou o COSE como uma padronização sobre o formato de estrutura de dados CBOR em substituição ao JOSE.

Especificamente, o formato da mensagem em JOSE contém o tipo objeto mapa que tem como tipo de dados cadeia de caracteres (*string*) na sua chave. Em COSE, os tipos de dados na chave do mapa que podem ser usados são: cadeia de caracteres, inteiros negativos e inteiros sem sinais.

Mas diferentemente do JOSE, a COSE descreve como criar e processar assinaturas, códigos de autenticação de mensagens e criptografia usando serialização do CBOR a fim de atender as restrições do ambiente restrito de IoT.

Nas mensagens em COSE, o tipo da mensagem é identificado por um rótulo

CBOR. Desse modo, as mensagens rotuladas são identificadas como rótulos CBOR, enquanto as demais sem um rótulo CBOR são conhecidas como mensagens sem rótulos. Nesta ocasião, as mensagens rotuladas podem ser mensagens criptografadas, assinadas ou autenticadas com MAC. O tipo de rótulos nas mensagens COSE podem ser Encript0, Encript0, MAC0, MAC, Sign0 e Sign.

- **Encript0**: Uma mensagem COSE criptografada que tem apenas um destino.
- **Encrypt**: Uma mensagem COSE criptografada que pode ter múltiplos destinos (a mensagem pode criptografada sobre chaves diferentes para diferentes destinatários).
- **MAC0**: Uma mensagem COSE de autenticação que tem um destino.
- **MAC**: Uma mensagem COSE autenticada que pode ter múltiplos destinos (a mensagem pode ser assinada com chaves diferentes para diferentes destinatários).
- **Sign0**: Uma mensagem COSE assinada com um assinante.
- **Sign**: Uma mensagem COSE que foi assinada por múltiplos assinantes.

Vale destacar que para codificar os objetos, o COSE oferece codificação binária que codifica uma estrutura de dados com conteúdo criptografado. Esses objetos são usados no fluxo do protocolo ACE. Com isso, a mensagem é solicitada para ser enviada criptografada a partir de uma entidade para outra. No desenvolvimento da solução da proposta se utilizou o tipo de rótulo nas mensagens COSE Encrypt0.

Estrutura da mensagem COSE. As mensagens COSE são também construídas usando o conceito de camada para separar diferentes tipos de conceitos criptográficos. Por exemplo, considere a mensagem rotulada com o tipo COSE_Encrypt. Este tipo do rótulo da mensagem é quebrado em duas camadas: a camada de conteúdo e a camada de destinatário. Na camada de conteúdo, o texto claro é criptografado e a informação sobre a mensagem é atribuída. Na camada do destinatário, a chave de criptografia do conteúdo é criptografada e a informação sobre como isso é criptografado para cada destinatário é atribuído.

As estruturas das mensagens do COSE são construídas sobre um tipo de dados, o mapa (*map)* CBOR. A estrutura das mensagens COSE consistem em três

partes principais cabeçalho protegido (*protected header*), Cabeçalho desprotegido (*desprotected header)*, carga útil (*payload*)/texto cifrado (*ciphertext*) e duas opcionais assinatura MAC e assinantes destinatários (*recipients signers*): como mostra Figura 39.

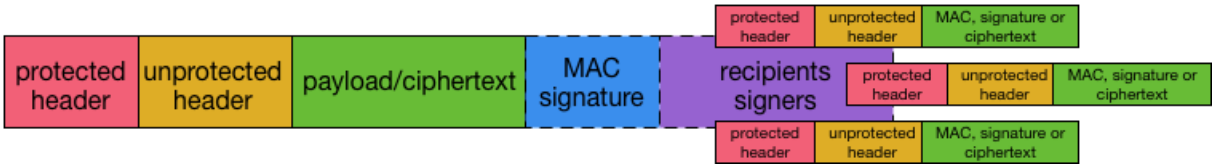

Figura 39 – Estrutura da mensagem COSE.
Fonte:(CLAEYS; ROUSSEAU; TOURANCHEAU, 2017).

Cabeçalho protegido. Contém informação, que não precisa ser serializada, mas que é necessária ser protegida sobre a configuração dos algoritmos criptográficos e que devem ser confidenciais e privadas.

Cabeçalho desprotegido. Contém a informação, que não precisa ser serializada, sobre a configuração dos algoritmos criptográficos que pode ser visível e compartilhada.

Texto cifrado. Contém a mensagem criptografada protegida e a configuração dos algoritmos criptográficos apresentados nos cabeçalhos.

Assinatura MAC[39]. É um mecanismo de checagem de integridade baseada em chave secreta. Tipicamente, os códigos de autenticação de mensagem (*message authentication codes*, MAC) são usados entre duas partes que compartilham uma chave secreta a fim de validar a informação valida transmitida entre essas partes.

Assinantes destinatários. Contem um vetor de estruturas de informações do destinatário.

Por exemplo, na parte principal da mensagem COSE, a estrutura COSE Encrypt0, usada neste trabalho, é um vetor contendo três elementos como ilustrado no Código 2.

[39] HMAC: *Keyed-Hashing for Message Authentication*: https://www.rfc-editor.org/rfc/rfc2104.html

```
1  [
2  protected_header, // Mapa encodificado como caracteres em byte
3  unprotected_header, // Mapa CBOR
4  ciphertext // Caracteres em byte
5  ]
```

Código 2 – Corpo do COSE_Encrypt0

CWT. O *Token* Web CBOR (**C**BOR ***W**eb **Token***, CWT)[40] é um conjunto de atributos declarados verdadeiramente, conhecido como (*claims*), que identifica as partes que estabelecem uma comunicação confiável sobre determinado assunto.

Os *claims* do CWT podem ser codificados usando CBOR. O CWT é derivado do (JSON *Web Token* , JWT)41, mas substitui o JSON por CBOR e COSE no lugar do (JSON *Object Signing and Encryption*, JOSE)42. O conjunto de evidências ou afirmações verdadeiras (*claims*) que compõe a estrutura CWT:

- **Emissor** (***Iss**uer*, iss): Identifica a parte principal que representa o CWT. O processamento desta *claim* é específico da aplicação.
- **Sujeito** (***Sub**ject*, sub): Identifica a parte principal que é o sujeito ou referencia um assunto do CWT. O assunto referenciado ou sujeito deve estar no escopo entre único localmente no contexto do emissor ou único globalmente.
- **Público-alvo** (**Aud**ience, aud): Identifica os destinatários para os quais o CWT é encaminhado. Cada parte principal é destinada a processar o CWT e ele próprio deve se identificar com o valor desta *claim*. Se não é a parte principal então o CWT deve ser rejeitado.
- **Data de expiração** (***Exp**ired Date*, exp): Identifica o tempo de expiração no qual ou após o qual o CWT não deve ser aceito para processamento.
- **Não antes** (***N**ot **B**e**f**ore*, nbf): Identifica o tempo antes do qual o CWT não deve ser aceito para processamento.
- **Data de emissão** (***I**ssue D**at**e*, iat): Identifica o horário em que o CWT foi emitido.
- **Identificador CWT** (**C**WT **ID**, cti). Fornece um identificador único para o CWT. O valor do identificador deve ser atribuído de maneira a garantir que

[40] CBOR *Web Token*: CWT: https://tools.ietf.org/html/rfc8392
[41] JSON *Web Token* (JWT): https://tools.ietf.org/html/rfc7519
[42] *Examples of Protecting Content Using JSON Object Signing and Encryption* (JOSE): https://tools.ietf.org/html/rfc7520

haja uma probabilidade insignificante de que o mesmo valor seja acidentalmente atribuído para outro objeto contendo dados diferentes; se o aplicativo usar vários emissores, colisões devem ser evitadas entre o valor do CWT ID produzido por diferentes emissores.

Na Tabela 2 é mostrado o sumário[43] dos nomes dos atributos (*claims*), chaves e o tipo de dados de cada valor

Tabela 2 – Sumário dos nomes dos atributos (*claims*), chaves e o tipo de dados de cada valor.

Parâmetro	Chave	Valor do tipo
iss	1	caracteres de texto
sub	2	caracteres de texto
aud	3	caracteres de texto
exp	4	número de ponto flutuante ou inteiro
nbf	5	número de ponto flutuante ou inteiro
iat	6	número de ponto flutuante ou inteiro
cti	7	caracteres de byte

Por exemplo, em uma mensagem da estrutura CWT[44], usado neste trabalho, é um vetor contendo sete elementos como ilustrado no Código 3.

```
1 / iss / 1: "coap://as.example.com", //emissor
2 / sub / 2: "erikw", //assunto
3 / aud / 3: "coap://light.example.com", //público ou audiencia
4 / exp / 4: 1444064944, //data de expiração
5 / nbf / 5: 1443944944,//não usar antes desta data
6 / iat / 6: 1443944944,//data de emissão
7 / cti / 7: h'0b71' //identificador CWT
```

Código 3 – Exemplo CWT

[43] Summary of the Claim Names, Keys, and Value Types: https://tools.ietf.org/html/rfc8392#section-4
[44] *Example CWT Claims Set*: https://tools.ietf.org/html/rfc8392#appendix-A.1

Chave como Prova de Posse para CWT. Em alguns contextos, há necessidade de estabelecer uma comunicação confiável e segura vinculando algum uma prova de posse (***P**roof-**o**f-**P**ossession*, PoP) (ex. chave criptográfica ou identificador para a chave) ou assegurador da chave (*holder-key*), aplicado na carga útil que está sendo transferido, para demonstrar autenticidade e confidencialidade. Dessa maneira, as chaves vinculadas sobre os conjuntos de atributos (*claims)* do CWT, pode oferecer prova de posse para CWT[45].

A prova de posse no formato CWT ocorre com adição do *claim* no atributo de confirmação ("`cnf`") que consiste de um mapa CBOR, contendo membros que identificam a chave de PoP (JONES et al., 2019). Para usar a PoP no formato CWT, o emissor do CWT, declara-se que o apresentador possui uma chave particular e que o destinatário confirma criptograficamente que o apresentador tem a posse daquela chave. As partes envolvidas que participam do processo de prova de posse são:

- **Emissor** (*Issuer*): Cria o CWT e vincula as afirmações verdadeiras (*claims*) sobre o tópico para chave PoP. No contexto do OAuth 2.0 esta parte envolvida é também conhecida como servidor de autorização.
- **Apresentador** (*Presenter*): Apresenta a PoP de uma chave privada (para criptografia de chave assimétrica) ou chave compartilhada (para criptografia de chave simétrica) para um destinatário. No contexto do OAuth 2.0 esta parte é também chamada cliente.
- **Destinatário** (*Recipient)*: Recebe CWT contendo a informação chave de PoP por meio do apresentador. No contexto do OAuth 2.0 esta parte envolvida é também conhecida como servidor de recurso do OAuth 2.0.

[45] *Proof-of-Possession Key Semantics for* CBOR *Web Tokens* (CWTs): https://tools.ietf.org/html/draft-ietf-ace-cwt-proof-of-possession-11

Um resumo entre a relação entre os participantes da prova de posse do CWT e OAuth 2.0 é apresentado na Tabela 3:

Tabela 3 – Relação entre os participantes da prova de posse do CWT

Semânticas de Chave PoP	OAuth 2.0
Emissor	Servidor de Autorização
Apresentador	Cliente
Destinatário	Servidor de Recurso

O atributo *claim* ("`cnf`") no CWT é usado para carregar métodos de confirmação. Vale salientar que alguns deles usam chaves PoP enquanto outros não. O valor do *claim* ("`cnf`") deve representar apenas uma chave simples da PoP. As representação do método com a chave assimétrica utiliza ("`COSE_key`"), representação do método com a chave simétrica utiliza ("`Encrypted_COSE_Key`") e representação por meio de método de identificado chave (key ID) para uma chave de PoP utiliza identificada do chave ("`kid`"), Tabela 4.

Tabela 4 – Sumário dos nomes cnf, chaves e o valor do tipo.
Fonte: (JONES et al., 2019).

Nome	Chave	Valor do Tipo
COSE_Key	1	COSE_Key
Encrypted_COSE_Key	2	COSE_ENCRYPT ou COSE_ENCRYPT0
Kid	3	Caracteres binários

Como mostrado na Tabela 4 o método utilizado para o atributo ("`cnf`") que carrega a prova de posse, no desenvolvimento deste trabalho, foi à chave simétrica ("`Encrypted_COSE_Key`") por meio da instância ("`COSE_ENCRYPT0`"), pois esta chave é mais adequada para atender as restrições de IoT e assegurar a confidencialidade na troca de credenciais de acesso

Baseado nesse tipo de prova de posse ("`COSE_ENCRYPT0`"), quando a chave simétrica é mantida pelo apresentador[46], o membro ("`Encrypted_COSE_Key`") é

[46] *Representation of an Encrypted Symmetric Proof-of-Possession Key*: https://tools.ietf.org/html/draft-ietf-ace-cwt-proof-of-possession-11#section-3.3

uma COSE_Key criptografada representando a chave simétrica e criptografada por meio de uma chave conhecida pelo destinatário usando (“COSE_Encrypt”) ou (“COSE_Encrypt0”) (JONES et al., 2019).

No Código 4, a representação da estrutura (“COSE_Key“) mostra o uso de uma chave simétrica[47] para criptografar o conteúdo que se quer proteger e depois outro processo criptografia é aplicado sobre toda estrutura ("Encrypted_COSE_Key") em que a chave é conhecida e compartilhada entre o servidor de autorização e o servidor de recurso.

```
1  /kty/ 1 : /Symmetric/ 4,
2  /alg/ 3 : /HMAC 256-256/ 5,
3  /k/ -1 :
h'6684523ab17337f173500e5728c628547cb37dfe68449c65f885d1b73b49eae1'
```

Código 4 – Exemplo COSE_Key.

Fonte: (JONES et al., 2019).

Na Código 5 ("Encrypted_COSE_Key") foi criptografado com a chave simétrica (“h'616263040506070809a0b0c0d0e0f10'”):

```
1 {
2    /iss/ 1 : "coaps://server.example.com",
3    /sub/ 2 : "24400320",
4    /aud/ 3: "s6BhdRkqt3",
5    /exp/ 4 : 1311281970,
6    /iat/ 5 : 1311280970,
7    /cnf/ 8 : {
8    /Encrypted_COSE_Key/ 2 : [
9        /protected header/ h'A1010A' /{ \alg\ 1:10 \AES-CCM-16-64-128\}/,
10       /unprotected header/ { / iv / 5: h'636898994FF0EC7BFCF6D3F95B'},
11       /ciphertext/  h'0573318A3573EB983E55A7C2F06CADD0796C9E584F1D0E3E
12                       A8C5B052592A8B2694BE9654F0431F38D5BBC8049FA7F13F'
13     ]
14    }
15  }
```

Código 5 – Exemplo Encrypted_COSE_Key.

Fonte: (JONES et al., 2019).

Dessa maneira, a partir do ACE-OAuth, adota-se novos protocolos adicionais ou reuso de protocolos já existentes na Internet e na Web, construindo uma pilha alternativa de protocolos para ser usada em IoT. Na Figura 40, e mostra o comparativo na estrutura dos dados da Internet e da IoT.

[47] 128-*Bit Symmetric Key*: https://tools.ietf.org/html/rfc8392#appendix-A.2.1

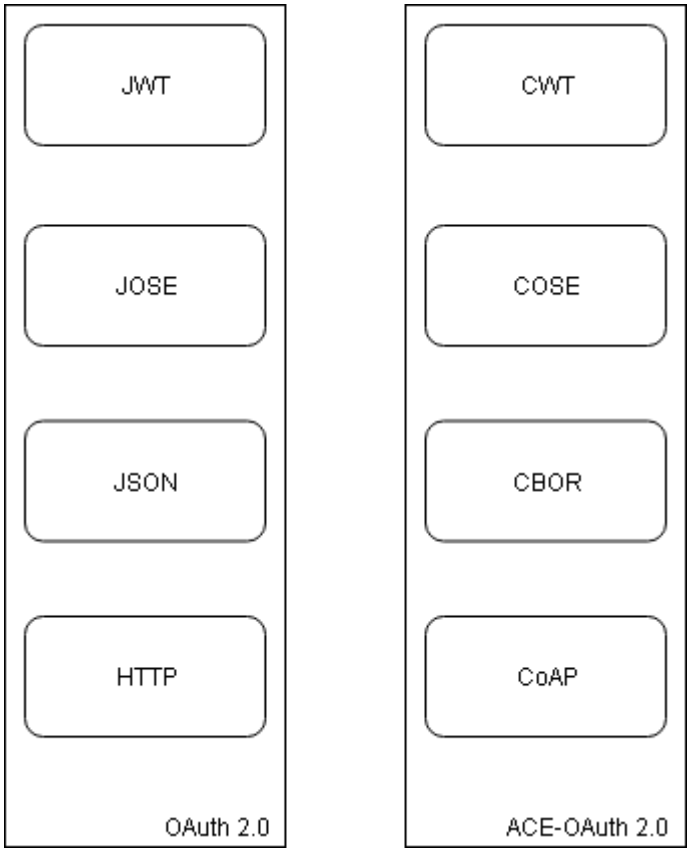

Figura 40 – Comparativo entre componentes do frameworks OAuth 2.0 e ACE-OAuth.

2.8.3 Fluxo do ACE-OAuth

As partes envolvidas no fluxo do ACE-OAuth são os mesmos atores do OAuth 2.0 que estão descritas na seção (2.7.2). Entretanto, os *endpoints* do ACE-OAuth se diferenciam do *endpoints* do OAuth 2.0[48]. Isso porque o ACE-OAuth possui além dos *endpoints* obrigatórios que são semelhantes aos do ACE-OAuth 2.0 *Endpoint* de Autorização *(Authorization Endpoint*, AE), *Endpoint* do *Token* (*Token Endpoint*, TE), *Endpoint* do Cliente (*Client Endpoint*, CE), possui o *endpoint* opcional o *Endpoint* de Introspecção (*Introspection Endpoint*, IE). A Tabela 5 mostra a relação entre atores e os *endpoint*s do ACE-OAuth.

[48] *Protocol Endpoints*: https://tools.ietf.org/html/rfc6749#section-3

Tabela 5 – Relação entre atores e os *endpoints* do OAuth 2.0

Entidade	*Endpoint*
Cliente	Retorno
Servidor de Autorização	Autorização
	Token
	Introspeção
Servidor de Recurso	Recurso
	Introspeção

***Endpoint* de Autorização**. É suportado pelo servidor de autorização, o servidor de autorização envia uma notificação para o dono do recurso. O dono do recurso que recebe uma solicitação de autorização sobre o recurso oferecido pelo servidor de recurso. Em seguida, o dono de recurso realiza uma requisição ao servidor de autorização com informação do controle de permissão baseado no escopo, por meio da decisão de consentimento, homologando ou revogando, o acesso do cliente ao recurso.

***Endpoint* do Cliente**. Realiza as solicitações ao servidor de autorização para obtenção do *token* e para poder consumir recursos através do servidor de recursos.

***Endpoint* de Introspecção**. É oferecido pelo servidor de autorização e pode ser usado pelo servidor de recurso, se for necessária uma requisição de informação adicional relacionada a um *token* de acesso recebido. Dessa forma, o servidor de recurso faz uma requisição para o terminal de introspecção sobre o servidor de autorização. O servidor de autorização recebe a informação sobre ela e a processa. Se a requisição for processada com sucesso, o *token* de acesso é retornado na resposta.

Endpoint* do *Token. É hospedado pelo servidor de autorização, o que permite o cliente realizar a requisição do *token* de acesso. Dessa forma, o cliente faz uma requisição para o terminal do *token* sobre o servidor de autorização, depois, o servidor de autorização recebe e a processa. Se a requisição for processada com sucesso, o *token* de acesso é retornado na resposta.

Endpoint* de Acesso ao *Recurso. É hospedado pelo servidor de recurso, o que permite o cliente realizar a requisição do recurso. Dessa forma, o cliente faz uma requisição para o terminal do bem-conhecido sobre o servidor de recurso, depois, o servidor de recurso recebe e a processa. Se a requisição for processada com sucesso, o *recurso* é retornado na resposta.

A Figura 41 ilustra a interação realizada entre o cliente, servidor de autorização, servidor de recurso, e o dono do recurso como concebido pelo grupo de trabalho ACE do IETF[49]. Além do mais, nota-se que a interação entre as partes consiste em cinco etapas básicas e duas são opcionais. As opcionais se concentram, principalmente, na comunicação entre o servidor de recurso e o servidor de autorização.

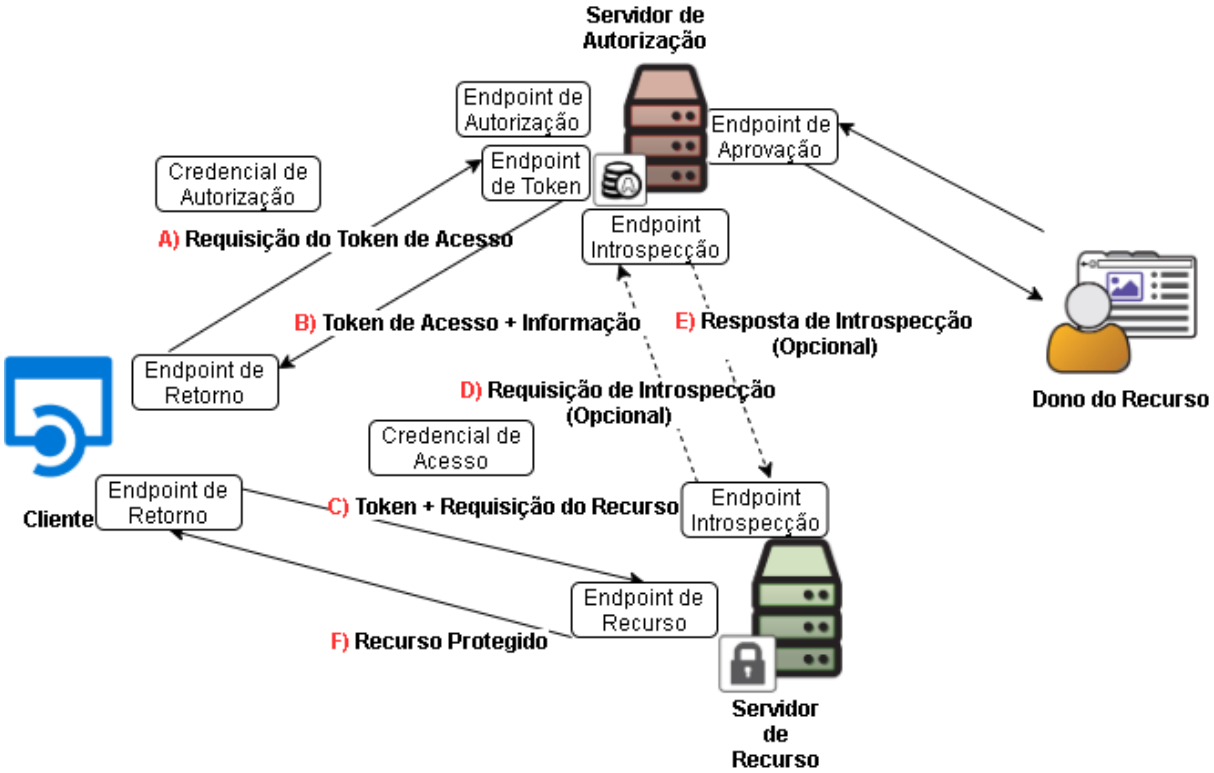

Figura 41 – Fluxo do protocolo OAuth 2.0 seguido pelo modelo proposto ACE-OAuth.
Fonte: Próprio autor.

Como apresentado às diversas etapas do fluxo do ACE-OAuth são apresentadas na Figura 41, de acordo com (SEITZ et al., 2020). Embora a

[49] *Authentication and Authorization for Constrained Environments* (ace): https://datatracker.ietf.org/wg/ace/documents/

especificação não evidencie a requisição e resposta na etapa de autorização no fluxo, de todo modo, elas pertencem ao fluxo de concessão código de autorização. A descrição das etapas apresentado na Figura 41, é destacado a seguir.

- **Requisição do *Token* de Acesso**: O cliente faz uma requisição ao *endpoint* do *token* no servidor de autorização.
- **Resposta do *Token* de Acesso**: Se o servidor de autorização processar com sucesso a requisição do cliente, ele retorna um *token* de acesso.
- **Requisição do Recurso**: O cliente interage com o servidor de recurso para solicitar acesso para o recurso protegido e oferece o *token* de acesso.
- **Requisição de Introspecção**: O servidor do recurso pode ser configurado para fazer uma introspecção realizando uma requisição para o servidor de autorização, mas essa etapa é opcional.
- **Resposta de Introspecção**: O servidor de autorização valida o *token* e retorna os parâmetros mais recentes associados com o *token* para o servidor de recurso, mas essa etapa é opcional.
- **Recurso Protegido**: Se a requisição do cliente for autorizada, o servidor de recurso executa a requisição e retorna uma resposta com o código de resposta apropriado.

2.9 DOS TIPOS DE AMEAÇAS E ATAQUES

Os ataques cobertos neste trabalho são os ataques de personificação e o ataque de replicação. Esses ataques são oriundos do uso inseguro e ineficaz da credencial no controle de acesso em ambiente de IoT. Para mitigar esses ataques e aumentar confiança no controle de acesso em ambiente de IoT, requisitos como autenticação mútua, confidencialidade e integridade na troca de credenciais entre cliente e o servidor de autorização e entre o cliente e o servidor de recurso são necessários para reforçar o controle de acesso (BELTRAN; SKARMETA, 2018).

Ataque de personificação. Visa capturar a credencial de identificação (identificador, segredo), a credencial de autorização (código de autorização) e credencial de acesso (*token* de acesso) na requisição do cliente para o servidor e na resposta do servidor para cliente, se não houver uma camada de segurança. Uma vez a

credencial capturada, ela pode ser usada para realizar ações em nome do cliente original.

Ataque de replicação. Refere-se à transmissão de dados válidos repetida maliciosamente ou fraudulentamente, entre solicitante ou uma terceira parte que intercepta o dado e o retransmite, possivelmente, como ataque disfarçado ou mascarado (SHIREY, 2007). No ataque de replicação, a partir da origem para o destino, o adversário copia mensagem do remetente e replica mais tarde. Mas, um ataque sofisticado pode ocorrer no envio da mensagem que é aceita dentro de uma janela de tempo válida, com isso o adversário pode replicar uma propriedade de tempo na mensagem, que interrompe ou não a mensagem original, ocasionando um conflito quando a mensagem chega ao receptor (GONG, 1993). Outro tipo de ataque de replicação ocorre no caminho inverso, isto é, a mensagem de volta ao emissor. Nessa ocasião, a fragilidade acontece quando o emissor não consegue diferenciar as respostas do destinatário (GONG, 1993).

De qualquer maneira, sabendo que as vulnerabilidades presentes no OAuth 2.0 podem ocasionar ataques de replicação e personificação, é importante analisar a adaptação do OAuth 2.0 para ambientes de IoT por meio do ACE-OAuth. Para isso, as ameaças catalogadas no OAuth 2.0 podem servir como aprendizado para o ACE-OAuth. Além do modelo de ameaças do OAuth 2.0, outras investigações (LI, 2017) (LODDERSTEDT; MCGLOIN; HUNT, 2013) (SUN; BEZNOSOV, 2012) contribuem no avanço na identificação das ameaças e na mitigação aos ataques de personificação e replicação.

2.9.1 Obtenção dos Segredos do Cliente

Ameaça. O atacante poderia tentar obter acesso ao segredo (*token*s de acesso ou de *refresh* e códigos de autorização) de um cliente e replicá-los para se passar pelo cliente a fim de obter privilégios, Figura 42.

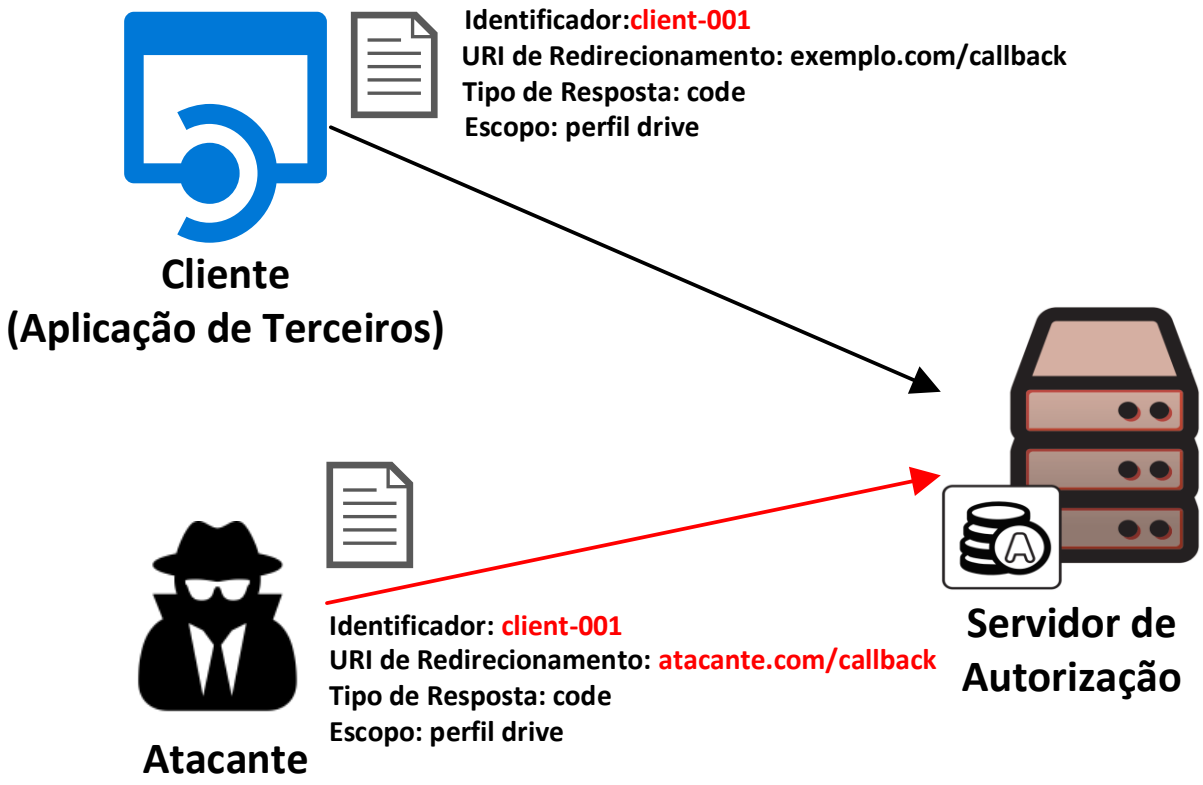

Figura 42 – Obtenção dos segredos do cliente.

Impacto. Na Figura 42, a autenticação do cliente no acesso ao servidor de autorização pode passar despercebido ou *token*s de *refresh* roubados ou códigos de autorização podem ser replicados.

2.9.2 Divulgação das Credenciais do Cliente durante a Transmissão

Ameaça. Um invasor pode tentar interceptar a transmissão de credenciais entre o cliente e o servidor durante o processo de autenticação do cliente ou durante solicitações de *token*, Figura 43 a Figura 47.

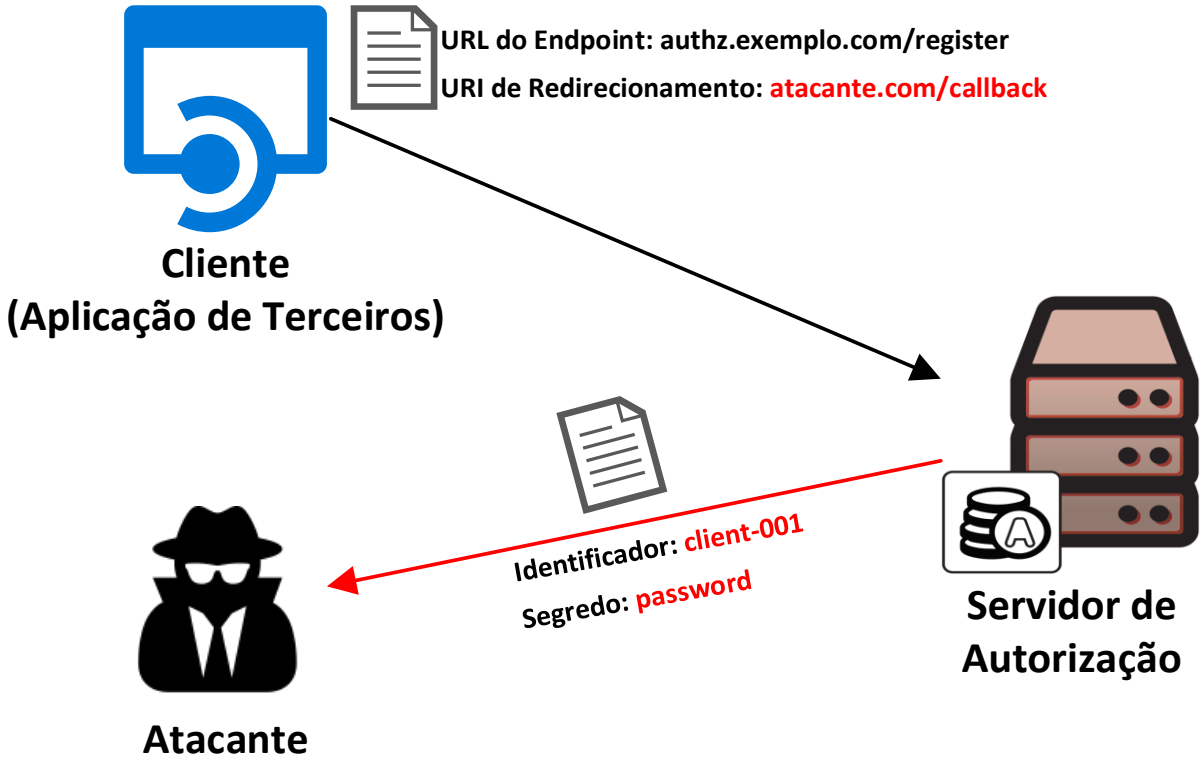

Figura 43 – Obtenção da credencial de identidade.

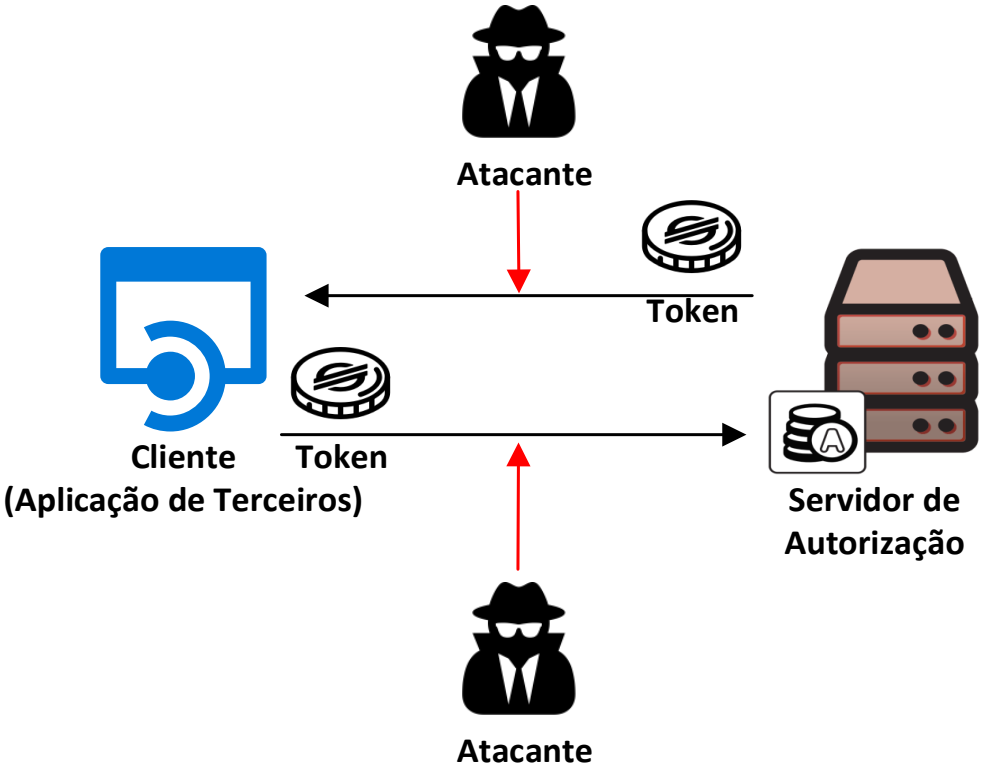

Figura 44 – Espionagem na comunicação entre cliente e servidor de autorização.

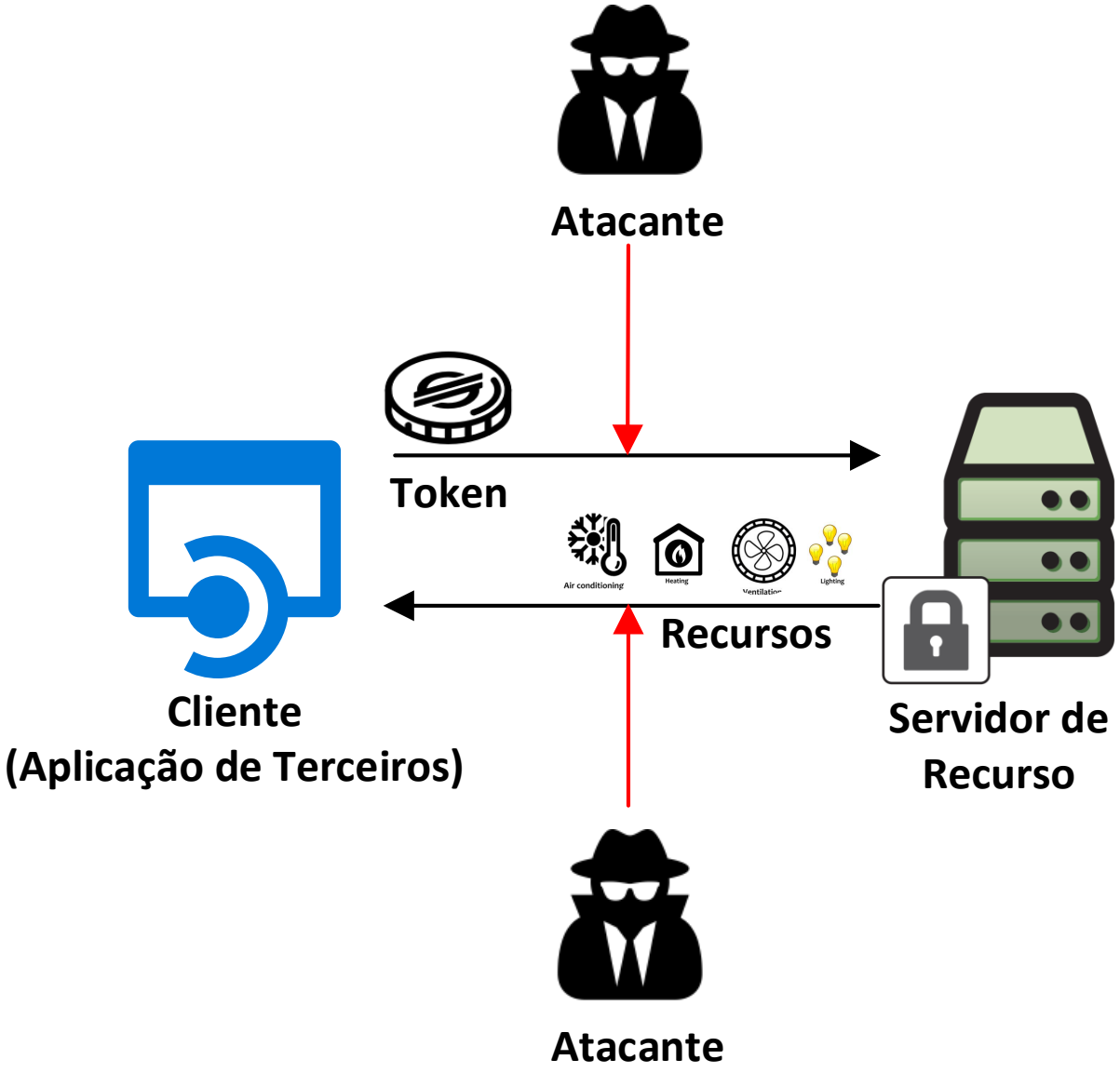

Figura 45 – Espionagem na comunicação entre cliente e servidor de recurso.

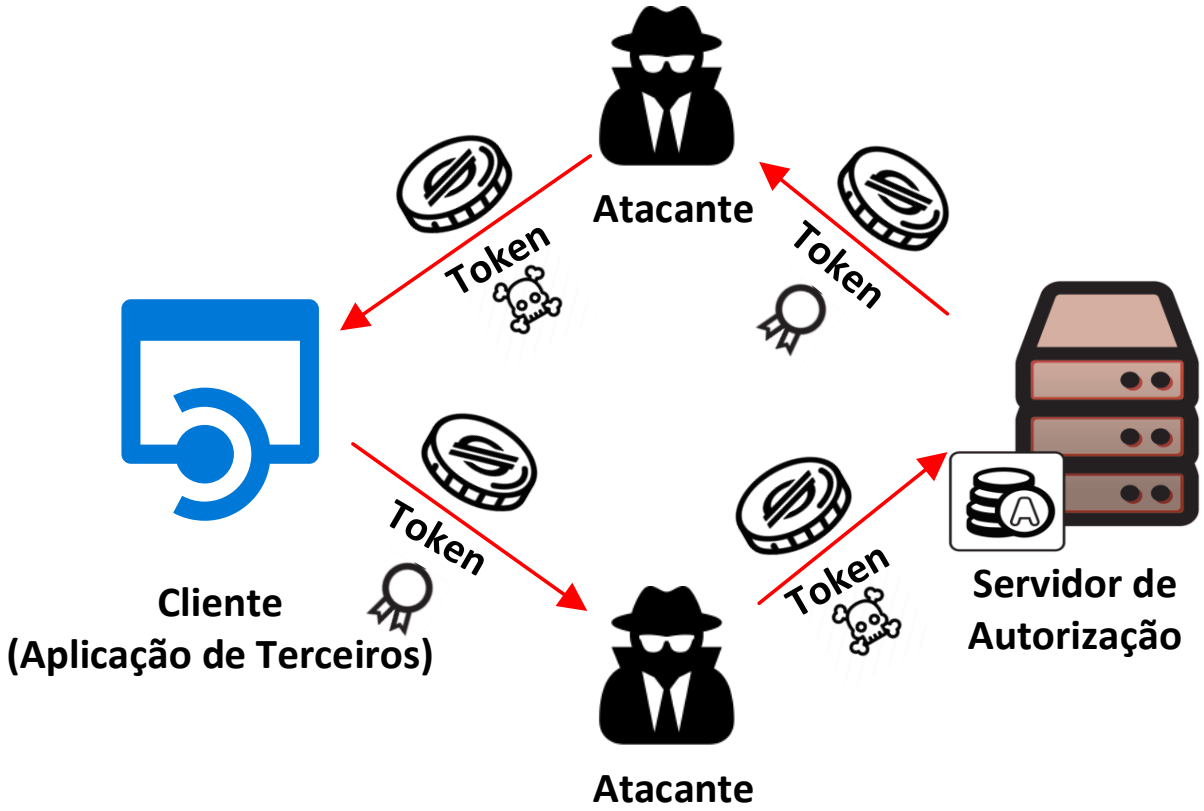

Figura 46 – Interceptação na comunicação entre cliente e servidor de autorização.

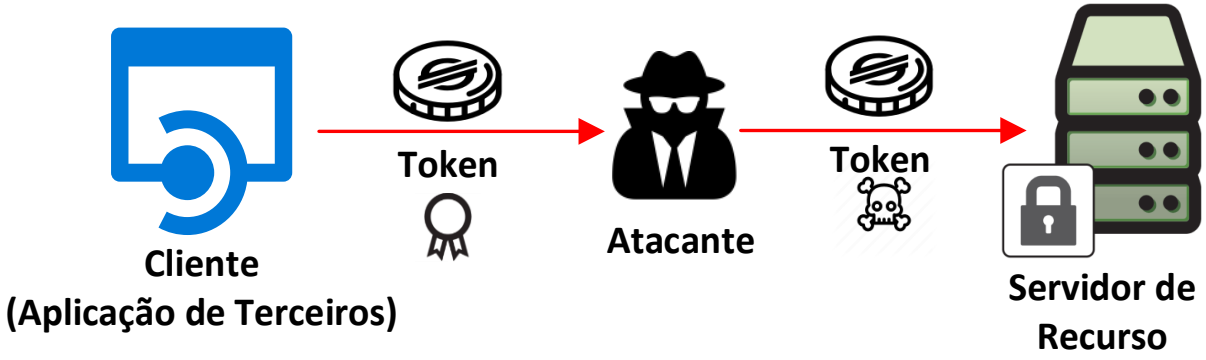

Figura 47 – Interceptação na comunicação entre cliente e servidor de recurso.

Impacto. Na Figura 43 e na Figura 45, a revelação de uma credencial do cliente permite realizar a pesca (*phishing*) e posterior personificação de um serviço ao cliente.

2.9.3 Pesca do Código de Autorização

Ameaça. Um atacante pode se passar por um sítio (*site*) do cliente e obter acesso ao código de autorização, Figura 48. Isso pode ser alcançado usando falsificação no Sistema de Nomes de Domínio (*Domain Name System*, DNS) ou Protocolo de Resolução de Nomes (*Address Resolution Protocol*, ARP). Isso se aplica aos

clientes, que usam redirecionamento de URI, que não seja para o local do nó de origem onde o navegador do usuário ou aplicação está executando.

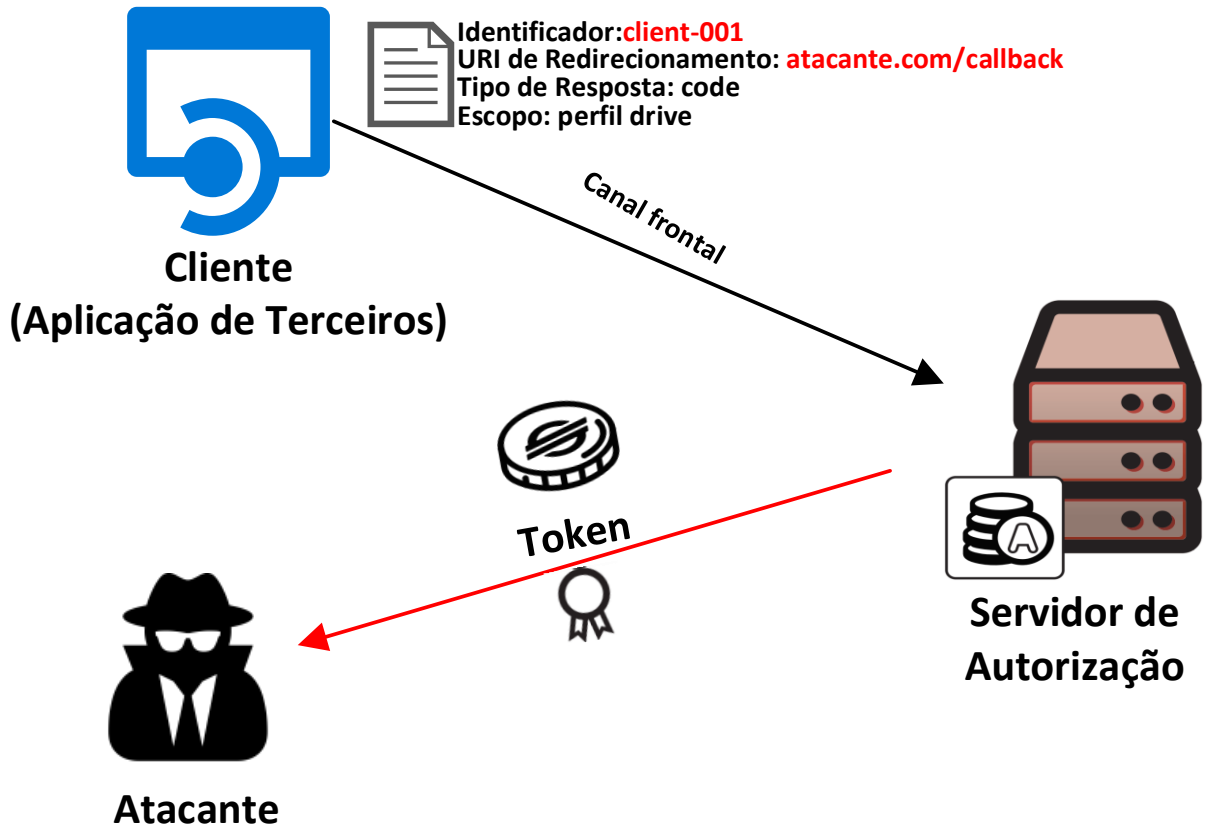

Figura 48 – Pesca do código de autorização.

Impacto. Na Figura 48, o ataque afeta os aplicativos que se utilizam do parâmetro de URI redirecionamento e pode levar à divulgação de código de autorização e, potencialmente, o acesso e atualização dos *token*s.

2.9.4 Personificação da Sessão do Usuário

Ameaça. Um atacante pode se passar pelo lado cliente e obter acesso na sessão do usuário, Figura 49. Isso pode ser alcançado usando falsificação DNS ou ARP. Isso se aplica aos clientes, que usam URI de redirecionamento que não seja para o local de origem onde o navegador do usuário está executando.

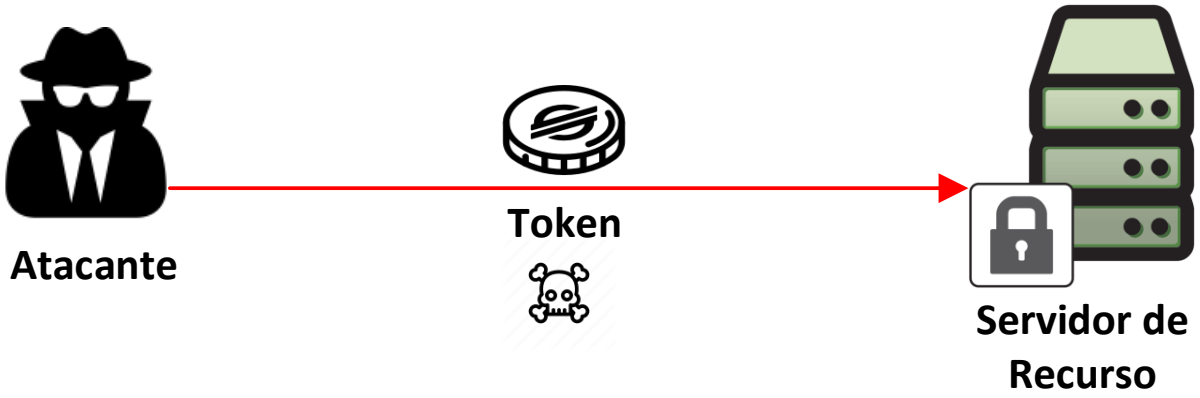

Figura 49 – Personificação da sessão do usuário.

Impacto. Um atacante que tenha capturado anteriormente o *token*, pode acessar indevidamente os recursos protegidos, Figura 47. Um segundo caso, seria um invasor que tenha interceptado o código de autorização que é uma das exigências para obter um *token* de acesso a fim de acessar recursos protegidos, Figura 42. Nesta ocasião, o atacante pode apresentar o código de autorização capturado e fazer com que o servidor encaminhe a resposta para o *endpoint* de retorno de chamada (*callback*) do atacante ao invés de enviá-la para o cliente.

“

3 Trabalhos Relacionados

“A mudança não acontecerá se nós esperarmos por outra pessoa ou se esperamos por algum outro momento. Nós somos as pessoas pelas quais esperávamos. Nós somos a mudança que buscamos”.

Obama.

Neste capítulo é apresentado o resultado da investigação do levantamento do estado da arte assim como é realizado um comparativo entre a solução proposta e as propostas dos trabalhos relacionados em termos do perfil ACE-OAuth, seus protocolos de comunicação e de segurança e, por último, dos casos de uso para automação residencial.

3.1 ESTADO DA ARTE

No trabalho de Claeys, Rousseau e Tourancheau (2017), os autores propõem a implementação do modelo ACE-OAuth a partir do OAuth 1.0. Nesse estudo, foram inseridos métodos de criptografia e também um esquema de estabelecimento de chaves para os casos de uso utilizados em que os dispositivos não podem se comunicar diretamente. Os autores também usaram a Infraestrutura de Chave Pública (*Public Key Infrastructure*, PKI) para estabelecer uma cadeia de confiança entre os dispositivos a fim de simplificar as trocas futuras do *token*. Neste estudo é utilizado o Diffie Hellman Efêmero sobre COSE (*Ephemeral Diffie Hellman over Cose*, EDHOC) e o Estabelecimento de Chave Autenticada (*Authenticated Key Establishment*, AKE). Entretanto, destacam-se dois pontos de desvantagens desse trabalho, primeiro, um ponto único de armazenamento de chaves e, segundo, a verificação do *token* que usa uma quantidade significativa de memória e processamento. Já em relação à avaliação de desempenho, o trabalho apresentou que a atualização do *token* é custosa sobre a latência e a verificação do *token* impacta no consumo de energia.

Por outro lado, Sciancalepore et al. (2017) apresentaram o OAuth-IoT. Neste trabalho, os autores aplicam a ideia de um *gateway* o qual realiza o controle de

acesso sobre as aplicações de terceiros aos dispositivos com limitação de recursos. Além disso, os autores apresentam um mecanismo de *cache* que é esquematizado em três componentes: tabela de roteamento, diretório de recursos e tabela de dados. Dessa forma, por meio desse esquema evita-se que a requisição realize mais saltos até chegar à parte que fornece a informação do recurso. Consequentemente, há um aumento no tempo de resposta quando todas as estratégias foram adotadas no modelo de avaliação de desempenho.

No trabalho de Gerber (2018), o autor codificou como prova de conceito o perfil de segurança OSCORE e esquema de troca de chave Diffie-Hellman para proteger a troca de mensagens baseado no ACE-OAuth, o protótipo da prova de conceito foi escrito em linguagem de programação Python. Além disso, o autor implantou dispositivo embarcado em uma aplicação que atua como servidor de recursos capaz de executar computações criptográficas assimétricas, escrito na linguagem de programação C.

Por outro lado, Fremantle (2018) propõe um modelo e protótipo funcional para IoT por meio do OAuthing. O OAuthing caracteriza os usuários e dispositivos por meio de instâncias alocadas por serviços em nuvem. Segundo resultados preliminares, houve melhorias significativas de segurança e privacidade devido ao desacoplamento entre o gerenciamento da identidade e o da autorização. Contudo, houve incremento na latência, aumento no uso da memória do dispositivo assim como no consumo de energia em comparação a outra linha base.

Contudo, no trabalho de Aragon et al. (2018) é apresentado um perfil IPSec ao ACE-OAuth. Esse perfil especifica como o cliente deve estabelecer um canal seguro na camada de rede com o servidor de recursos para controlar o acesso autorizado sobre os recursos. Para isso, os autores consideram o protocolo IPsec e o protocolo de gerenciamento de chave IKEv2. Contudo, a principal desvantagem desse trabalho é o impacto no tráfego. Além disso, houve aumento no tamanho da mensagem quando transmitido o *token* de acesso o que acaba prejudicando a comunicação dos dispositivos com restrição de recursos.

Dessa maneira, soluções apresentadas na literatura usam a assinatura digital, tunelamento da comunicação e os controles de acesso sem serviço de confidencialidade. Entretanto, o processo de geração e verificação da assinatura digital requer considerável gasto computacional. Por outro lado, o tunelamento da comunicação (ex. IPSec) é custoso e complexo, o que impacta no desempenho no

tráfego ou na rede. Por último, os controles de acesso, quando não aplicados em conjunto com o estabelecimento do canal de segurança, expõem a credencial. Entretanto, nenhuma dessas soluções aplicou mecanismos de proteção nas requisições feitas pela aplicação cliente sobre o servidor de autorização no fluxo de código de autorização no ACE-OAuth.

Para minimizar os riscos em torno do fluxo código de autorização no ACE-OAuth, nas requisições do cliente para o servidor de autorização, a solução proposta adiciona dois fluxos de concessão adicionais no fluxo original do código de autorização no ACE-OAuth. O primeiro é o fluxo de concessão PKCE e o segundo é o fluxo de concessão de veracidade. No fluxo de concessão PKCE, o método de desafio-resposta é incluído entre a aplicação cliente e o servidor de autorização. O objetivo do fluxo de concessão PKCE é assegurar a origem da requisição seja a mesma na próxima requisição. Já no fluxo de concessão de veracidade, é introduzido um conjunto de declarações verdadeiras sobre a identidade do solicitante da requisição. Neste fluxo, a aplicação cliente declara um conjunto de atributos verdadeiros que o credencia para realizar alguma operação ou consumir algum recurso e o servidor verifica os valores contidos nesses atributos confirmam se o cliente é legítimo ou não para tais solicitações.

3.2 COMPARAÇÃO ENTRE AS PROPOSTAS

Para compreender melhor a estrutura das soluções em termos de comunicação e segurança foi realizada uma comparação entre os trabalhos relacionados em termos de características recomendadas e opcionais do ACE-OAuth, na Tabela 6. A proposta agrupa conceitos abordados nos trabalhos relacionados, envolvendo características, funcionalidades e configurações que serão apresentadas no Capítulo 4.

Tabela 6 – Comparativo entre a solução proposta e as soluções dos trabalhos relacionados em relação ao fluxo da solicitação do *token*.

Perfis de Segurança	Protocolo de Aplicação	Protocolo de Troca de Mensagem	Camada de Segurança do Protocolo de Troca de Mensagem	Estrutura da Codificação dos Dados	Perfil de Segurança	Esquema de Troca de Chave
Trabalho proposto	CoAP	CBOR	COSE	CWT	DTLS v 1.2	-
(GERBER, 2018)	HTTP	CBOR	COSE	CWT	TLS + OSCORE	EDHOC
(FREMANTLE; AZIZ, 2018)	-	-	-	-	-	-
(ARAGON et al., 2018)	-	-	-	-	-	IKEv2
(CLAEYS; ROUSSEAU; TOURANCHEAU, 2017)	-	-	-	-	-	EDHOC

Na Tabela 7, é realizado um comparativo das características opcionais no ACE-OAuth entre a proposta e os trabalhos relacionados.

Tabela 7 – Comparativo de características opcionais no ACE-OAuth.

Perfis de Segurança	Alguma Técnica de Gerenciamento de Identidade	Aplicação de Segurança além da Camada de Aplicação	Implementação no lado cliente
Trabalho proposto	✓	✗	✗
(GERBER, 2018)	✗	✗	✓
(FREMANTLE; AZIZ, 2018)	✓	-	-
(ARAGON et al., 2018)	-	✓	-
(CLAEYS; ROUSSEAU; TOURANCHEAU, 2017)	-	-	-

Adicionalmente, foram levados em consideração, para fins de comparação entre as funcionalidades tanto da solução proposta quanto as soluções já existentes observadas dos trabalhos relacionados, os casos de uso em automação residencial, Tabela 8.

Tabela 8 – Casos de uso em automação residencial que as soluções atendem.

Caso de Uso em Automação Residencial	Solução			
Descrição	Perfil IPSec ACE-OAuth	Oauthing	OAuth-IoT	Proposta
O proprietário de uma casa deseja fornecer espontaneamente meios de autorização aos visitantes.	✓	✓	✓	✓
O proprietário da casa deseja alterar espontaneamente as políticas de controle de acesso da casa.	✗	✗	✗	✗
Um proprietário da casa quer aplicar diferentes direitos de acesso para diferentes usuários (incluindo outros habitantes).	✗	✗	✗	✗
Os proprietários da casa desejam conceder permissões de acesso a alguém durante um período de tempo especificado.	✓	✓	✓	✓
Os dispositivos domésticos inteligentes precisam ser capazes de se comunicar com segurança com diferentes dispositivos de controle (por exemplo, painéis de toque montados na parede, smartphones, porta-chaves eletrônicos e gateways de dispositivos).	✓	✗	✓	✗
O proprietário da casa deseja ser capaz de configurar as políticas de autorização remotamente.	✗	✗	✗	✗
Usuários autorizados querem ser capazes de obter acesso com pouco esforço.	✗	✗	✗	✗
Os proprietários da central automatizada querem impedir que entidades não autorizadas possam deduzir perfis comportamentais de dispositivos na rede doméstica.	✗	✗	✗	✗

Caso de Uso em Automação Residencial	Solução			
Descrição	Perfil IPSec ACE-OAuth	Oauthing	OAuth-IoT	Proposta
Maior usabilidade para realizar as tarefas relacionadas à autorização dos dispositivos pode ser complicada. Os proprietários que, na maioria dos casos, têm pouco conhecimento de segurança.	✘	✘	✘	✘
Os proprietários das residências querem que seus dispositivos atinjam perfeitamente seu propósito (e, em alguns casos, até mesmo imperceptível). Portanto, o esforço de administração de autorização precisa ser mantido no mínimo.	✔	✔	✔	✔
Os proprietários de casas querem ser capazes de transferir a propriedade de seus sistemas automatizados quando vendem a casa.	✘	✘	✘	✘
Os proprietários das casas querem ser capazes de limpar/higienizar os registros dos sistemas automatizados ao transferir a propriedade sem excluir dados operacionais importantes.	✘	✘	✘	✘
Quando ocorre uma transferência de propriedade, o novo proprietário deseja garantir que os direitos de acesso criados pelo proprietário anterior não sejam mais válidos.	✘	✘	✘	✘

Na Tabela 8, é constatado que nenhuma das soluções observadas pelo pesquisador alcançou todos os requisitos dos casos de uso para automação residencial. No estado atual da proposta, apresentada nesta dissertação, alcançou (23%) dos casos de uso, um valor próximo da média dos casos de uso alcançados pelas soluções observadas.

4 Da solução proposta e sua implementação

“Se você não for teimoso, abandonará os experimentos cedo demais. E se você não for flexível, baterá a cabeça contra a parede e não verá uma solução diferente para o problema que está tentando resolver”.

Jeff Bezos.

Neste capítulo, é apresentada a solução proposta, são descrito seus componentes, protocolos e o seu perfil de comunicação e de segurança. Na Seção 4.4, são apresentadas as medidas de mitigação baseada nas características e nos objetivos de segurança da solução proposta contra ameaças e ataques de personificação e replicação. Por último, para assegurar a eficácia das medidas de segurança da solução proposta foram realizadas uma avaliação de não conformidade de ameaças (Seção 4.3) e uma análise de riscos baseada em árvore de ataque (Seção 4.6).

4.1 DO PROBLEMA

As ameaças e consequentes ataques de personificação e de replicação têm como uma das principais razões o uso ineficaz e inseguro das credenciais. Essas ameaças e ataques possuem custo e esforço mais baixos em relação à segurança em domínios de IoT, como no caso da automação residencial (DEEP; ZHENG; HAMEY, 2019).

Em ambientes de IoT, soluções como o ACE-OAuth usa o fluxo código de autorização no controle de acesso aos recursos por parte de aplicações de terceiros. Entretanto, uma das etapas do fluxo de concessão código de autorização no ACE-OAuth, a requisição de autorização, por exemplo, fica suscetível às ameaças e ataques de personificação e replicação. Isto ocorre dado que não há o controle de confirmação de origem e verificação da identidade podendo, dessa forma, permitir a exposição da credencial na troca de mensagens entre as partes (BELTRAN; SKARMETA, 2018).

4.2 DA SOLUÇÃO

Dessa maneira para atingir objetivos de segurança de sistemas em IoT, é necessária a identificação de funcionalidades e objetivos de determinados componentes de segurança (ABOMHARA; KØIEN, 2014), conforme pode ser visto na Tabela 9.

Tabela 9 – Componentes que implementam funcionalidades de segurança em IoT.
Fonte: (ABOMHARA; KØIEN, 2014).

Componente	Funcionalidade do Componente	Objetivo de Segurança
Autorização	Controle de Acesso das Aplicações aos Dispositivos e Serviços	Confidencialidade de Dados Integridade de Dados
Autenticação	Autenticação de Aplicações, Dispositivos e Serviços dos Usuários	Autenticação Auditoria
Gerenciamento de Identidade	Gerenciamento de Identidades, Pseudônimos, Políticas de Acesso Relacionadas	Privacidade do Usuário Privacidade de Serviço Privacidade da Aplicação do Usuário
Gerenciamento e Troca de Chave	Troca de chaves criptográficas	Integridade de Comunicação Confidencialidade da Comunicação
Gerenciamento Confiável e Reputação	Nível de Confiança de Serviço e Coleção de Pontuação de Reputação do Usuário	Confiança de Serviço Reputação do Serviço

Dessa forma, baseado nas funcionalidades e objetivos dos componentes de segurança, apresentados na Tabela 9, a solução proposta foi projetada para ter: a) a funcionalidade de controle de acesso das aplicações aos dispositivos recursos para atingir os objetivos de confidencialidade dos dados e de integridade dos recursos; b) a funcionalidade de autenticação de aplicações com objetivo de verificação da identidade da aplicação; c) a funcionalidade de gerenciamento de identidade para

atingir objetivo de privacidade do cliente em nome dele próprio e em nome do usuário; d) a funcionalidade de nível de confiável para atingir o objetivo de confiança no serviço. Para desenvolver essas funcionalidades e atingir os objetivos de segurança de sistema em IoT, a solução proposta adotou os mecanismos a seguir.

Fluxo de desafio-resposta. Para evitar ataques de personificação e de replicação oriundos da monitoração ou interceptação da credencial da autorização) no fluxo de código de autorização, utilizou o fluxo de concessão de desafio-resposta. O fluxo desafio-resposta contribui na confirmação da origem da requisição do cliente, já que o uso de atributos aleatórios (“`nonce`”) ou (“`state`”) são apontados como inseguros por motivos de descoberta e colisão (RICHER; SANSO, 2017).

Fluxo de veracidade. Para evitar ataques de personificação e de replicação partir da monitoração ou interceptação da credencial (de identificação e de acesso). Para isso, em vez da solicitação de consentimento passar pelo dono do recurso para autorizar a emissão da credencial de autorização (*código de autorização*) pelo servidor de autorização no original fluxo código de autorização, por meio do fluxo de veracidade em que o próprio servidor de autorização assume a responsabilidade de verificar o consentimento, sem a necessidade da participação do dono do recurso, uma vez que o servidor de autorização assume estabelecer vínculos de confiança com clientes confidenciais ou privados na emissão de credenciais de identificação. Os clientes confidenciais ou privados possuem o vínculo pré-estabelecido de confiança com a autoridade de autorização para poder acessar o sistema, diferente de clientes públicos que podem interagir com o sistema e sem pré-estabelecer relação de confiança ou atuarem com um grau de confiança mínimo.

Tipo de resposta ID Token. Para realizar o fluxo de veracidade é necessário usar uma estrutura de dados que dê suporte à verificação da identidade. A solução, OpenID Connect já oferece soluções de respostas que contribuem na veracidade e gerenciamento de identidade por meio do ID Token. No entanto, a solução proposta, adota nas respostas de autorização e de *token* um dos fluxos de resposta do OpenID Connect, que é baseado na estrutura ID Token. Além disso, no sistema proposto, O ID Token é codificado no formato serializado CWT.

Prova de posse. O uso da prova de posse serve para proteger a credencial de acesso na requisição do cliente ao servidor de recurso. A prova de posse substitui o mecanismo de autorização do portador de *token*, o qual é inseguro, quando não se estabelece canal de segurança, ou quando se usa mecanismo inadequado de criptografia ou não inutiliza assinatura sobre a mensagem. Dentro das opções da prova de posse, utilizou criptografia simétrica, como explicado na seção (4.2.4) sobre a utilização perfil CoAP/DTLS para a prova de posse, para proteger o conteúdo da credencial de acesso, o que possui custo aceitável para ambientes em IoT.

Portanto, os mecanismos apresentados visam reforçar o uso seguro e eficaz da credencial no controle de acesso aos recursos sobre as aplicações de terceiros em ambientes de IoT. Dessa maneira, a justificativa de escolha cada mecanismo de segurança utilizado na solução proposta é mostrado em contraste às ameaças e os ataques, na Tabela 10.

Tabela 10 – Relação entre etapas dos fluxos de concessão, credenciais usadas e ataques.

Solução	Credencial	Direcionado à Ameaça	Minimiza o risco de ataque
Fluxo Veracidade	Credencial de identificação(identificador) Credencial de acesso (*token*)	Personificação da sessão do usuário Divulgação das credenciais do cliente durante a transmissão Obtenção dos segredos do cliente	Ataque de personificação Ataque de replicação
Fluxo Desafio-Resposta	Credencial de autorização (código)	Pesca do código de autorização	Ataque de personificação Ataque de replicação
Prova de Posse	Credencial de acesso (*token*)	Personificação da sessão do usuário	Ataque de personificação Ataque de replicação
ID Token	Credencial de identificação(identificador) Credencial de autorização (código) Credencial de acesso (*token*)	Requisições não assinadas	Ataque de replicação,

4.2.1 Visão Geral da Solução

Para reforçar fluxo código de autorização no ACE-OAuth no combate aos ataques de personificação e replicação em ambiente de IoT, solução proposta deste trabalho, adota o fluxo de concessão desafio-resposta e fluxo de veracidade por meio da estrutura ID Token.

O fluxo desafio-resposta contribui na confirmação da origem da requisição do

cliente, já que o uso de atributos aleatórios (“nonce”) ou (“state”) são apontados como inseguros por motivos de descoberta e colisão (RISCHER; SANSO, 2017)

No fluxo de veracidade, os clientes confidenciais ou privados possuem o vínculo de confiança pré-estabelecido com a autoridade de autorização para poder interagir com a API. Isso significa que o cliente possui um segredo compartilhado armazenado para se autenticar ao conversar com o servidor de autorização (RICHER; SANSO, 2017), diferentemente dos clientes públicos que podem interagir sem de ter grau de confiança mínimo estão sujeitos a todos os tipos de ataques como roubo de *token* e de obtenção do código de autorização (RICHER; SANSO, 2017)

Já o uso do ID Token nas trocas de mensagens contribui para veracidade e a confirmação da identidade. A solução proposta adota as estratégias inspiradas no fluxo de comunicação do OpenID Connect que utilizam os tipos de resposta (“code”, “ID Token”) e (“token”, “ID Token”), respectivamente, nas respostas de autorização e de *token* dadas pelo lado servidor a fim de credenciar os clientes e obter uma grau de confiança maior em relação a autenticação.

4.2.2 Componentes da Solução

A Figura 50 mostra uma visão dos componentes da solução proposta.

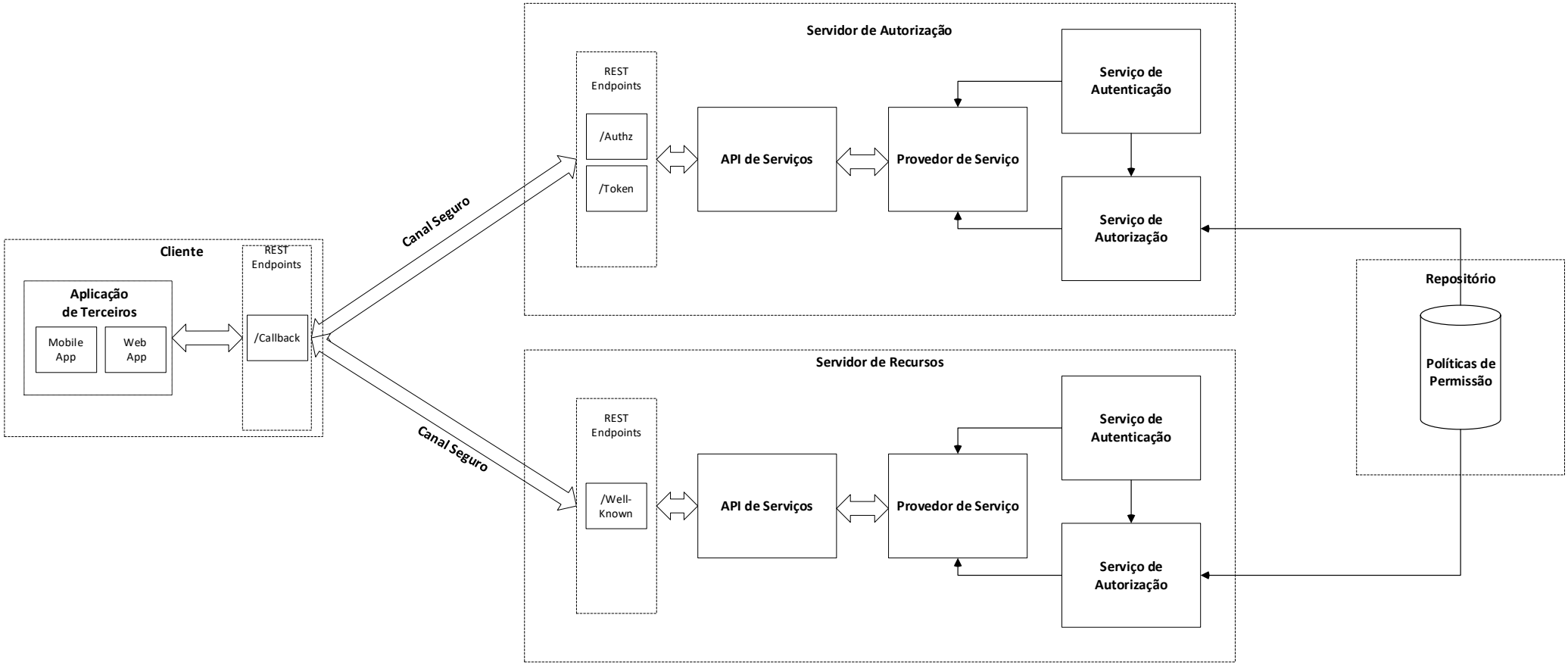

Figura 50 – Componentes da solução proposta.

Aplicação Cliente de Terceiros. Aplicação de terceiros que quer acessar ou consumir os recursos.

API de serviços. Consiste dos *endpoints* disponibilizados pelo provedor do serviço que funcionam como uma interface de comunicação. No servidor de autorização os *endpoints* fornecidos são para os serviços de solicitação de autorização e de solicitação de *token*. No servidor de recurso o *endpoint* é fornecido para o serviço de solicitação de recursos.

Provedor de Serviço. Consiste em toda infraestrutura reunida e concentrada pela API RESTFul de serviços como serviços de autorização e, serviços de autenticação para fins de controle de acesso.

Serviço de Autorização. Serviço responsável por toda lógica de gerenciamento do *token* de acesso e validação de acessos aos recursos.

Serviço de Autenticação. Adição do gerenciamento de identidade para informar e verificar a credencial de identidade do cliente.

Políticas de Permissão. Persistem as políticas de acesso aos recursos baseado nos dados das credenciais de identidade, de autorização, de acesso dos clientes.

4.2.3 Protocolos do Perfil

Dado que uma das recomendações da especificação do ACE-OAuth é utilizar soluções leves e seguras diante da comunicação que é estabelecida entre as suas entidades para atender requisitos de ambientes restritos, optou-se pela seguinte configuração de protocolos e soluções como apresentado na Figura 51.

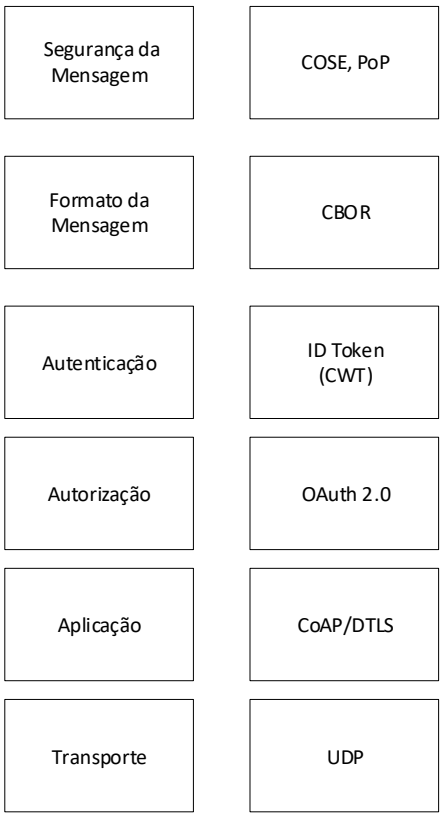

Figura 51 – Definição dos protocolos que fazem parte do perfil.

A especificação do ACE-OAuth recomenda que um perfil a ser criado contenha o CoAP como protocolo para transferência de dados na camada de aplicação, na Figura 51. Já em relação à estrutura das mensagens, é sugerido o CBOR, já que gera uma redução no tamanho nas trocas de mensagens e menor *overhead* no poder de processamento e de transmissão, Figura 52.

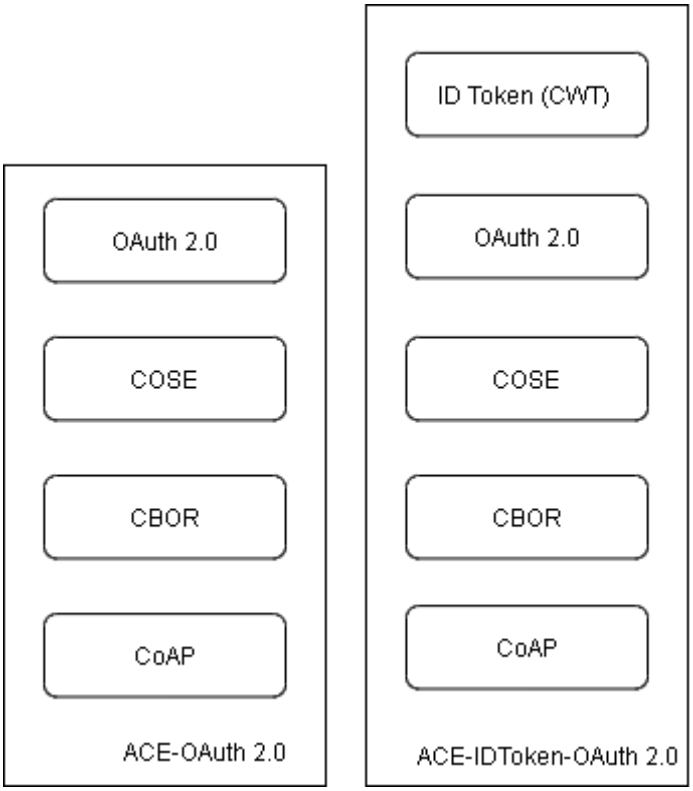

Figura 52 – Comparação entre a solução proposta ACE-OAuth e *framework* ACE-OAuth.

Para utilizar mecanismos de criptografia e assinatura digital sobre os atributos afirmativos (*claims*) ou conteúdo do *token,* por exemplo, é utilizado o objeto COSE. É importante destacar que os *claims* podem ser formatados pelo padrão CBOR Web *Token* (CWT) em que um conjunto de atributos afirmativos é serializado pelo padrão CBOR. Dessa maneira, na solução proposta o ID Token foi serializado pelo de acordo com padrão CWT. Já as características utilizadas no objeto COSE para criptografar a credencial (ID Token ou *token)*, na solução proposta, foram por meio do rótulo ("`COSE_Encryption0`"), a fim de garantir a confidencialidade na comunicação entre as partes

4.2.4 Perfil de Comunicação e Segurança entre as Partes

Na Figura 53 é mostrado o perfil de comunicação seguro CoAP/DTLS para o ACE-OAuth adotado para a solução proposta neste trabalho.

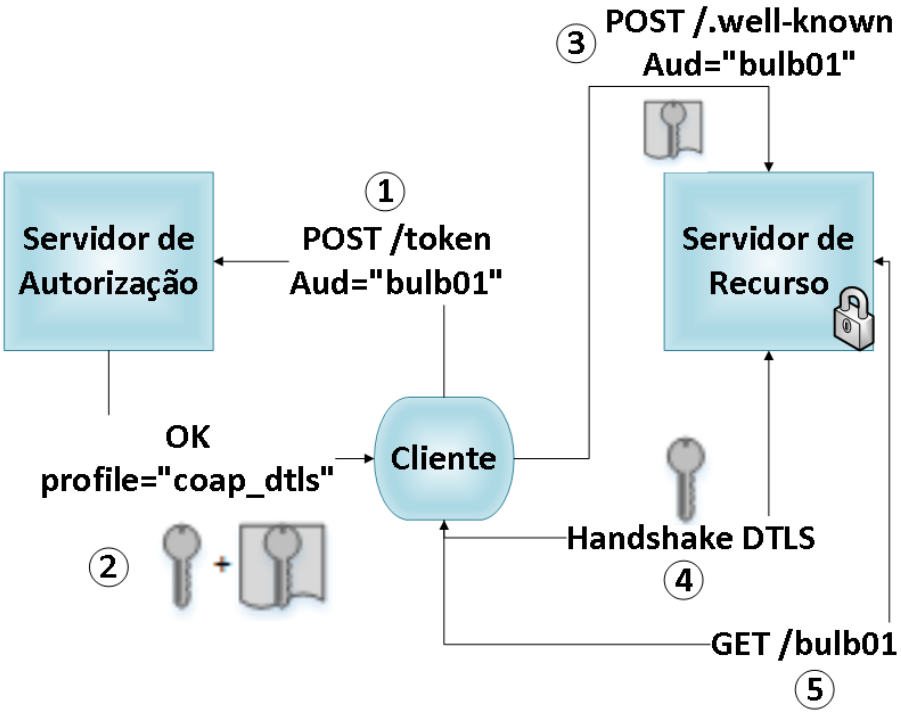

Figura 53 – Etapas da comunicação segura no perfil CoAP/DTLS para ACE-OAuth.
Fonte: (BELTRAN; SKARMETA, 2017).

A Figura 53 mostra as etapas na comunicação segura do perfil CoAP/DTLS com a prova de posse simétrica:

1) Inicialmente, o dispositivo cliente solicita autorização ao servidor de autorização para acessar o recurso;
2) O servidor de autorização responde com uma chave simétrica e um *token* PoP que contém essa chave para o cliente;
3) O cliente envia o *token* recebido ao servidor de recursos e depois inicia um *handshake* DTLS, com a chave simétrica que foi pré-compartilhada entre e o cliente e servidor de recurso;
4) O servidor de recurso responde o estabelecimento do canal de segurança;
5) Uma vez que o canal de segurança esteja estabelecido, o cliente solicita o recurso ao servidor de recursos. Para isso, o cliente apresenta a prova de posse e o servidor de recurso a descriptografa utilizando a chave simétrica que foi pré-compartilhada entre ele e o servidor de autorização.

No perfil de comunicação e segurança CoAP/DTLS, o servidor de autorização, além de ter a função de autorização, emite o ID Token e confirma o seu uso. Dessa forma, no cenário criado, o servidor de autorização pode ser reconhecido como

servidor de autorização assim como um provedor de veracidade.

Desse modo, as etapas percorridas no fluxo proposto pelo perfil da solução proposta são: registro das credenciais, solicitação de autorização, solicitação do *token*, solicitação do recurso.

Registro das credenciais. O processo de registro e emissão da credencial de identificação pode ser feito estaticamente ou dinamicamente como razão para ter uma relação de confiança entre o cliente e com o servidor de autorização (BELTRAN; SKARMETA, 2017). Para tratar dessa relação de confiança, propostas como já utilizadas para Web como Protocolo de Registro de Cliente Dinâmico (*Dynamic Client Registration Protocol)*[50] por RICHER et al.(2015) poderiam ser estendidas, mas sua aplicabilidade precisa ser estudada para ambiente de IoT (BELTRAN; SKARMETA, 2017). Segundo (HOVSMITH, 2017) a credencial no modo estático não é muito seguro em aplicações, já que pode ser alvo de engenharia reversa e o atacante pode descobrir os dados das credenciais e depois reusá-las em uma aplicação genuína.

De qualquer maneira, o registro estático pode ser realizado de duas formas. Na primeira, o dono do recurso solicita as credenciais de identificação ao servidor de autorização, Figura 54, o qual as emite e as envia para o dono do recurso, Figura 55. Em seguida, na Figura 56, o dono do recurso aplica as credenciais na aplicação cliente de terceiros. Na solução adotada para reforçar a autenticação das aplicações, a credencial de identificação é composta pela credencial (identificador, senha) e mais o ID Token com a finalidade de reduzir a chance do atacante se passar por uma aplicação cliente válida para obter uma sessão do usuário com intenção de modificar ou destruir os recursos.

Por meio do ID Token, torna-se mais difícil o atacante supor ou reusar valores atribuídos aos parâmetros de identidade utilizados nas requisições para criar uma solicitação válida em qualquer requisição, pois ID *token* usado numa requisição atual precisa ser emitido ou atualizado numa requisição anterior pelo servidor de autorização.

Dessa forma ID Token é uma opção para substituir o uso da credencial de identificação (identificador/segredo) nas requisições, já que dados da identidade são emitidos e atualizados a cada resposta da entidade de autorização. Além disso, o ID

[50] OAuth 2.0 *Dynamic Client Registration Protocol*: https://tools.ietf.org/html/rfc7591

Token oferece mecanismos alternativos de autenticação ao cliente (CAMPBELL et al., 2015).

Entretanto, como o objetivo deste trabalho é reforçar, sobretudo, a proteção da comunicação entre o cliente e o servidor de autorização, foram desenvolvidas duas possíveis soluções dinâmicas de registro e emissão da credencial de identificação: Registro de Dinâmico da Credencial e Registro Dinâmico da Credencial com Solicitação de Concessão de Acesso, no Apêndice B. Elas podem ser adotadas na etapa de registro das credenciais e emissão da credencial de identificação na evolução deste trabalho. A solução dinâmica de registro da credencial de identificação é adequada quando uma API tem uma relação mutável com diversos clientes e por meio desses, contém várias instâncias de aplicações cliente (RICHER; SANSO, 2017). No registro estático, se atribuí um único identificador para uma cópia do software do cliente. No entanto, para clientes nativos em plataformas móveis, pode haver múltiplas instâncias do mesmo software, e cada instância pode ter seu próprio identificador e segredo, oferecendo escalabilidade à emissão da credencial (RICHER; SANSO, 2017).

Dentre as vantagens de se utilizar solução dinâmica de registro e emissão da credencial de identificação em contraste das soluções estáticas são: maior garantia de confiança, escalabilidade e praticidade. Em relação à garantia de segurança, a credencial é gerada aleatoriamente e se utiliza de dados contextuais, conhecidos como metadados. Quanto à escalabilidade, na solução dinâmica, o cliente interage com mais de uma autoridade de autorização. Em relação à praticidade, o registro dinâmico das credenciais é gerado em tempo de execução e automatizado pelo servidor de autorização ou pela própria aplicação.

Por outro lado, as desvantagens de utilizar solução dinâmica no registro e emissão da credencial de identificação em relação às soluções estáticas são: há possibilidade de monitoração e de interceptação, se as credenciais não forem trafegadas em canal seguro ou criptografadas; há um esforço para vincular parâmetros uns com outros para criar regras e políticas para fins de controle de acesso; na solução dinâmica há uma quantidade bem superior de metadados utilizados do que na solução estática e, por conta disso, pode impactar no tamanho das mensagens.

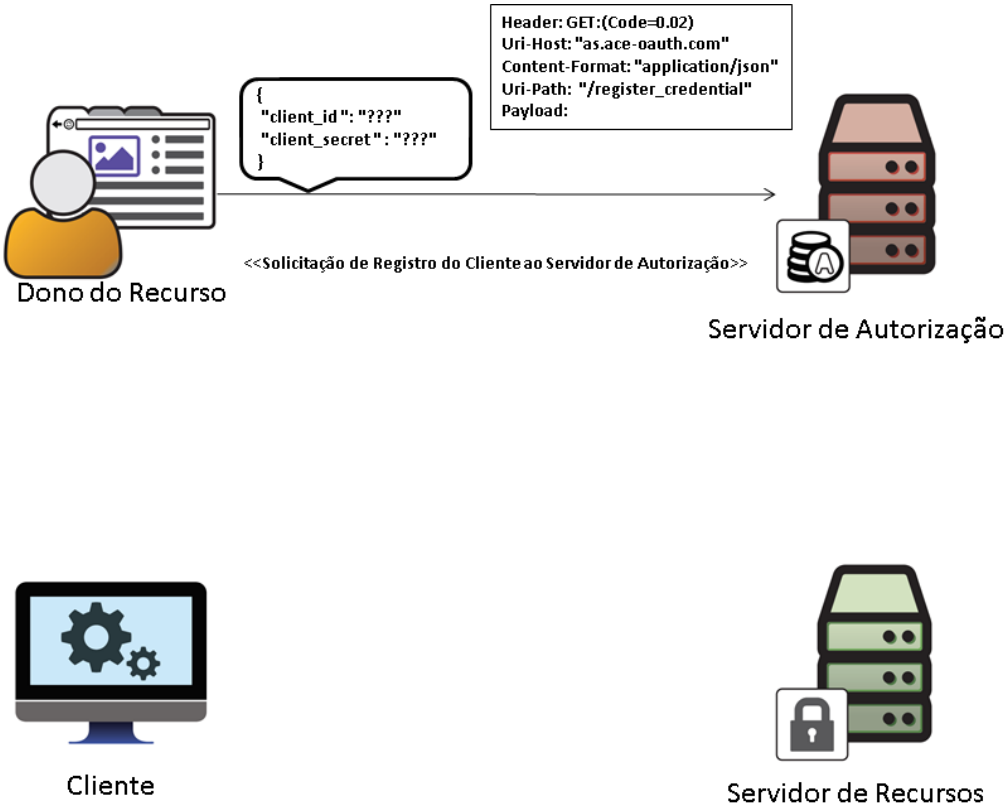

Figura 54 – Solicitação do registro do cliente ao servidor de autorização.

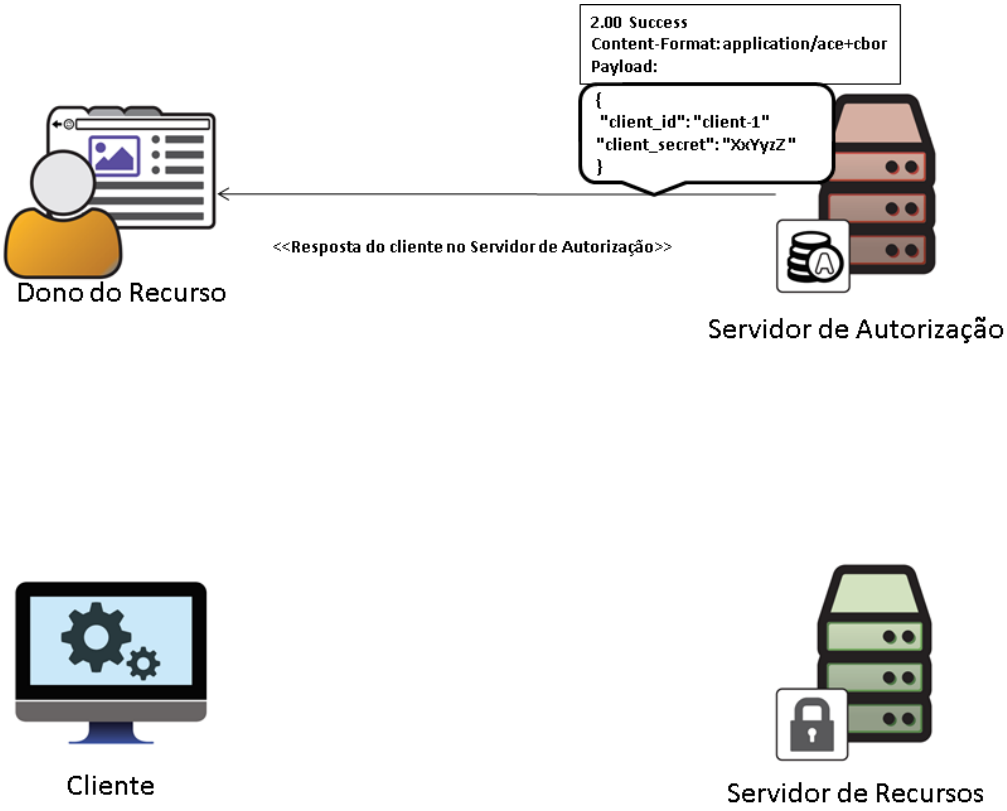

Figura 55 – Resposta da solicitação indireta do registro da identificação do cliente.

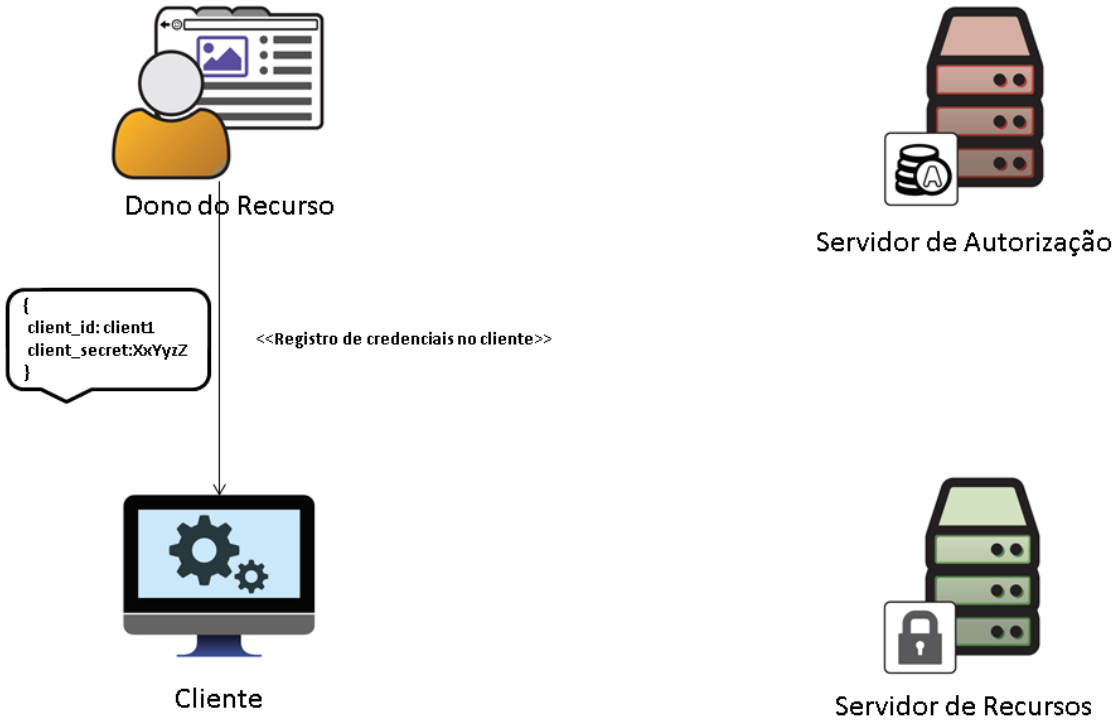

Figura 56 – Dono do recurso fornece a credencial de identificação para aplicação cliente de terceiros.

Na segunda forma, no lado cliente, a aplicação de terceiros solicita diretamente ao servidor de autorização a credencial de identificação, Figura 57, o qual, em seguida, emite as credenciais de identificação e responde para a aplicação cliente de terceiros, Figura 58.

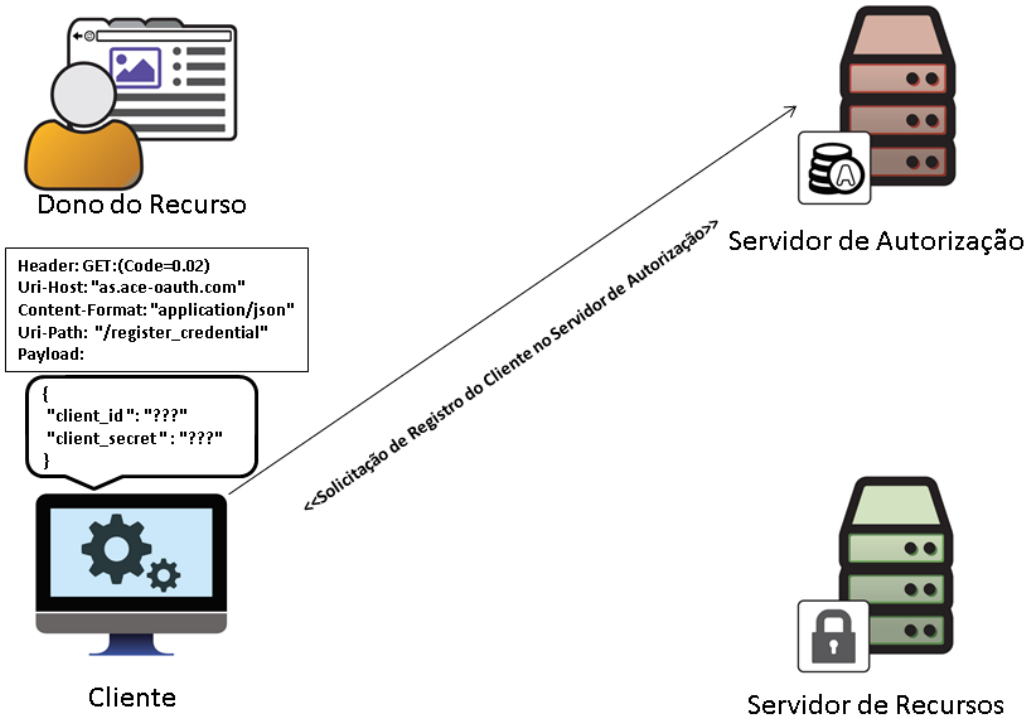

Figura 57 – Solicitação direta do registro de identificação do cliente.

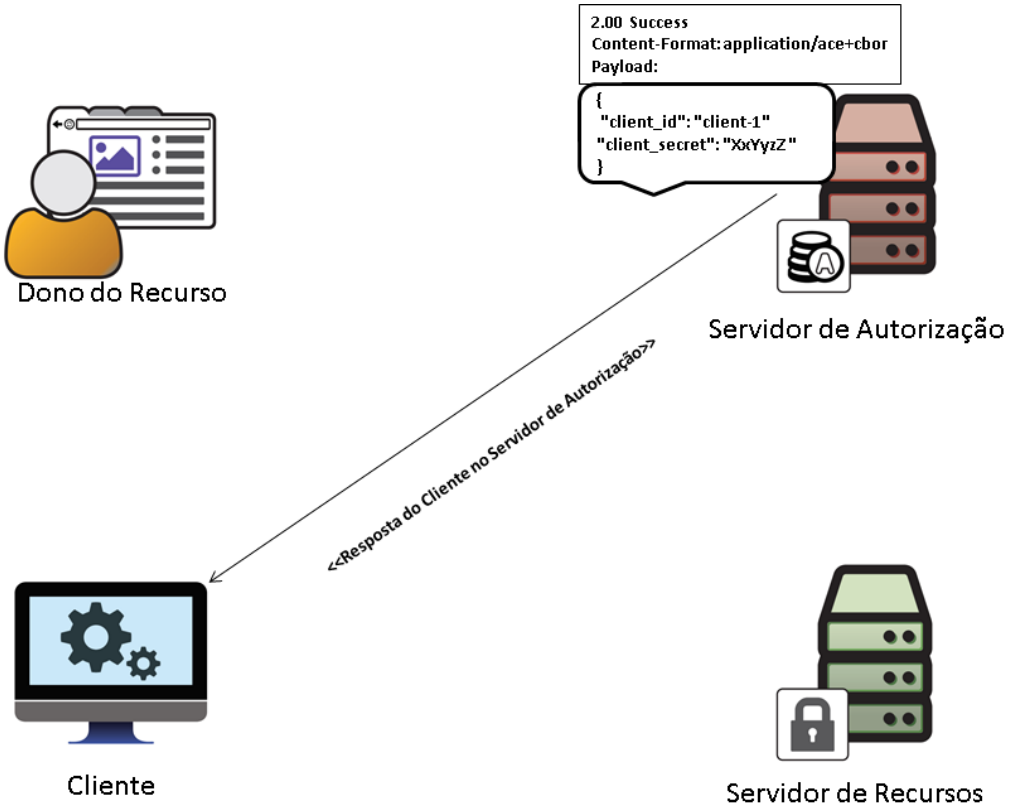

Figura 58 – Resposta a solicitação direta do registro de identificação do cliente.

Após o registro de identificação da aplicação cliente, é realizado um comparativo entre o fluxo de concessão do código de autorização normal e o fluxo proposto com o reforço do desafio-resposta e da estrutura ID Token. E por fim, é realizada uma consideração de segurança sobre as requisições e repostas no fluxo de comunicação entre as partes. Foi assumido para o registro da identificação, o modo direto para ambos os fluxos, como apresentado anteriormente. Vale destacar que a aplicação de terceiros foi tratada genericamente como cliente.

Requisição de autorização do fluxo original. Na requisição de autorização, o cliente realiza um POST para *endpoint* (“`/authorization`”) do servidor de autorização, Figura 59. O cliente passa os atributos de identificação (“`id`”), escopo (“`scope`”), e URI de redirecionamento (“`redirect_uri`”) e estado (“`state`”) na requisição para o servidor. Vale destacar que o atributo estado é utilizado na tentativa confirmar a origem da requisição. Dessa forma o servidor de autorização armazena o valor do atributo estado para ser comparado na próxima requisição. Na

requisição de autorização no fluxo de concessão código de autorização necessita do atributo tipo de reposta (“`response_type`”) com valor código (“`code`”) e o tipo de concessão é especificado no atributo (“`grant_type`”) com o valor (“`authorization_code`”). O cliente usa o atributo identificador (“`cliente_id`”) e o escopo (“`scope`”) para indicar qual a classe de recursos ele quer acessar.

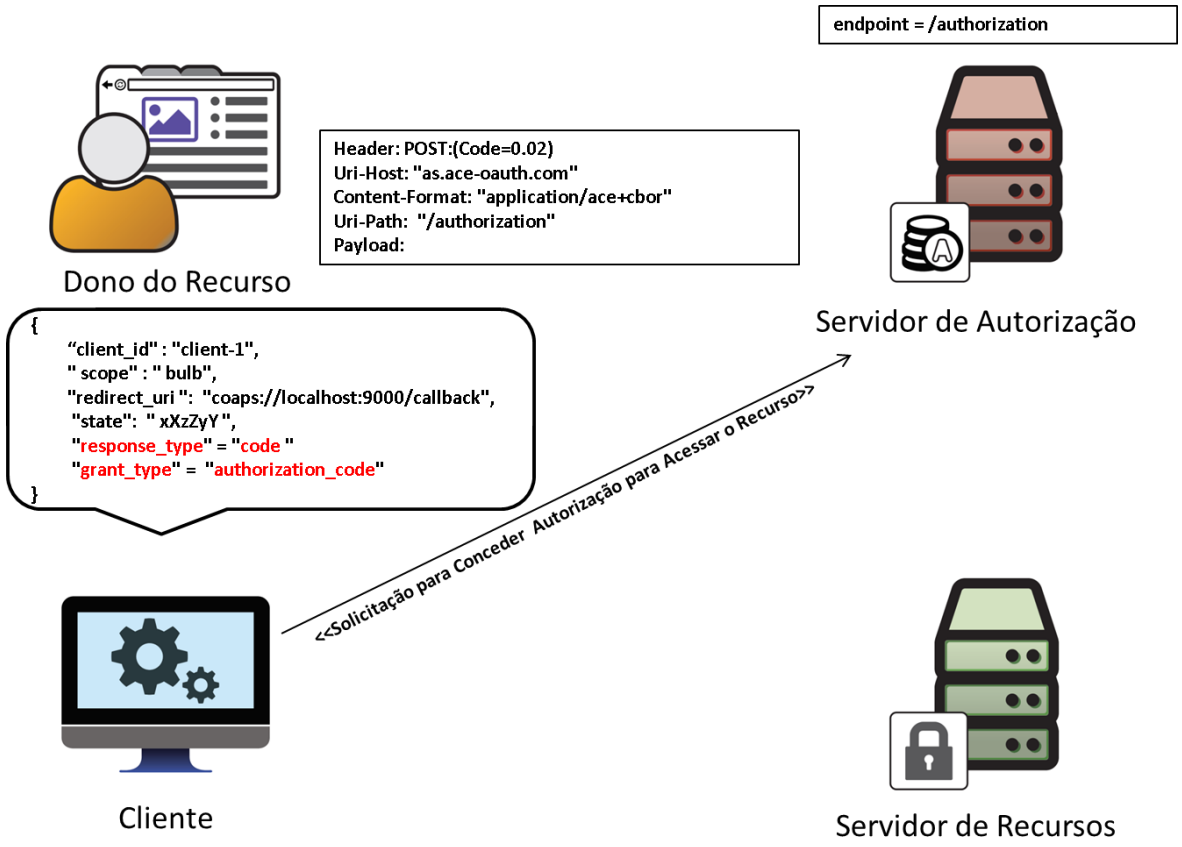

Figura 59 – Requisição de autorização no fluxo normal.

Assim que o servidor de autorização recebe a requisição de autorização do cliente, ele processa e depois encaminha a solicitação de autorização para o dono de recurso, Figura 60. Além disso, o servidor de autorização notifica o dono do recurso da solicitação de autorização do cliente.

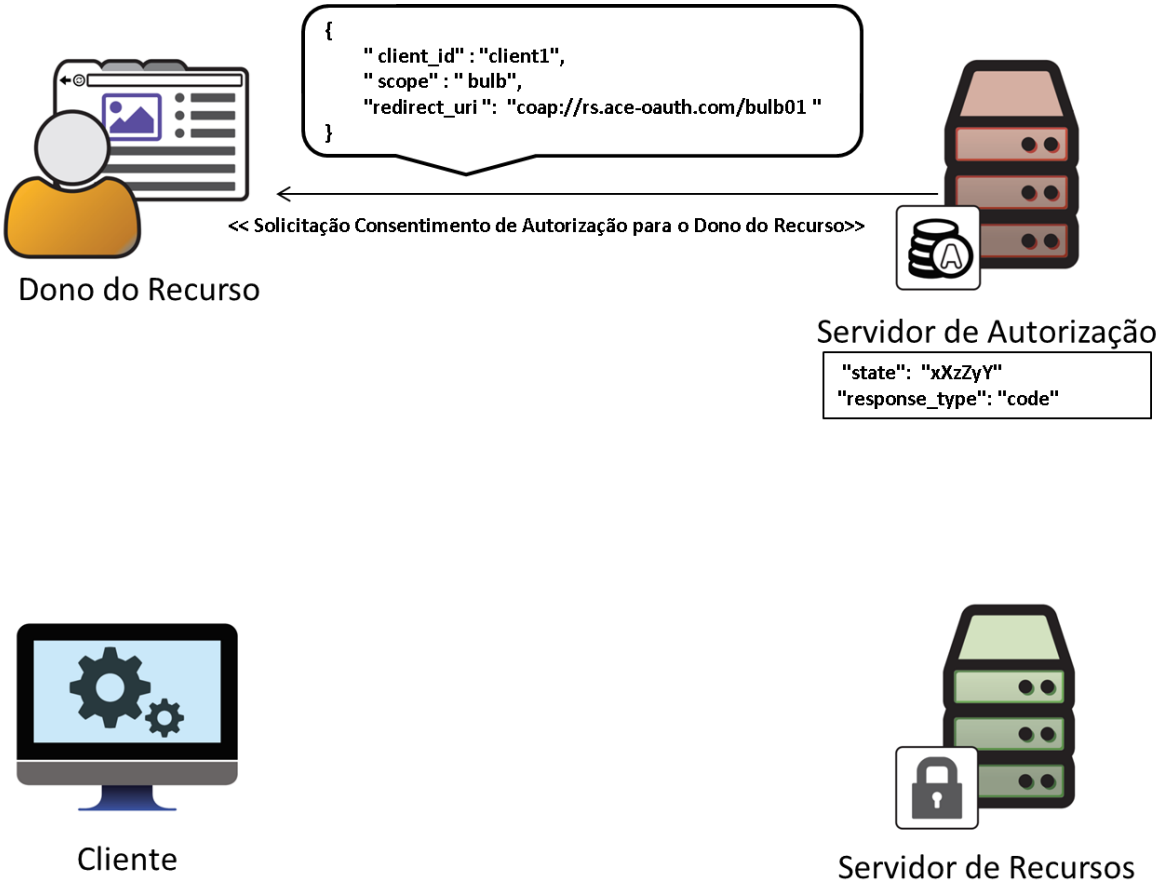

Figura 60 – Solicitação de consentimento de autorização para o dono do recurso.

Em seguida, o dono de recurso toma a decisão de aceitar ou negar a solicitação de autorização para o cliente. Caso o dono do recurso tome a decisão de aprovar a autorização, ele envia uma requisição POST para o *endpoint* (“`/approve`”) do servidor de autorização, Figura 61.

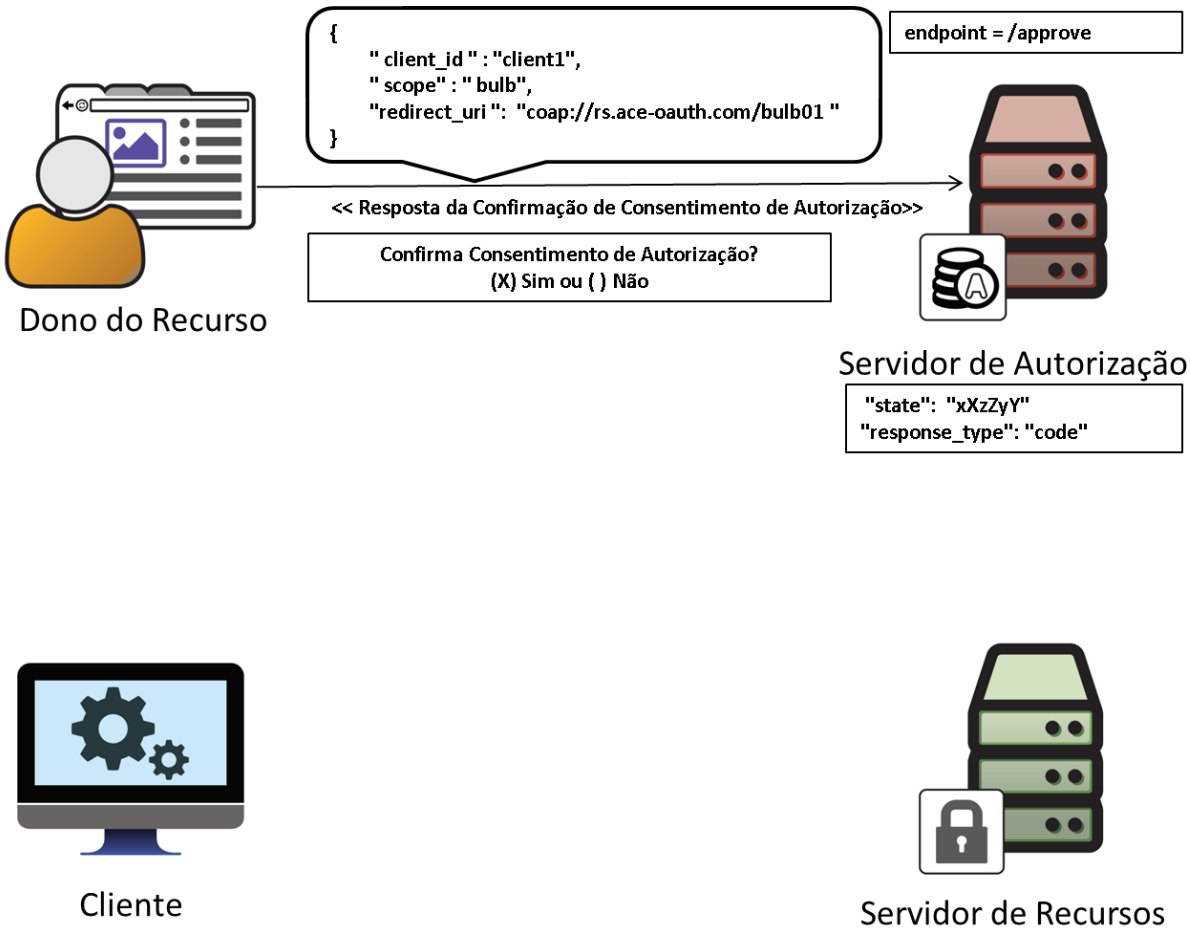

Figura 61 – Resposta do consentimento de autorização do dono do recurso.

Requisição de autorização da proposta. Para a etapa de requisição de autorização, foi reaproveitado um dos tipos de resposta já existente nos fluxos[51] utilizados no OpenID Connect[52] (KAWASAKI, 2017) para realizar autenticação das requisições do cliente por meio da estrutura ID Token. Para a proposta, a estrutura do ID Token serve como identificação do cliente para suas requisições, Figura 62. Diferentemente de um segredo do cliente ou um valor aleatório, o ID Token contém o atributo que corresponde ao tempo de uso e mostra como o cliente está autorizado a acessar o recurso. No fluxo proposto, o ID Token é emitido antecipadamente pelo servidor de autorização e deve ser apresentado para cada nova solicitação feita pelo cliente ao servidor de autorização para evitar a personificação do cliente. Todavia, para impedir a captura do ID Token por algum atacante que queira se fazer passar

[51] Todos os diagramas dos Fluxo OpenID Connect: https://medium.com/@darutk/diagrams-of-all-the-openid-connect-flows-6968e3990660
[52] Tipos de Resposta do OAuth 2.0: https://openid.net/specs/oauth-v2-multiple-response-types-1_0.html#Combinations

pelo cliente, o ID Token deve trafegar em canal seguro ou ser criptografado com objeto COSE.

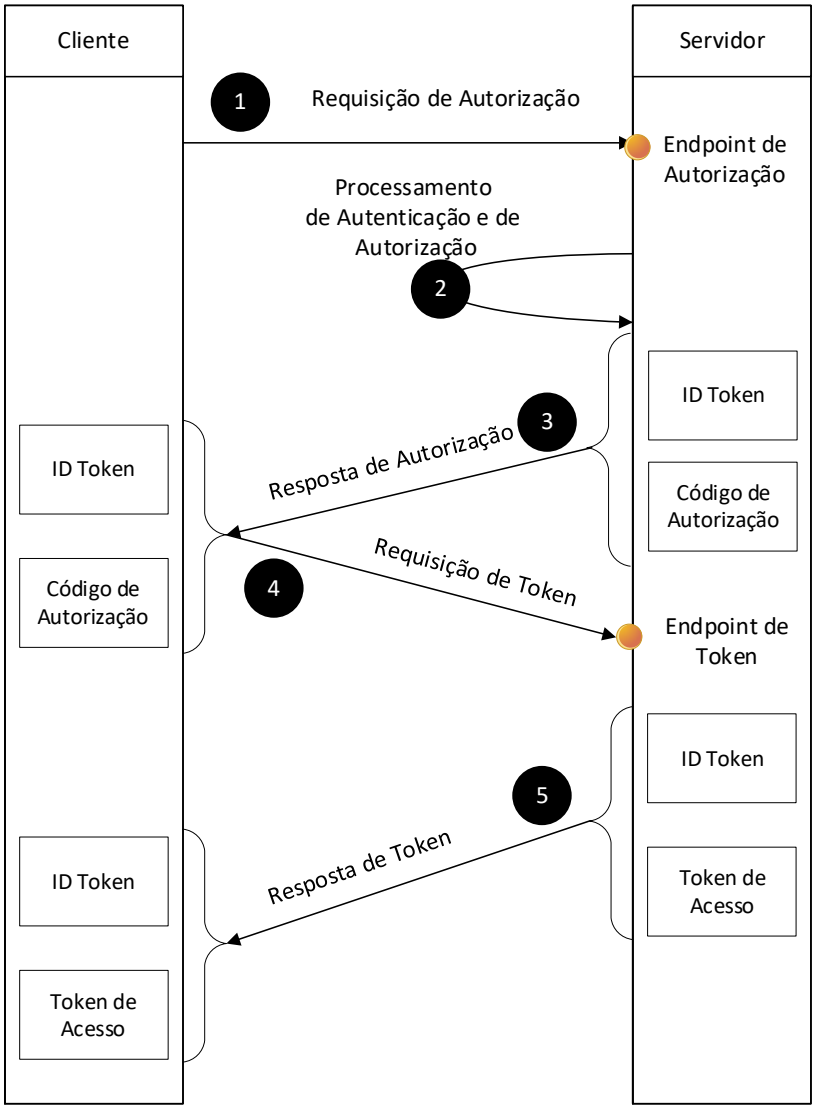

Figura 62 – Tipo de resposta adotado no fluxo do OpenID Connect.
Fonte: (KAWASAKI, 2017)

De acordo com a Figura 62, é descrita cada uma de suas etapas:

1) As requisições de autenticação ocorrem pela apresentação da credencial de identidade (identificador) e requisições de autorização por meio da credencial de autorização (código de autorização) as quais são enviadas pelo cliente para o servidor de autorização;
2) O servidor processa com a finalidade de realizar a autenticação e autorização da aplicação cliente;
3) Se a requisição for aceita pelo servidor de autorização ele envia o código de autorização com o ID Token;

4) O cliente utiliza código de autorização e o ID Token recebidos na etapa anterior para realizar a requisição de *token*;
5) Se a requisição for aceita pelo servidor de autorização, o servidor de autorização envia o ID Token junto com o *token* de acesso. Por fim, o cliente solicita o acesso ao recurso, para isso o cliente adiciona na requisição para o servidor de recurso o ID Token e *token* de acesso recebidos como resposta na da requisição do *token*.

Nesta solução, para solicitar a credencial de autorização, o cliente envia uma mensagem POST para o *endpoint* ("`/authorization`") no servidor de autorização, Figura 63, fazendo o uso do perfil de comunicação e de segurança CoAP/DTLS (Subseção 4.2.4).

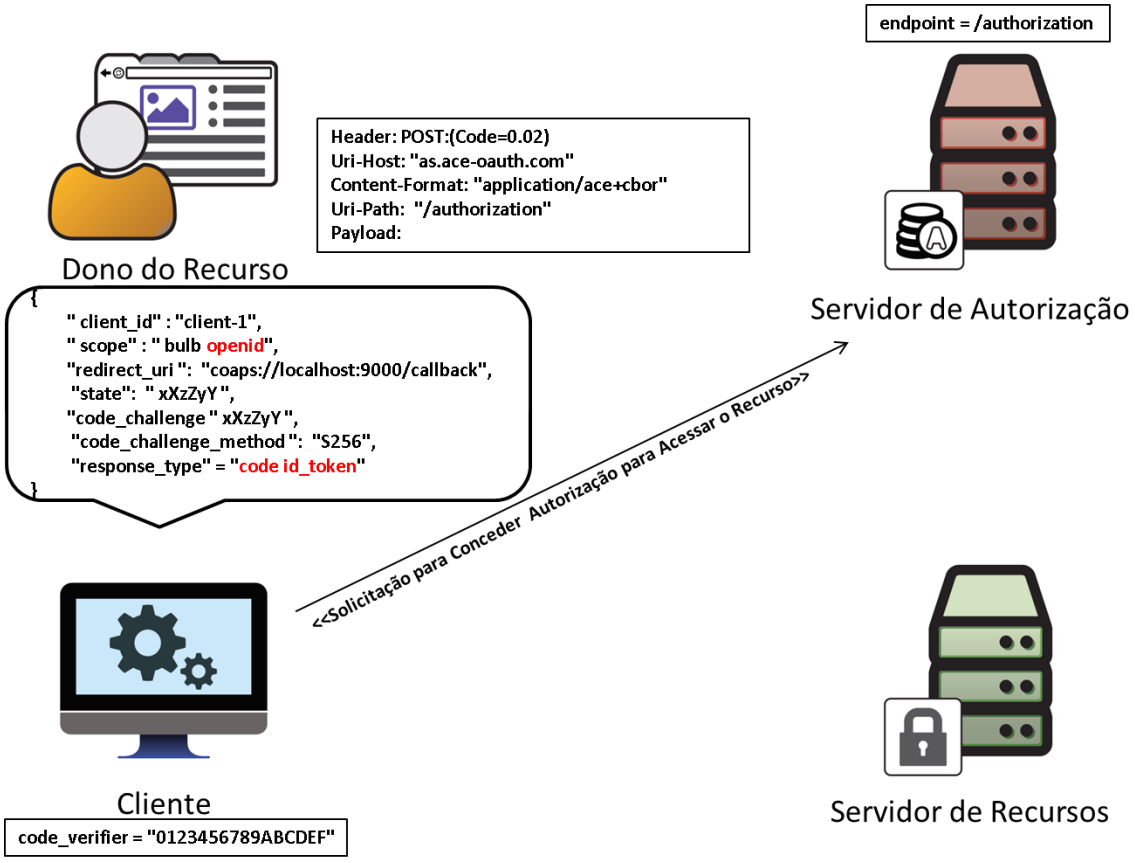

Figura 63 – Solicitação para conceder autorização.

Resposta de autorização do fluxo original. Após o servidor de autorização receber a confirmação afirmativa do dono do recurso, o servidor de autorização emite um código de autorização por meio do atributo código ("`code`") e a resposta de requisição de autorização é enviada para a aplicação cliente de terceiros, Figura 64.

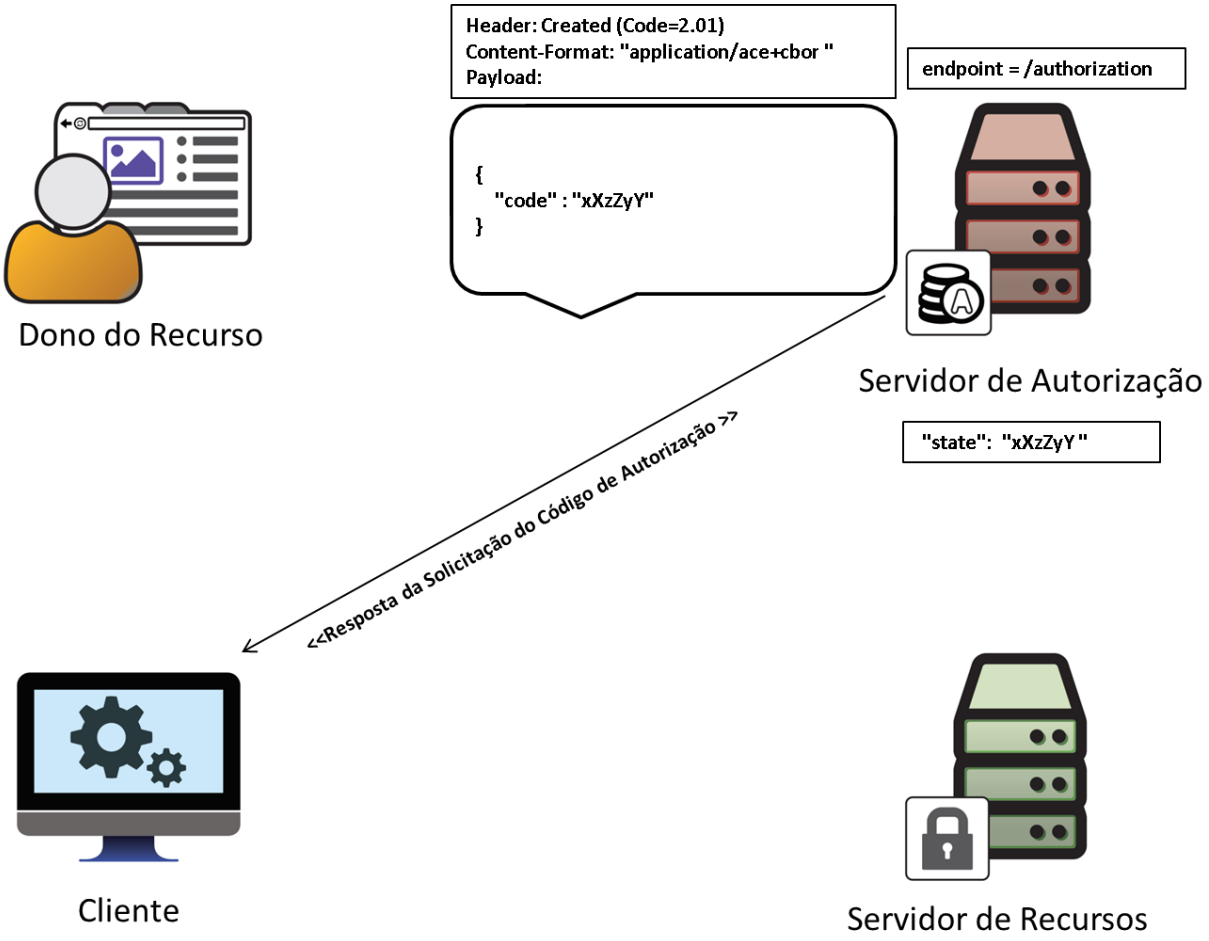

Figura 64 – Resposta de autorização no fluxo normal.

Resposta de autorização da proposta. Antes do servidor de autorização enviar uma resposta para o cliente, o servidor precisa verificar se o cliente tem permissão para emitir o *token* de acesso. Para isso, o servidor de autorização pode encaminhar solicitação de autorização para o dono do recurso para permitir ou revogar a solicitação. Entretanto, como o fluxo de veracidade foi adotado na solução proposta, o servidor de autorização emite a credencial de autorização sem a necessidade do consentimento do dono do recurso, Figura 65. Dessa maneira, o servidor de autorização toma a decisão do consentimento de autorização, já que assume uma relação de confiança entre o cliente confidencial e o servidor de autorização por meio do registro credencial anterior, sendo pré-atribuído o consentimento do cliente.

A Figura 65 mostra quando o servidor concede autorização para a solicitação do *token* de acesso. O servidor de autorização emite um ID Token referente à requisição de autorização junto com o código de autorização.

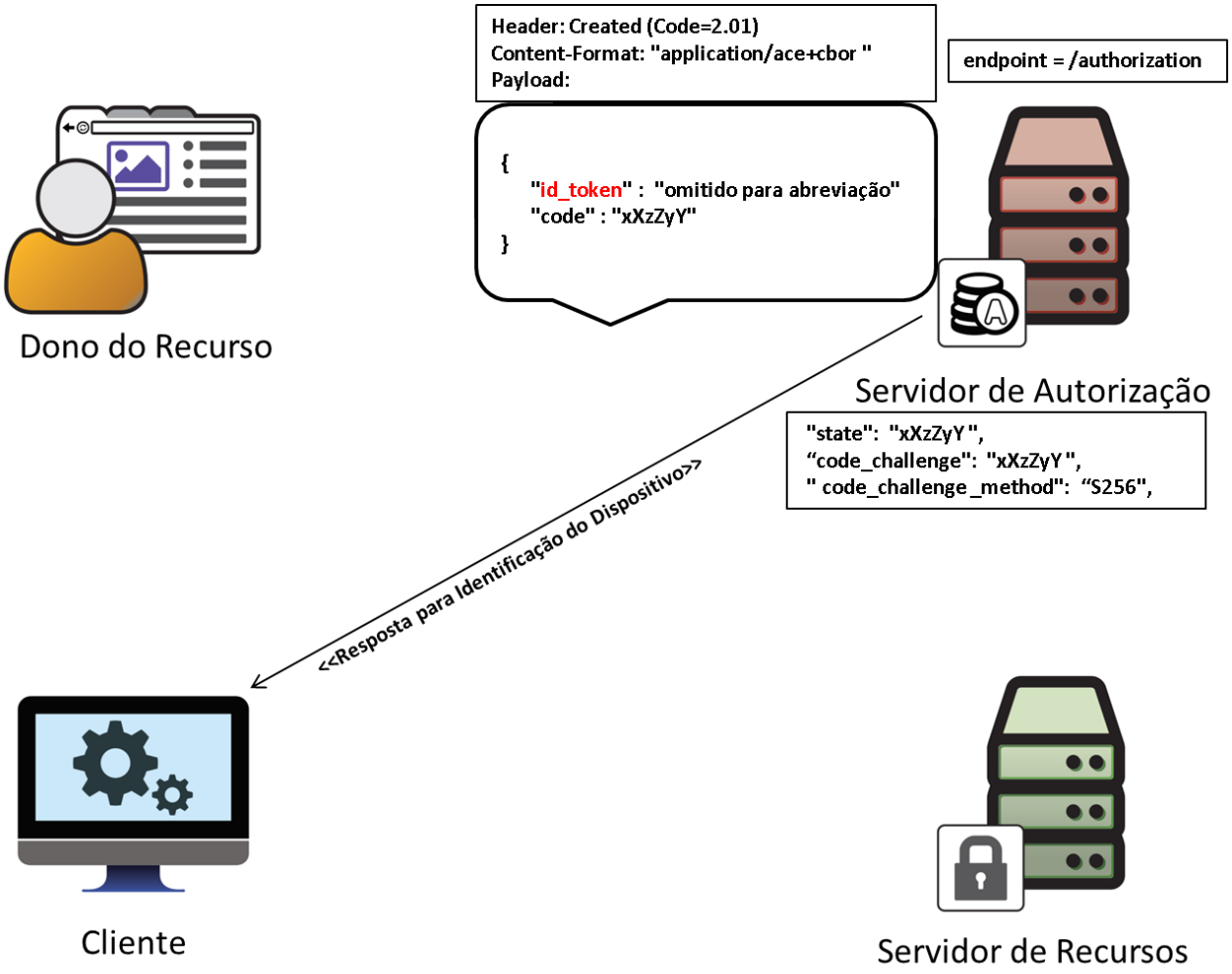

Figura 65 – Resposta para concessão do código e do *token* de identificação.

Requisição do *token* do fluxo original. Depois de obtida a credencial de acesso (código de autorização), o cliente realiza uma requisição POST para o *endpoint* (“`/token`”) do servidor de autorização para solicitar a credencial de acesso (*token*), Figura 66. O cliente realiza a requisição do *token* em que são passados os atributos de identificação (“`client_id`”), URI de redirecionamento (“`redirect_uri`”), estado (“`state`”), escopo (“`scope`”), código de autorização (“`code`”) e o tipo de resposta (“`response_type`”) como prova de posse, que consiste de chave criptográfica vinculada ao *token*. O servidor de autorização utiliza o atributo estado (“`state`”) recebido na requisição de *token* e compara com o estado armazenado na requisição de autorização a fim de confirmar a origem**.**

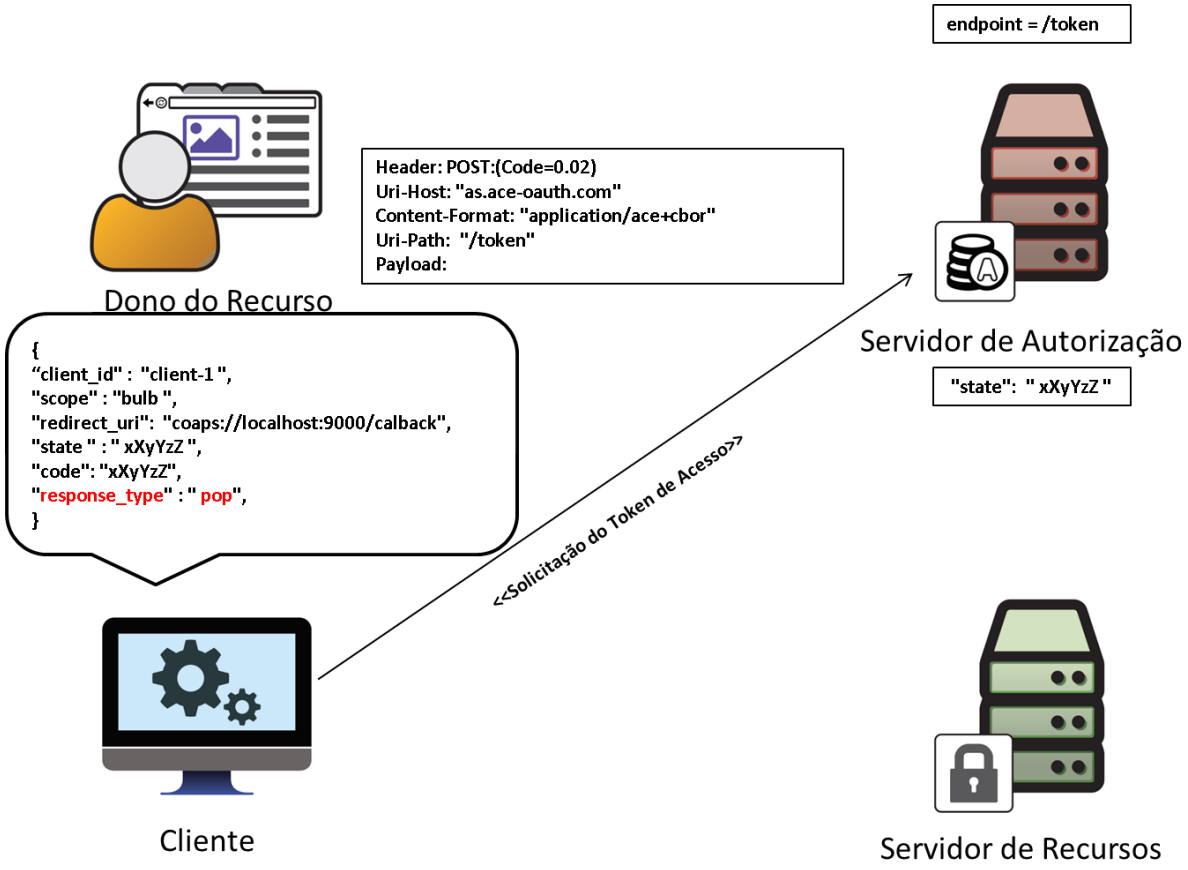

Figura 66 – Solicitação do token de acesso no fluxo normal.

Requisição do *token* da proposta. Na Figura 67 é realizada a requisição do *token* de acesso em que o cliente envia uma requisição POST sobre o *endpoint* de *token* (“*/token*”). Dessa vez, passa os seguintes parâmetros (“`client_id`”), (“`redirect_uri`”) e (“`state`”) os mesmos utilizados na requisição de autorização Junto a isso, são incluídos o ID Token e o código que são obtidos na resposta da requisição de autorização. O valor do parâmetro (“`code_verifier`”) contém a chave que gerou o desafio na etapa da requisição de autorização no lado cliente. Quando o parâmetro (“`code_verifier`”) é enviado na requisição para o servidor de autorização, o servidor de autorização tenta gerar o mesmo desafio recebido com a chave recebida e o valor resultante é comparado ao desafio recebido na requisição de autorização para confirmar se a origem da requisição *token* é a mesma da requisição de autorização.

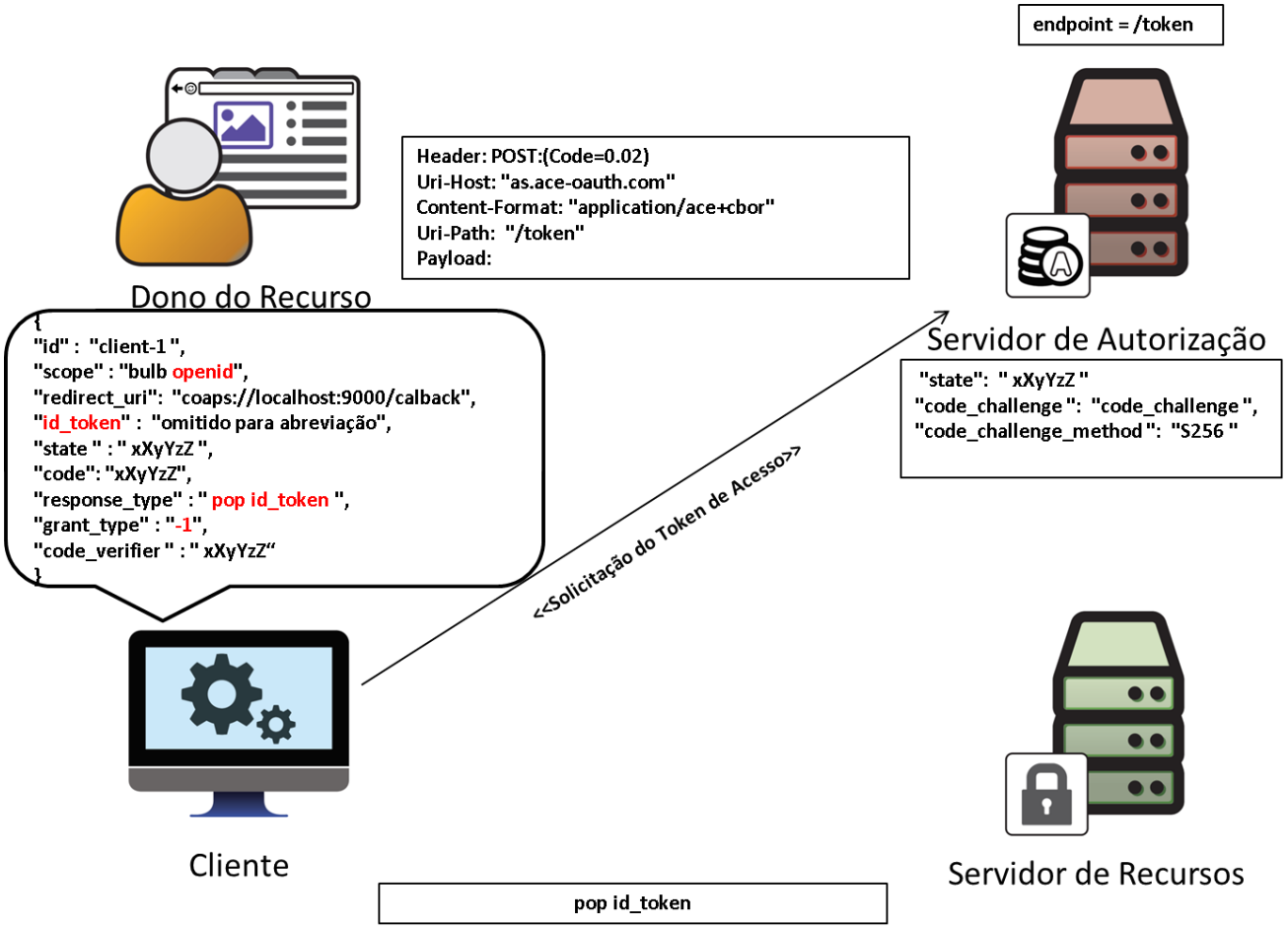

Figura 67 – Solicitação de autorização para geração do *token* de acesso.

Resposta do *token* do fluxo original. Caso a verificação dos atributos esteja em conformidade, o servidor de autorização emite uma resposta à requisição de *token* enviada pela aplicação de cliente de terceiros, Figura 68. O servidor de autorização envia em sua resposta os atributos perfil (“`profile`”), que é a configuração utilizada entre o servidor de autorização e o servidor de recurso, tempo de expiração (“`expires_in`”) do *token* e a confirmação (“`cnf`”). O atributo de confirmação contém a prova de posse (PoP), um objeto criptografado COSE que esconde o conteúdo do *token*.

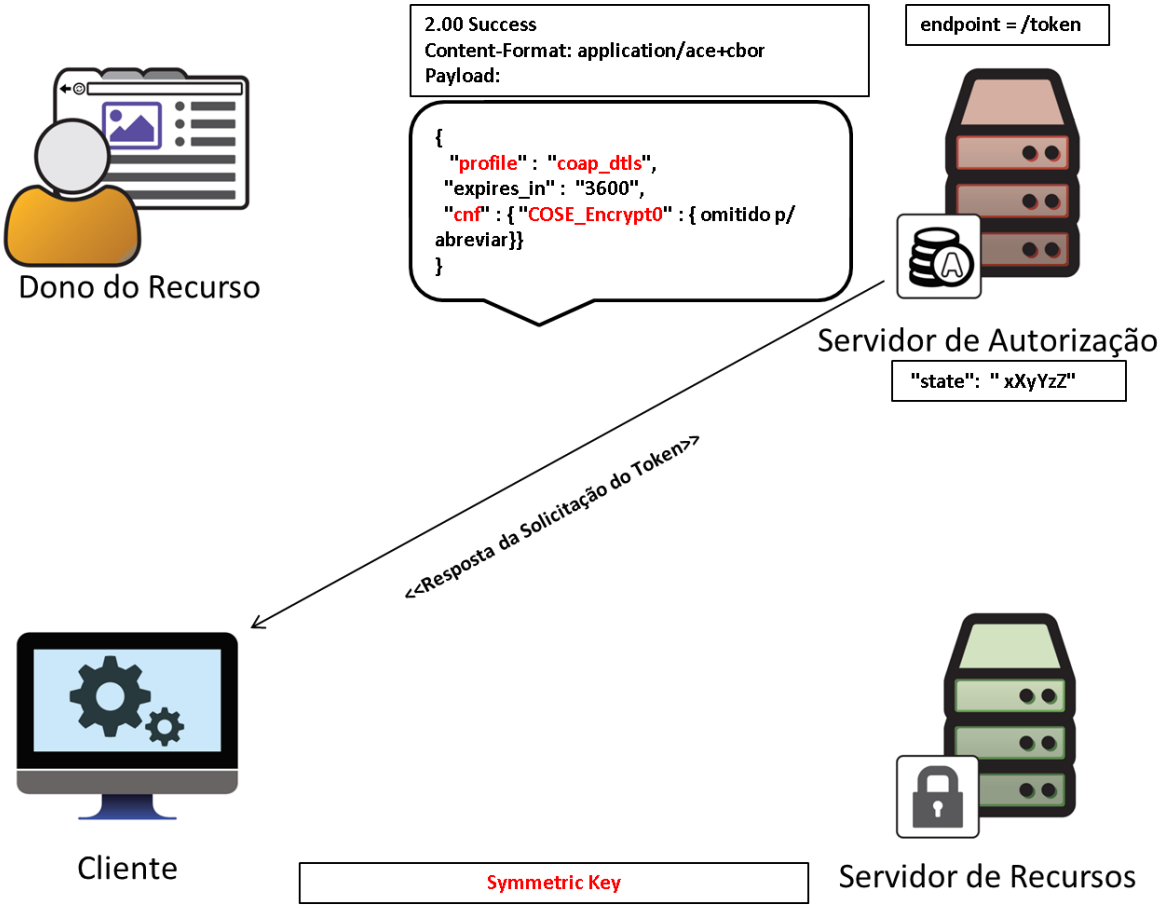

Figura 68 – Resposta da solicitação do token no fluxo normal.

Resposta do *token* da proposta. Na Figura 69, assim que o servidor de autorização verifica os parâmetros recebidos na requisição do *token* de acesso ele envia uma resposta com a emissão da prova de posse por meio do parâmetro de confirmação (“`cnf`”), criptografado com um objeto COSE e o parâmetro de identificação do *token* (“`id_token`”) no corpo da mensagem de resposta.

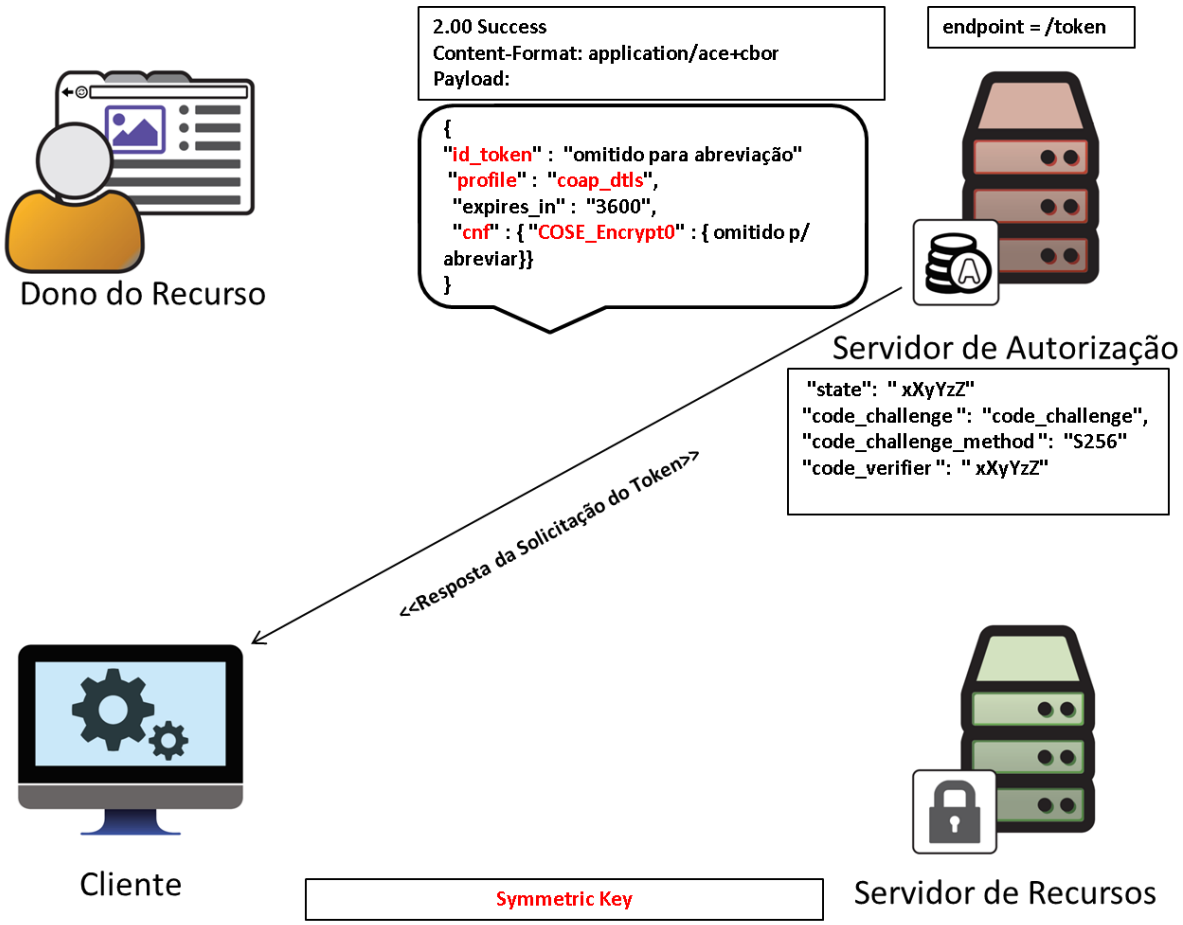

Figura 69 – Resposta da solicitação do *token* de acesso.

Outros parâmetros estão contidos na resposta do servidor de autorização, como o parâmetro perfil (“`profile`”) que demonstra ao cliente que a comunicação com o servidor do recurso deverá ser realizada por meio do perfil CoAP/DTLS. Já o parâmetro de expiração do *token* de acesso (“`expires_in`”) corresponde ao valor, em segundos, para usar o *token* de acesso no servidor de recursos.

.

Requisição do recurso do fluxo original. Após o cliente receber a prova de posse (PoP), que serve como credencial de acesso para a requisição de recurso, a aplicação cliente envia uma requisição GET para recuperar o estado do recurso ou uma requisição POST para atualizar o estado do recurso no *endpoint* (“`/.well-known`”) do servidor de recurso, Figura 70. A aplicação cliente de terceiros, na requisição do recurso, adiciona os atributos de identificação (“`client_id`”), escopo (“`scope`”), redirecionamento de uri (“`redirect_uri`”), perfil (“`profile`”), tempo de expiração (“`expires_in`”) e a confirmação (“`cnf`”).

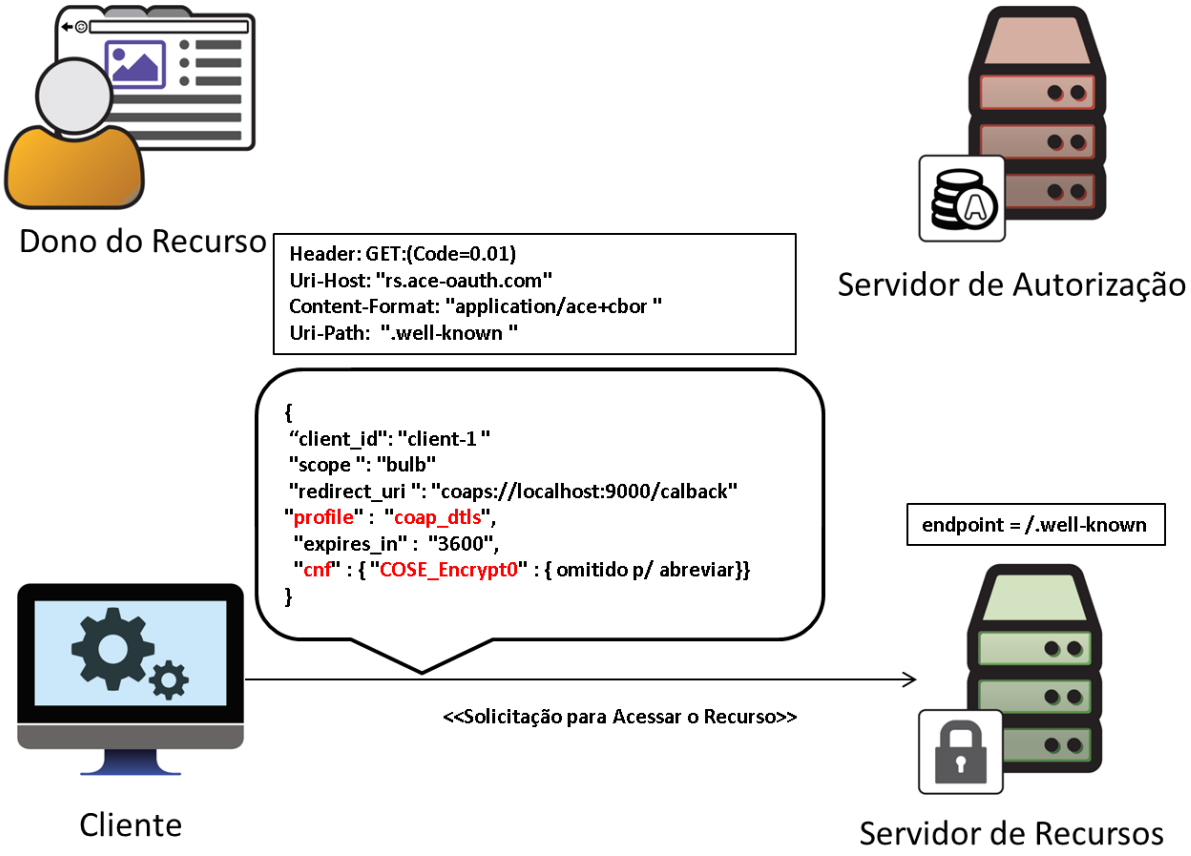

Figura 70 – Requisição do recurso no fluxo normal.

Requisição do recurso da proposta. Em detenção da prova de posse e do ID Token, o cliente envia uma requisição GET para o servidor do recurso para recuperar o estado do recurso ou objeto, por exemplo, para recuperar o estado da lâmpada inteligente, como mostra a Figura 71. Para realizar essa ação o cliente passa os seguintes parâmetros (“`client_id`”), (“`redirect_uri`”) e (“`scope`”) na requisição de recurso, os mesmos utilizados na requisição de autorização.

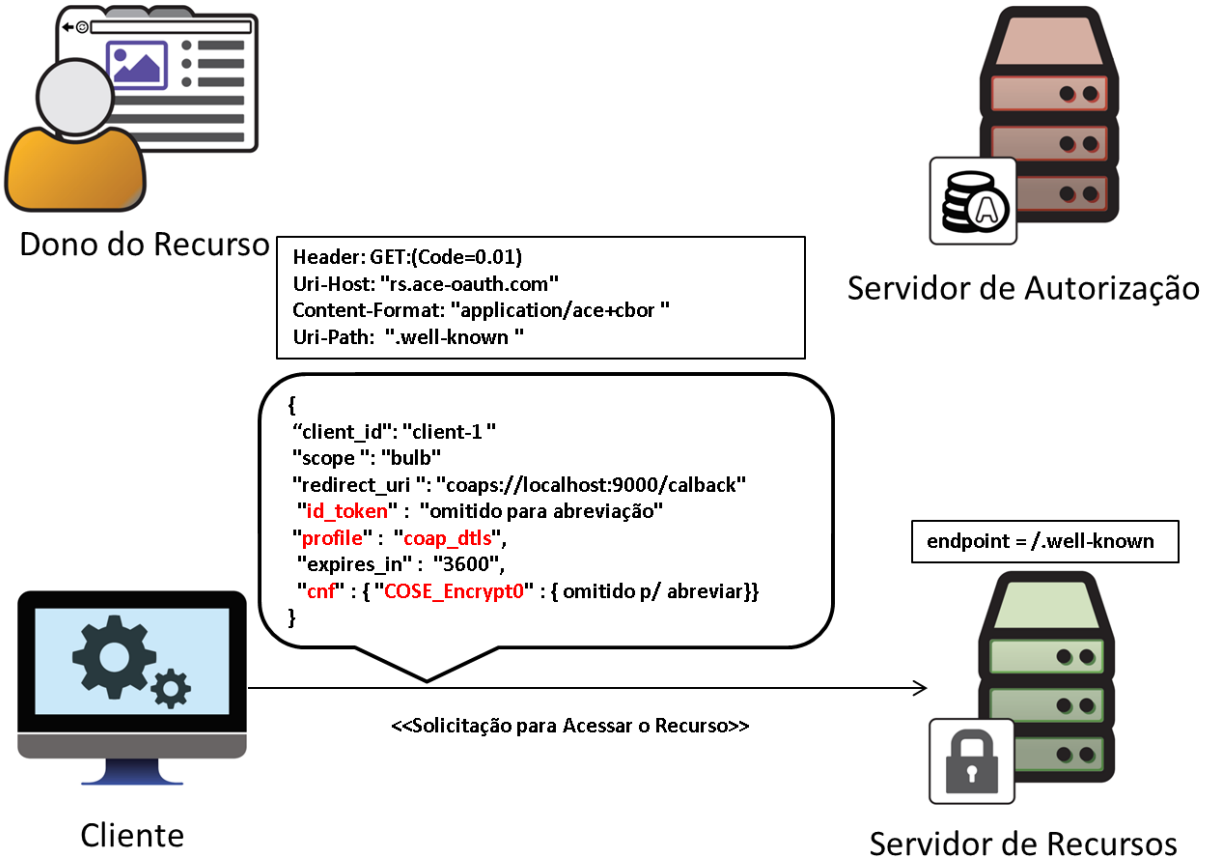

Figura 71 – Solicitação para acessar o recurso.

Na Figura 71 o servidor de recurso verifica se possui suporte para o perfil atribuído no parâmetro perfil (“`perfil`”) na requisição do cliente. Caso o servidor de recurso suporte o perfil, ele verifica o ID Token. Se o ID Token estiver em conformidade, o servidor de recurso descriptografa o conteúdo do atributo de confirmação (“`cnf`”) com a chave simétrica que é compartilhada entre o servidor de autorização e o servidor de recurso. Uma vez descriptografado a PoP é possível validar o conteúdo do *token*. Então, o servidor de recursos verifica se o *token* de acesso é válido. A verificação do seu tempo de uso de acesso não está expirado é confirmado pelo valor do atributo (“`expires_in`”).

Resposta do recurso do fluxo original. Em seguida, o servidor de recurso confirma os atributos recebidos na requisição enviada pela aplicação cliente de terceiros. Caso os atributos sejam válidos, o servidor de recurso retorna o estado do recurso ou estado da operação realizada sobre o recurso no atributo de confirmação encapsulada no objeto COSE, Figura 72. Para isso, a aplicação cliente de terceiros e o servidor de recurso devem compartilhar uma chave criptográfica, em tempo de

configuração ou implantação, sobre o conteúdo protegido enviado na resposta.

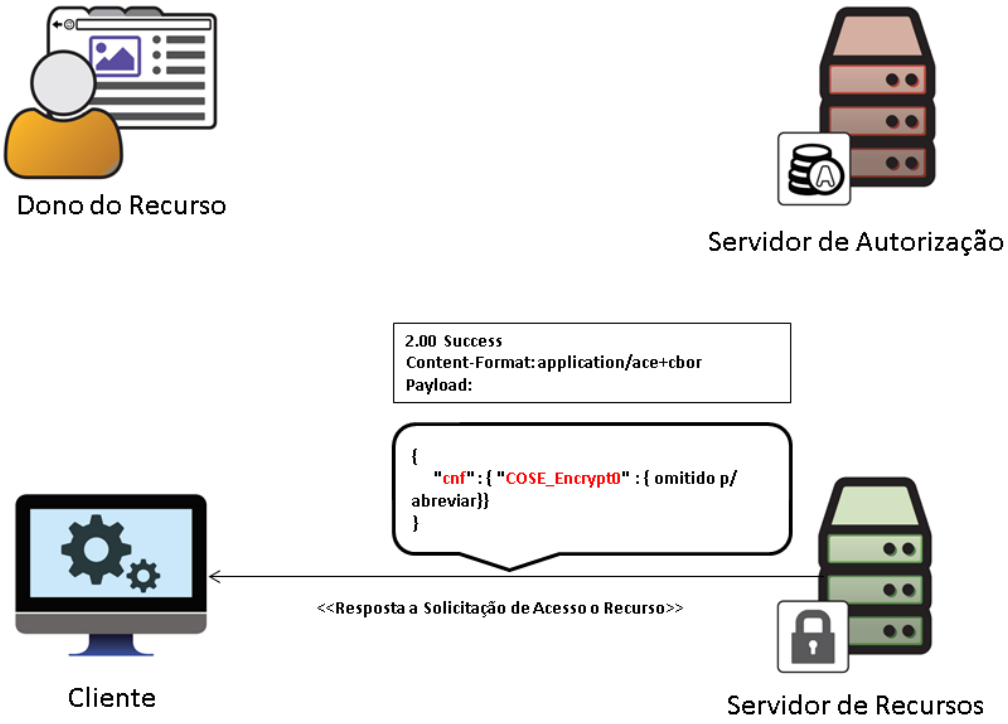

Figura 72 – Resposta a solicitação do recurso no fluxo normal.

Resposta do recurso da proposta. Se a requisição do recurso passar pelas verificações do ID Token e descriptografia com chave simétrica, o servidor envia uma resposta ao cliente com o objeto solicitado pelo cliente, Figura 73.

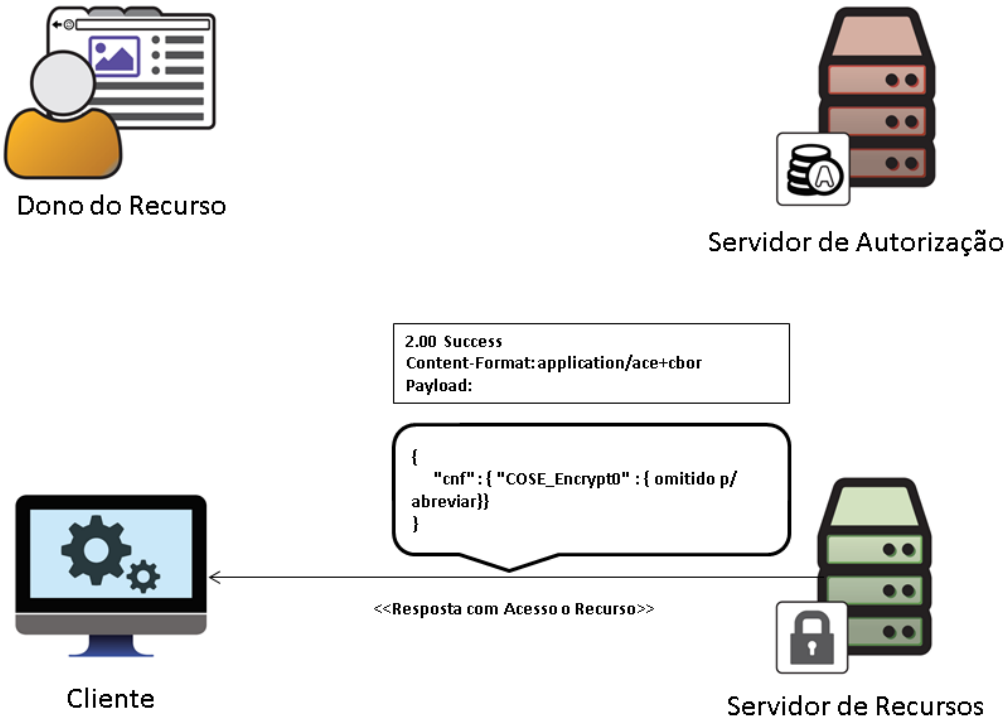

Figura 73 – Resposta com acesso o recurso.

4.3 CONSIDERAÇÃO DE SEGURANÇA

A seguir, é mostrado um resumo das características e objetivos das medidas de segurança adotadas na solução proposta nas etapas da comunicação no controle de acesso ao recursos sobre aplicações de terceiros a fim de combater os ataques de personificação e de replicação de ambientes em IoT.

Requisição e resposta de autorização. Na requisição de autorização do fluxo de concessão código de autorização foi utilizado o fluxo desafio-resposta, apresentado na Subseção 2.8.3, já que o atributo estado (“`state`”), uma sequência aleatória, já se mostrou vulnerável a ataques de personificação (RISCHER; SANSO, 2017). Destaca-se que, o desafio-resposta é adequado para comunicação orientada a conexões (STALLINGS, 2014) com a finalidade de verificar se a origem da requisição de *token* é a mesma da etapa anterior, sendo uma das medidas de mitigação contra os ataques de personificação e de replicação.

Além do uso do fluxo desafio-resposta na requisição de autorização, também foi utilizada o tipo solução de resposta (“`code id_token`”) como modo de resposta dada pelo servidor a fim de credenciar os clientes e obter um grau de confiança maior em relação à autenticação. Nesse tipo de resposta, o ID Token é emitido como mecanismo de identificação da aplicação cliente que contém atributos afirmativos, *claims*, relativos ao contexto da requisição da aplicação do cliente. Nesse caso, o cliente recebe mensagem contendo a estrutura ID Token, como resposta do servidor e, numa próxima requisição do fluxo, o cliente apresenta de volta ID Token como identificação ao servidor.

Por meio da estrutura ID Token, torna-se mais difícil o atacante supor ou reusar valores atribuídos aos parâmetros de identidade utilizados nas requisições. Porém, para criar uma solicitação válida na requisição de autorização, os valores do ID token são atualizados na requisição de registro pelo servidor de autorização, na requisição de *token*, os valores são atualizados na requisição de autorização pelo servidor autorização e na requisição de recurso, os valores são atualizados na requisição de *token* pelo servidor de autorização.

Assim sendo, o ID Token é uma opção para substituir o uso da credencial de identificação (identificador/segredo) nas requisições, já que dados da identidade são emitidos e atualizados a cada resposta da entidade de autorização. Além disso, o ID Token oferece mecanismos alternativos de autenticação ao cliente (CAMPBELL et al., 2015). Desse modo, a estrutura ID Token é uma contramedida útil contra o ataque de personificação, já que reduz a chance do atacante se passar por uma aplicação cliente válida para obter uma sessão do usuário e acarretar a modificação ou a destruição dos recursos.

Entretanto, tanto o desafio-resposta quanto a estrutura ID Token devem trafegar por meio do canal seguro ou precisam ser criptografados. Nesse sentido, o estabelecimento do canal de segurança com DTLS é fundamental no envio da requisição do cliente e na resposta do servidor de autorização, pois evita que o conteúdo confidencial das mensagens seja monitorado. Desse modo, a partir do uso canal de segurança, há uma garantia de confidencialidade na troca de mensagens.

Requisição *e* resposta do *token*. Na requisição do *token*, é enviado o código de autorização, emitido na resposta de autorização pelo servidor de autorização, e a chave *hash,* que gerou desafio-resposta, criada na requisição de autorização pelo

cliente. Essas duas medidas reforçam a confirmação da origem e evitam os ataques de personificação e replicação. Além disso, foi utilizado na solução proposta o tipo de resposta (“`token id_token`”) dada pelo servidor de autorização.

Requisição e resposta do recurso. O perfil de comunicação e de segurança CoAP/DTLS (Subseção 4.2.4) utilizado na proposta permite proteger as credenciais de acesso por meio da prova de posse na solicitação de recurso, evitando ataque de personificação, já que o atacante precisaria descobrir a chave criptografada para alterar o conteúdo confidencial e, em consequência disso, pode conseguir consultar ou alterar os recursos protegidos.

4.4 MEDIDAS DE MITIGAÇÃO CONTRA AS AMEAÇAS E ATAQUES

Com base no fluxo de desafio-resposta e no fluxo de veracidade como mostrado para reforçar o fluxo de concessão código de autorização utilizado no controle de acesso em ambiente de IoT, a solução proposta fornece as seguintes medidas de mitigação contra as ameaças e os ataques de personificação e replicação, Tabela 11.

Tabela 11 – Medidas de mitigação contra as categorias dos ataques.

Entidades Envolvidas	Vulnerabilidades	Mitigação
Cliente<->Servidor de Autorização	Personificação da sessão do usuário	(1) Utiliza a estrutura ID Token para controle das requisições a fim de permitir a emissão de credencial de acesso e de código de autorização
Cliente<->Servidor de Autorização	Combinação diferente dos atributos do identificador do cliente e redirecionamento URI	(1) Para toda requisição do cliente é necessário vincular o redirecionamento do URI e realizar a verificação pelo servidor de autorização de identificador
Cliente<->Servidor de Autorização	Obtenção dos segredos do cliente	(1) Utiliza a estrutura ID Token para controle das requisições a fim de permitir a emissão de código de autorização
Cliente<->Servidor de Autorização	Replicação de solicitações ao servidor de autorização	(1) Canal de Segurança com Perfil CoAP DTLS e com chave simétrica compartilhada (2) Camada de Segurança sobre a Prova de Posse e Estrutura ID_TOKEN
Cliente<->Servidor de Autorização	Não limitar o número de requisições por uso ou uso único sobre o tempo	(1) Uso da estrutura do ID_TOKEN para limitar número de requisições baseado no tempo

Entidades Envolvidas	Vulnerabilidades	Mitigação
Cliente<->Servidor de Autorização	Solicitações não assinadas	(1)Autenticação mútua, (2)Esquema de troca de chave (3)Código de Autenticação da Mensagem baseado em hash
Cliente<->Servidor de Autorização	Replicação do código de autorização	(1) Uso da estrutura do ID_TOKEN (2) Canal de Segurança com Perfil CoAP/DTLS que com chave simétrica compartilhada
Cliente<->Servidor de Autorização	Sessão do usuário	(1)Autenticação mútua, (2)Esquema de troca de chave (3)Código de Autenticação da Mensagem baseado em hash
Cliente<->Servidor de Autorização	Confidencialidade insegura	(1) Canal de Segurança com Perfil CoAP/DTLS que com chave simétrica compartilhada (2) Camada de Segurança sobre a Prova de Posse e Estrutura ID_TOKEN
Cliente<->Servidor de Autorização	Divulgação de credenciais de cliente durante uma transmissão	Canal de Segurança com Perfil CoAP DTLS com chave simétrica compartilhada

Entidades Envolvidas	Vulnerabilidades	Mitigação
Cliente<->Servidor de Recurso	Replicação de solicitações do servidor de recursos	Utiliza uma estrutura ID *Token* para controle das requisições a fim de permitir emissão de credencial de acesso e de código de autorização
Cliente<->Servidor de Recurso	Reproduzir *tokens* em um servidor de recurso específico (público-alvo)	(1) O servidor de autorização é pré-configurado com os atributos de identificação do cliente, redirecionamento de URI e escopo juntos terão acesso quais recursos (2) É necessário obter a prova de posse e a estrutura do ID Token
Cliente<->Servidor de Recurso	Requisições não assinadas	Utilizou a estrutura ID Token para controle das requisições sobre o controle dos recursos protegidos
Cliente<->Servidor de Recurso	Divulgação de credenciais de cliente durante uma transmissão	(1) Canal de Segurança com Perfil CoAP/DTLS com chave simétrica compartilhada; (2) Camada de Segurança sobre a Prova de Posse e Estrutura ID_TOKEN
Cliente<->Servidor de Recurso	Personificação da sessão do usuário	Utiliza a estrutura ID Token para controle das requisições sobre o controle dos recursos protegidos

Entidades Envolvidas	Vulnerabilidades	Mitigação
Cliente<->Servidor de Recurso	Combinação diferente dos atributos do identificador do cliente e redirecionamento URI	O servidor de autorização é pré-configurado com quais os atributos de identificação do cliente, redirecionamento de URI e escopo juntos terão acesso quais recursos.

4.5 AVALIAÇÃO DE NÃO CONFORMIDADE A AMEAÇAS

Nesta seção, é apresentada a avaliação de segurança que visa confirmar se os mecanismos com reforço de segurança propostos na solução funcionam como contramedidas aos ataques de personificação e de replicação. Essa avaliação de segurança consistiu na avaliação de não conformidade a ameaças sobre o modelo de comunicação.

O processo de avaliação de não conformidade a ameaças adotado neste trabalho consistiu na seguinte metodologia: objetivo da avaliação, levantamento da base de dados de ameaças, definição e representação do modelo de comunicação, construção das restrições de não conformidade a ameaças e resultados da avaliação da não conformidade a ameaças sobre modelos de comunicação.

4.5.1 Objetivo da Avaliação

O objetivo da avaliação de não conformidade a ameaças é confirmar ou não se o reforço aplicado pela solução proposta pode mitigar os ataques de personificação e replicação abordados na solução proposta neste trabalho. Para confirmar a hipótese abordada neste trabalho, a avaliação de não conformidade a ameaças pode direcionar na busca da resposta a partir da seguinte questão: “O fluxo com reforço da proposta possui menor chance de ser de alvo de uma ameaça do que o fluxo normal?

4.5.2 Levantamento de Base de Ameaças

A base de ameaças considerada como de referência neste trabalho foi a

especificação do modelo de ameaças do OAuth 2.0[53], dado que o ACE-OAuth ainda não possui um conjunto de ameaças desenvolvida até o momento pela comunidade. Da base de ameaças do modelo de ameaças do OAuth 2.0, apenas um conjunto de treze ameaças fazem referências às categorias de ataques de personificação e replicação que ocorrem a partir da troca de mensagens do modelo de comunicação. E a base de ameaças do OAuth 2.0 é mais adequada para avaliar o modelo de comunicação do ACE-OAuth 2.0, já que parte das ameaças do ACE-OAuth pode ser herdadas do OAuth 2.0 e a extração das ameaças de propósito geral da base STRIDE, como apresentado no trabalho (ECHEVERRÍA; KLINEDINST; SEITZ, 2019), é muito genérica para ser utilizada na avaliação de segurança em domínio específico, como é no caso de ambientes em IoT.

4.5.3 Definição e Representação do Modelo de Comunicação

Como forma de combater as ameaças dirigidas ao ACE-OAuth e que estejam associadas aos ataques de personificação e replicação, foram desenvolvidos os modelos de comunicação baseados em diagrama de fluxo de dados. O primeiro modelo de comunicação foi baseado no fluxo de concessão de código de autorização original e ao segundo modelo, desse mesmo fluxo, foi acrescentado o reforço de segurança. A ferramenta de modelagem dos fluxos de dados e da avaliação de não conformidade de ameaças utilizados deste trabalho foi o Microsoft *Security Development Lifecycle* (SDL)[54].

Na modelagem de comunicação, os atores foram representados por círculos, as fronteiras por linhas tracejadas e a trocas de mensagens pelos arcos. O modelo de comunicação da proposta contém cinco atores, duas fronteiras e dezoito trocas de mensagens.

As atribuições nos atores e nas fronteiras foram desconsideradas, já que o foco da avaliação de conformidade se preocupou nas ameaças presentes, exclusivamente, na troca de mensagens dos modelos de comunicação, onde pode ocorrer investida de monitoramento ou de interceptação.

Por outro lado, nos arcos, foram informados valores aos atributos (autenticação da origem, autenticação do destino, confidencialidade de origem e confidencialidade

[53] OAuth 2.0 *Threat Model and Security Consideration*: https://tools.ietf.org/html/rfc6819.

de destino, integridade, cifragem do conteúdo da mensagem, uso de credencial de autorização, uso da credencial de *token*, uso da credencial do cliente, *claims*, vinculação de atributos, requisições assinadas, consentimento do dono do recurso). Os arcos que ligam os atores formam as interações que são alvos da análise e da avaliação que são descritas a seguir.

Cliente e armazenamento de credenciais. A Figura 74 mostra os objetos (cliente e armazenamento) e os fluxos (armazena a credencial de identificação e recupera a credencial de identificação).

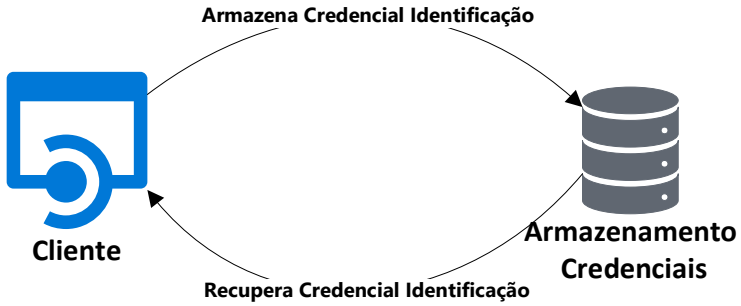

Figura 74 – Cliente e armazenamento credencial.

Cliente e o servidor de autorização. A Figura 75 mostra os objetos (cliente e o servidor de autorização) e os fluxos (solicita credencial de identificação, emite credencial de identificação, solicita credencial de autorização, emite credencial de autorização, solicita credencial de acesso, emite credencial de acesso).

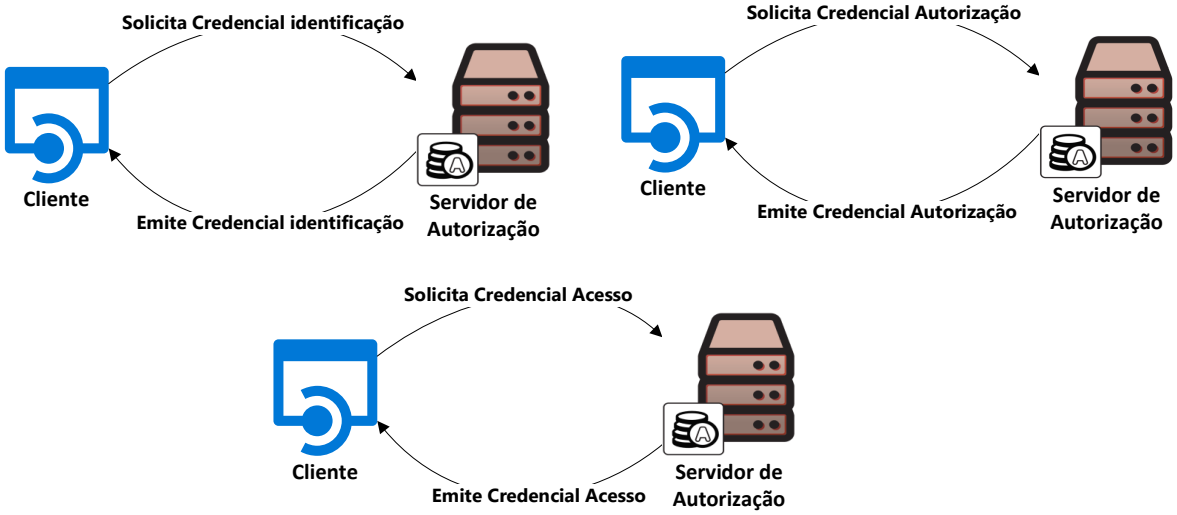

Figura 75 – Cliente e o servidor de autorização e de recursos.

Servidor de autorização e o armazenamento da credencial. A Figura 76 mostra os objetos (servidor de autorização e o armazenamento de credenciais) e os fluxos (armazena da credencial de identificação, recupera credencial de identificação, armazena credencial de autorização, recupera credencial de autorização, armazena credencial de acesso, recupera credencial de acesso).

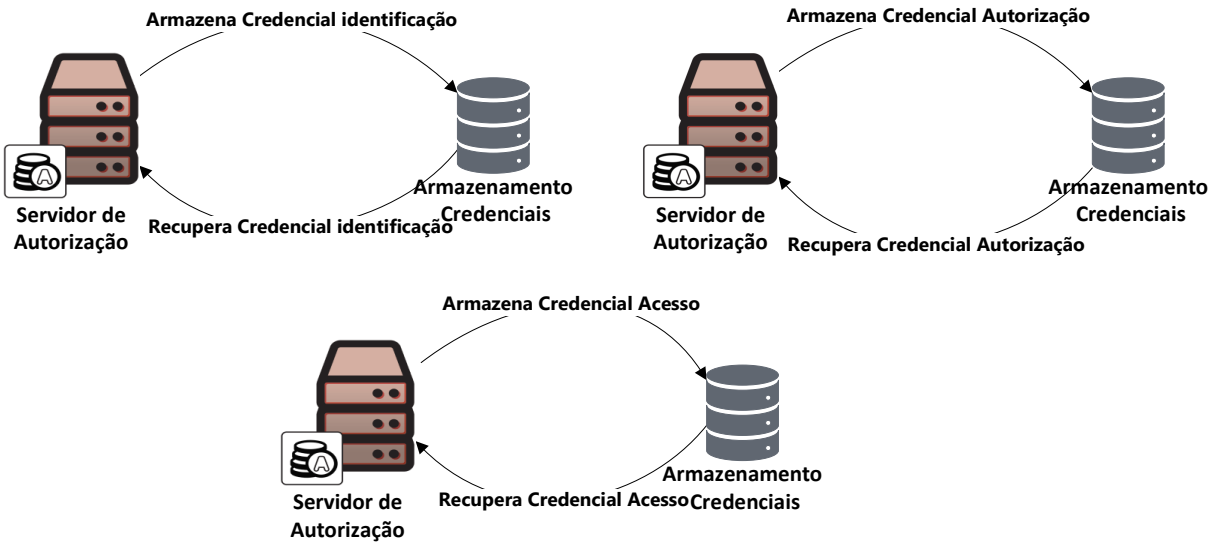

Figura 76 – Servidor de autorização e de recursos e o armazenamento da credencial.

Cliente e servidor de recurso. A Figura 77 mostra os objetos (cliente e o servidor de recurso) e os fluxos (transfere credencial acesso, recupera estado do recurso).

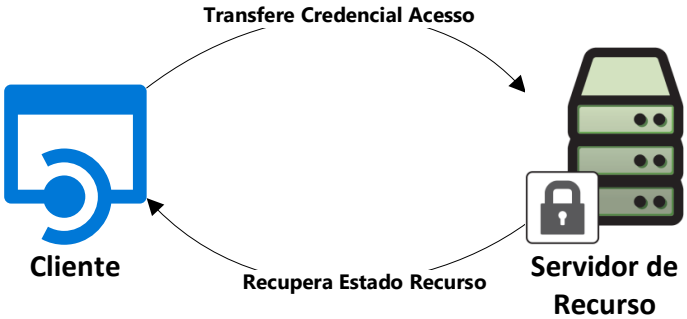

Figura 77 – Cliente e servidor de recurso.

Servidor de recurso e servidor de descoberta de recursos. A Figura 78 mostra os objetos (servidor de recurso e servidor de descoberta de recurso) e os fluxos (solicita estado do recurso e envia estado do recurso).

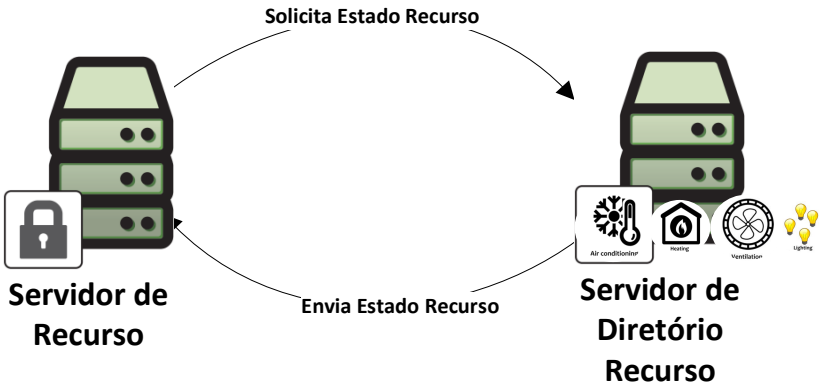

Figura 78 – Servidor de recurso e servidor de descoberta de recursos.

4.5.4 Construção das Restrições de Não Conformidade a Ameaças

Depois de construído os modelos de comunicação, foram desenvolvidas as restrições ou condições de conformidade não a ameaças sobre o domínio do modelo de comunicação. O resultado da avaliação de não conformidade a ameaças depende das restrições ou condições criadas que indicam se ameaça pode ocorrer ou não no modelo de comunicação. Nesta ocasião, essas condições avaliam diversos atributos apresentados na definição do modelo de comunicação, principalmente, nas trocas de mensagens. Desse modo, por exemplo, uma condição de ameaça de replicação é considerada verdadeira sobre a troca de mensagem, quando o valor do atributo comunicação assinada é como falsa, Tabela 12.

Tabela 12 – Algumas condições lógicas como restrição de ameaças para avaliação de não conformidade a ameaças.

ID	Restrição
1	(fluxo é [Solicita Credencial de Autorização]) ou (fluxo é [Solicita Credencial Identificação]) ou (fluxo é [Solicita Credencial Acesso]) e (fluxo.[Comunicação Assinada] é 'Não')
2	(fluxo é [Transfere Token Acesso]) e (fluxo.[Token] é 'Sim') e (fluxo.[claims] é 'Não')
3	(fluxo é [Solicita Consentimento Dono Recurso]) e (fluxo.[Compartilhamento Recursos Origem Cruzada] é 'Não')

Portanto, resumindo, no resultado da avaliação de não conformidade a ameaças, as ameaças são contabilizadas quando valores assumidos no modelo de comunicação não estão em conformidade. As restrições ou condições na avaliação de não conformidade a ameaças no modelo de comunicação utilizadas neste trabalho se encontram no Apêndice H .

4.5.5 Avaliação de Não Conformidade a Ameaças sobre Modelos de Comunicação

Na Tabela 13 são relacionados os resultados encontrados a partir da avaliação e não conformidade de ameaças sobre o modelo de comunicação levando em consideração se a restrição não obedecia à conformidade, em qual interação ocorreu e qual categoria de ameaça foi diagnosticada.

Tabela 13 – Relacionamento de categoria do ataque, interação e identificador da restrição.

Proposta	Categorias de Ameaças Diagnosticadas	Interação na Comunicação	Identificador Restrição ou Condição
Fluxo de concessão código de autorização original	Requisições não assinadas	Solicita Credencial de Acesso	1
	Requisições não assinadas	Solicita Credencial de Identificação	1
	Requisições não assinadas	Solicita Credencial de Autorização	1
	Personificação do dono do recurso	Solicita Consentimento Dono Recurso	3
	Tempo de expiração elevado no uso do token	Transferir Token de Acesso	2
Esta Proposta	Requisições não assinadas	Solicita Credencial de Acesso	1
	Requisições não assinadas	Solicita Credencial Identificação	1
	Requisições não assinadas	Solicita Credencial de Autorização	1

Como mostrado na Tabela 13, as modificações no modelo de comunicação do fluxo proposto reduziram em torno de 30 pontos percentuais o percentual de ameaças não mitigadas em comparação com o modelo de comunicação do fluxo original. Com isso, pode-se afirmar que, a resposta é positiva para indagação da avaliação de não conformidade das ameaças e sustenta a solução proposta como resposta a hipótese deste trabalho.

Tabela 14 – Percentual de ameaças não mitigadas de acordo com avaliação de não conformidade a ameaças sobre o modelo.

Modelo de Comunicação	Atores	Fluxo de Comunicação	Pontos Percentuais de Não-Conformidade
Fluxo de concessão código de autorização	5	20	53,84%
Esta Proposta	6	18	23,07%
(ECHEVERRÍA; KLINEDINST; SEITZ, 2019)	6	18	23,07%

4.6 ANÁLISE DE RISCOS BASEADA EM ÁRVORE DE ATAQUES

A análise de riscos adotada nesta avaliação de segurança é baseada na árvore de ataques. A árvore de ataques costuma ser usada para identificar as ações realizadas pelo atacante (SCHNEIER, 1999). Adicionalmente, o modelo da árvore de ataques descreve as vulnerabilidades em um sistema e as ações que poderiam ser realizadas por um adversário a fim de ter sucesso no ataque (AMENAZA, 2006). Além disso, a árvore de ataques inclui estimativas dos recursos necessários para realizar ataques específicos, qual é o impacto desses ataques sobre a vítima e os quais os benefícios adquiridos pelo atacante (AMENAZA, 2006). Além do mais, a análise de riscos baseada em árvore de ataques permite predizer quais ataques são mais prováveis e ajuda a tomar decisões mais estratégicas sobre a defesa (AMENAZA, 2006).

4.6.1 Objetivos da Análise de Risco

A análise de risco tem como finalidade identificar os ataques mais prováveis de serem executados pelo atacante. Esses ataques podem ser priorizados em relação à baixa complexidade, ao uso de recursos disponíveis e aos benefícios adquiridos pelo agente da ameaça, por exemplo. A análise de risco realizada nesse trabalho usou a estrutura de árvore de ataques para modelar, classificar e avaliar os ataques.

4.6.2 Metodologia do Processo de Análise de Risco

Geralmente, uma metodologia universal requer quatro passos que são recomendados para realizar o processo de análise de risco (AMENAZA, 2003):

1. Produzir um modelo de ataque que mostre como um incidente ocorre – Isso inclui a representação de vulnerabilidades e de recursos requeridos para explorá-las assim como o impacto para a vítima de vários ataques.
2. Definir o tipo de agente da ameaça que explorará as vulnerabilidades – Identificar quais são os inimigos. Predizer onde e como eles agem.
3. Produzir uma lista priorizada dos riscos associados com cada tipo de ataque – A lista priorizada oferece toda a informação necessária para justificar quais problemas necessitam ser resolvidos e demonstrar que ação corretiva proposta tem um custo justificado. Como parte do processo para obter a lista priorizada, foi aplicada a técnica poda da árvore (Subseção 4.6.7) para análise da árvore de ataque.
4. Propor estratégias de mitigação efetiva – Mostra como as mudanças propostas para o sistema irão oferecer melhoramentos efetivos.

4.6.3 Árvore de Ataque

A estrutura da árvore é composta por nós. Um nó na árvore pode ser uma raiz, galho ou folha. O nó raiz representa o objetivo a ser alcançado pelo atacante. Esse nó, também, representa a consequência negativa para o defensor, Figura 79.

Figura 79 – Nó raiz.

Por outro lado, o nó raiz pode ser decomposto em vários objetivos secundários que representam as subárvores ou galhos do nó raiz, também conhecido como nós intermediários, Figura 80.

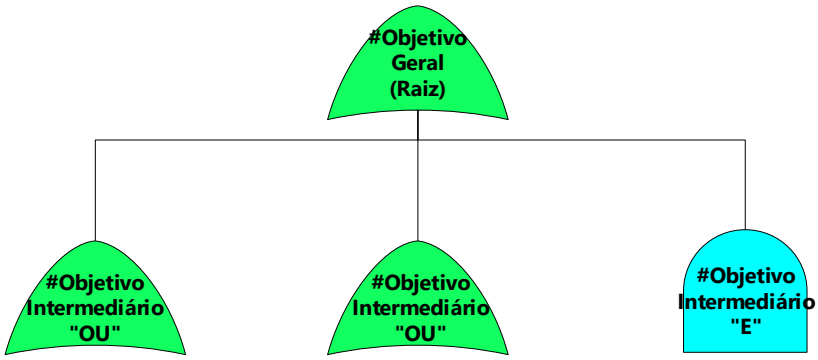

Figura 80 – Nós intermediários.

Por fim, os nós filhos da subárvore podem conter nó galho ou nó folha. O nó folha representa uma tarefa específica realizada pelo atacante e é último grau da árvore. Dessa maneira, a árvore representa o cenário de ataque em que permite analisar as opções de ataque do atacante, Figura 81.

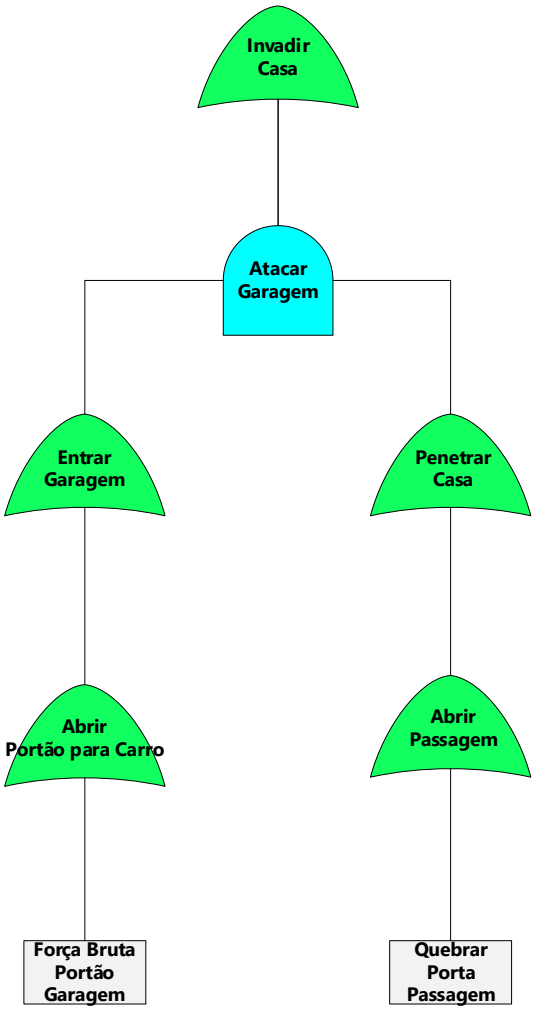

Figura 81 – Exemplo de árvore de ataque para invasão da casa.

4.6.4 Modelagem da Árvore de Ataques

A modelagem da árvore de ataque foi gerada tanto para o fluxo de concessão código de autorização do ACE-OAuth como para o fluxo de concessão código de autorização mais reforço de segurança da solução proposta.

A árvore de ataque foi hierarquizada em quatro níveis. O nível raiz da árvore é o objetivo ou o estado que um atacante deseja alcançar (AMENAZA, 2006). O objetivo principal é decomposto em subobjetivos ou tarefas específicas (AMENAZA, 2006). Após o nível raiz, o primeiro nível indica os subobjetivos que são representados pela requisição e pela resposta (autorização, *token* e acesso) realizadas entre cliente e os servidores. Já o segundo nível divide a subárvore em ataques de personificação ou replicação. E, o terceiro nível representa a categoria do ataque. Por fim, no último nível, as folhas da árvore representam os ataques específicos, as quais descrevem os ataques que um agente de ameaça pode realizar (AMENAZA, 2006).

A modelagem da árvore de ataque para a proposta foi dividida entre as requisições (autorização, *token* e acesso) e respostas (autorização, *token* e acesso). Foi utilizada a ferramenta SecurlTree[55] para a modelagem da árvore de ataque. Por meio da modelagem, foi definido o nó inicial da árvore que represente o objetivo principal o qual é dividido em subojetivos representados pelos nós das requisições (autorização, autorização, *token*) e nós das respostas (*token*, acesso, acesso), Figura 82.

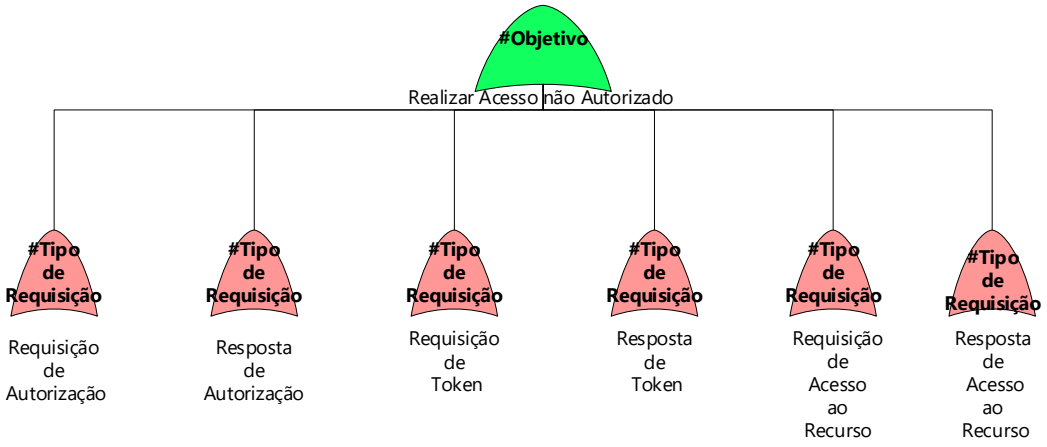

Figura 82 – Árvore de ataque sobre a proposta no nível 1

[55] Ferramenta para análise de risco baseado em árvore de ataque (SecurlTree): https://www.amenaza.com/request_trial.php

Requisição de Autorização. A Figura 83 mostra as possíveis categorias dos ataques e os diversos ataques específicos descritos como:

- Confidencialidade insegura (acesso completo do conteúdo da requisição);
- Obtenção do segredo do cliente (replicação dos *tokens* de *refresh* e códigos de autorização);
- Requisições não assinadas (requisição sem um identificador único);
- Não limitar o número de requisições por uso ou uso único sobre o tempo (o servidor de autorização não restringe o número de requisições, operações realizadas podem ser realizadas com certo *token*);
- Replicação de solicitações ao servidor de autorização (replicação requisições válidas para obter dados do usuário, modificar ou destruir dados do usuário);
- Replicação do código de autorização (o servidor de autorização não vincula o identificador do cliente ao código de autorização);
- Vazamento da credencial do cliente durante a transmissão (escuta na transmissão de credenciais de cliente entre o cliente e o servidor de recurso);
- Personificação da sessão do usuário (personificar a sessão do usuário sobre o cliente);
- Combinação diferente dos atributos (o servidor de autorização poderá não vincular o atributo identificador do cliente ao atributo de redirecionamento URI pré-configurado).

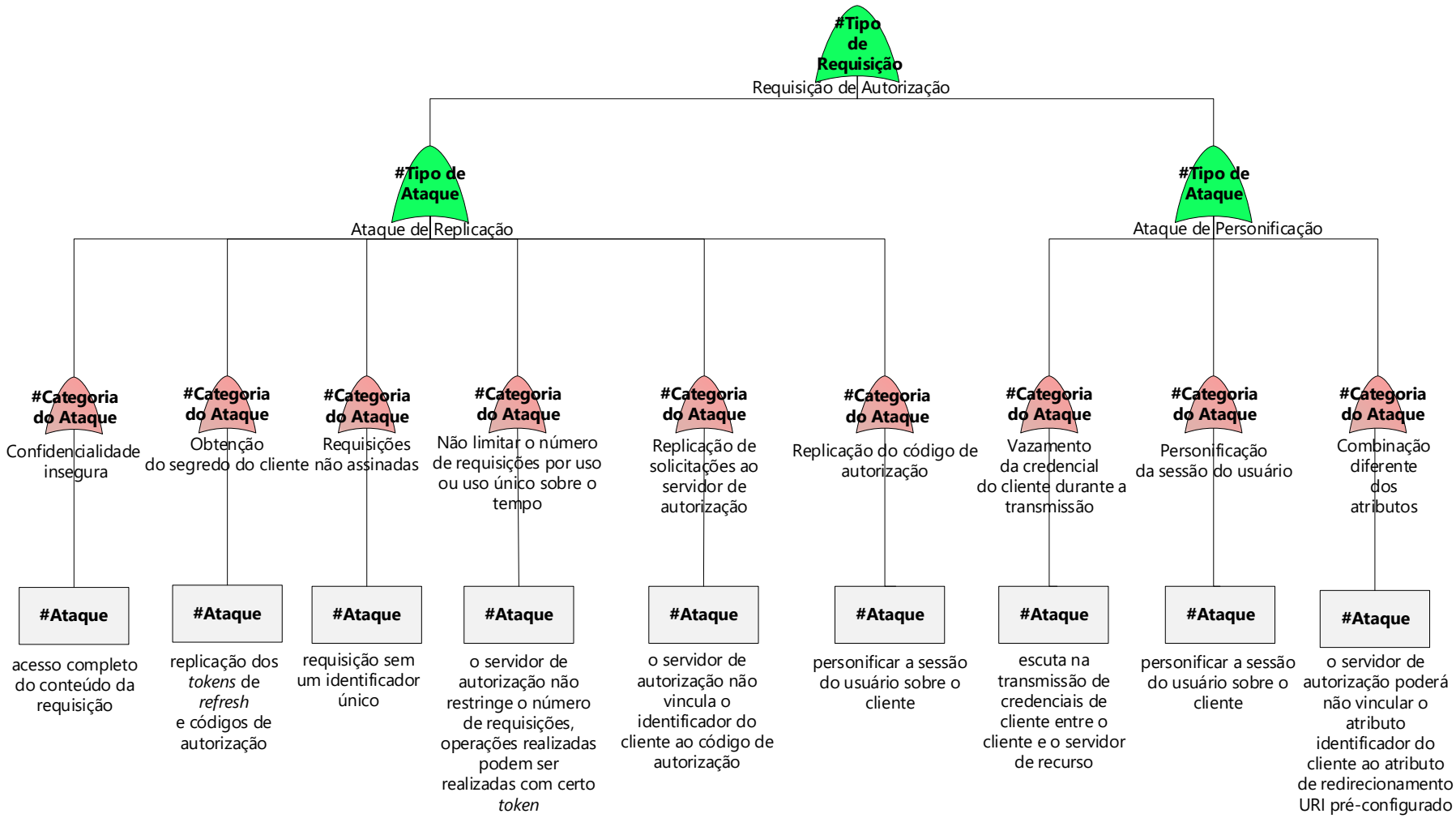

Figura 83 – Subárvore de ataque da requisição de autorização.

Resposta de Autorização. A Figura 84 mostra as possíveis categorias dos ataques e os diversos ataques específicos descritos como:

- Vazamento da credencial do cliente durante a transmissão (escuta na transmissão de credenciais entre o cliente e o servidor de autorização);
- Personificação da sessão do usuário (personificação no lado cliente);
- Confidencialidade insegura (acesso completo do conteúdo da resposta).

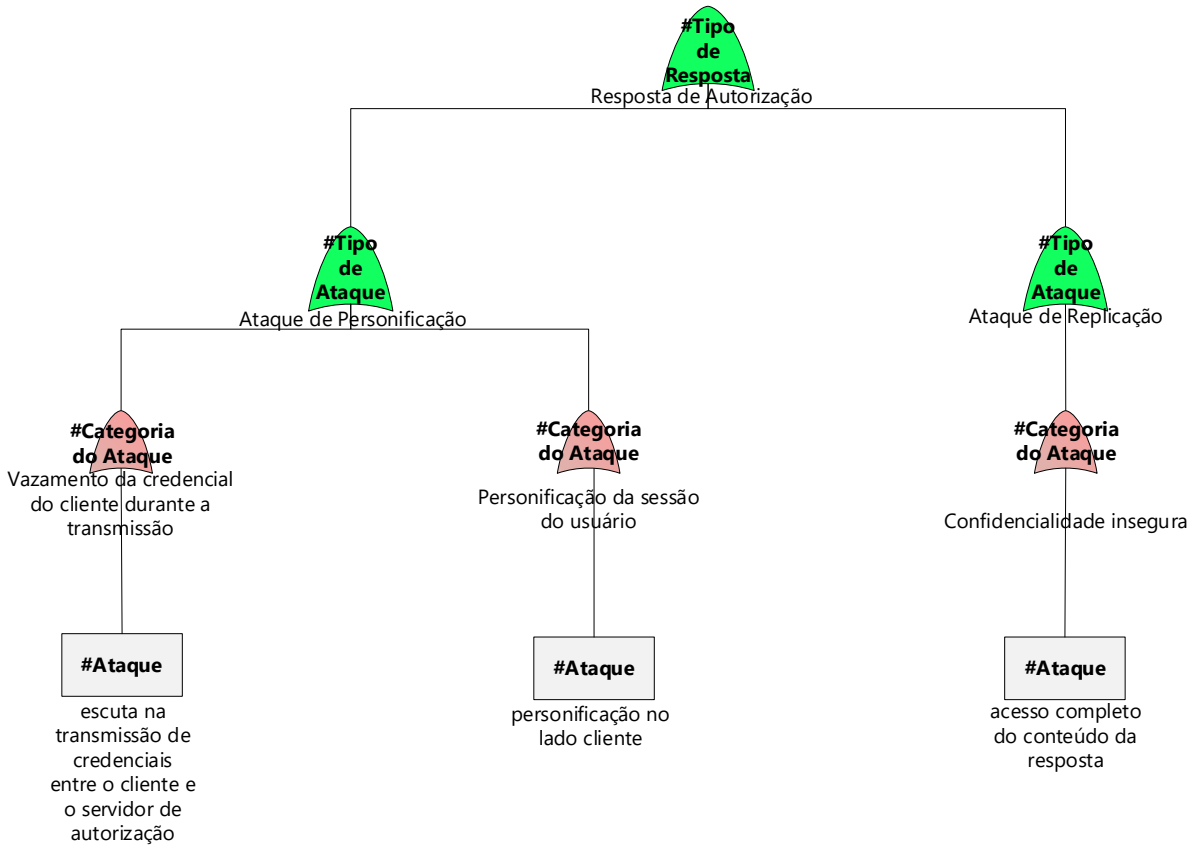

Figura 84 – Subárvore de ataque da resposta de autorização

Requisição de Token. A Figura 85 mostra as possíveis categorias dos ataques e os diversos ataques específicos descritos como:

- Confidencialidade insegura (acesso completo do conteúdo da requisição);
- Obtenção do segredo do cliente (replicação dos *tokens* de *refresh* e códigos de autorização);
- Requisições não assinadas (requisição sem um identificador único);
- Não limitar o número de requisições por uso ou uso único sobre o tempo (o servidor de autorização não restringe o número de requisições, operações realizadas podem ser realizadas com certo *token*);

- Replicação de solicitações ao servidor de autorização (replicação requisições válidas para obter dados do usuário, modificar ou destruir dados do usuário);
- Replicação do código de autorização (o servidor de autorização não vincula o identificador do cliente ao código de autorização);
- Vazamento da credencial do cliente durante a transmissão (escuta na transmissão de credenciais de cliente entre o cliente e o servidor de recurso);
- Personificação da sessão do usuário (personificar a sessão do usuário sobre o cliente);
- Combinação diferente dos atributos (o servidor de autorização poderá não vincular o atributo identificador do cliente ao atributo de redirecionamento URI pré-configurado).

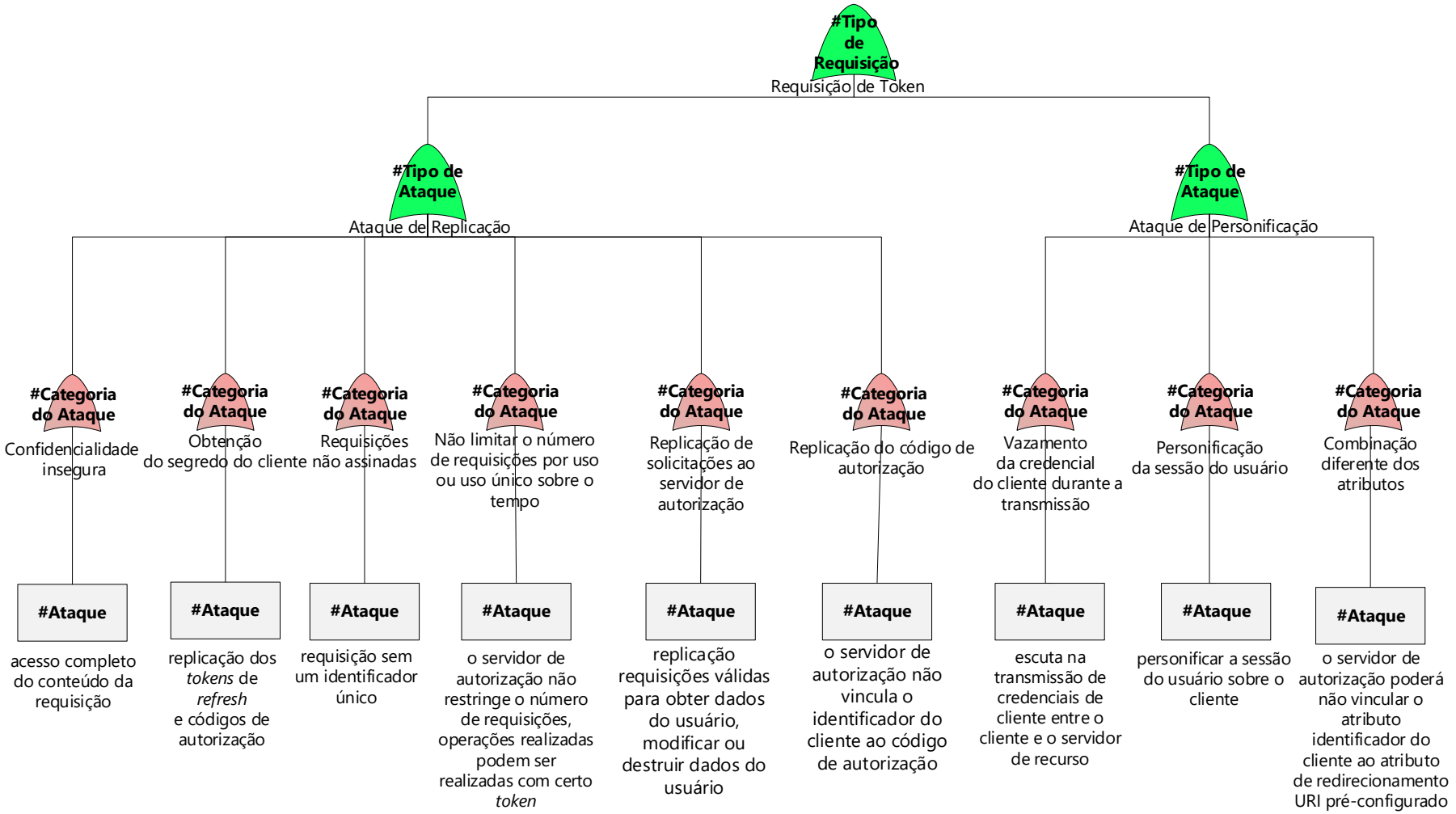

Figura 85 – Subárvore de ataque da requisição de *token*.

Resposta de Token. A Figura 86 mostra as possíveis categorias dos ataques e os diversos ataques específicos descritos como:

- Vazamento da credencial do cliente durante a transmissão (escuta na transmissão de credenciais entre o cliente e o servidor de autorização);
- Personificação da sessão do usuário (personificação no lado cliente);
- Confidencialidade insegura (acesso completo do conteúdo da resposta).

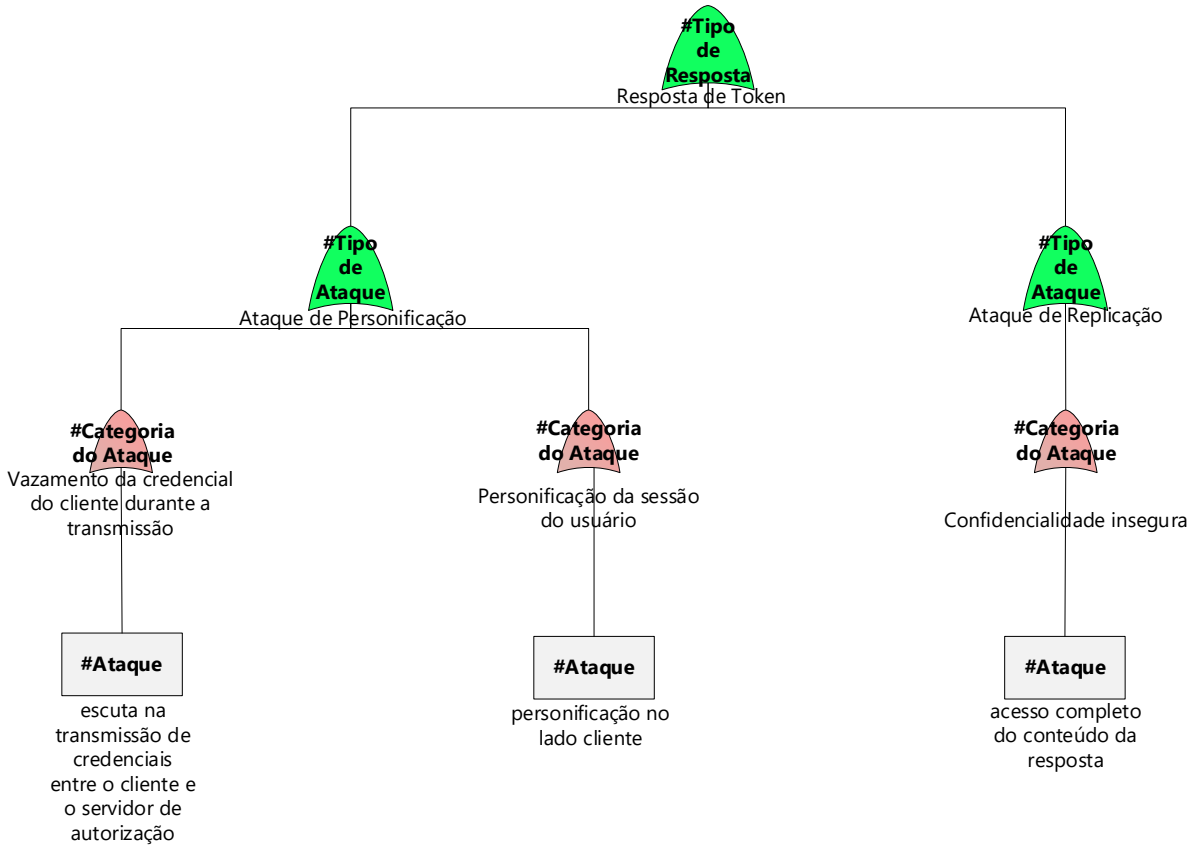

Figura 86 – Subárvore de ataque da resposta de token.

Requisição de Acesso ao Recurso. A Figura 87 mostra as possíveis categorias dos ataques e os diversos ataques específicos descritos como:

- Replicação de solicitações ao servidor de recurso (replicação requisições válidas para obter dados do usuário, modificar ou destruir dados do usuário);
- Replicação de *tokens* para o servidor de recurso particular (servidores de autorização desconsideram emitir *tokens* com conteúdo diferente para

diferentes servidores de recursos, não indicar explicitamente no *token* qual o servidor de recurso alvo para o qual um *token* deve ser enviado);

- Requisições não assinadas (requisição sem um identificador único);
- Combinação diferente dos atributos (o servidor de recurso poderá não vincular o identificador ao atributo de redirecionamento URI pré-configurada);
- Confidencialidade insegura (acesso completo ao conteúdo da requisição);
- Não limitar o número de requisições por uso ou uso único sobre o tempo (o servidor de autorização não restringe o número de requisições, operações realizadas podem ser realizadas com certo *token*);
- Vazamento da credencial do cliente durante a transmissão (escuta na transmissão de credenciais de cliente entre o cliente e o servidor de recurso);
- Personificação da sessão do usuário (personificar a sessão do usuário no lado cliente);
- Combinação diferente dos atributos (o servidor de recurso poderá não vincular o identificador ao atributo de redirecionamento URI pré-configurada).

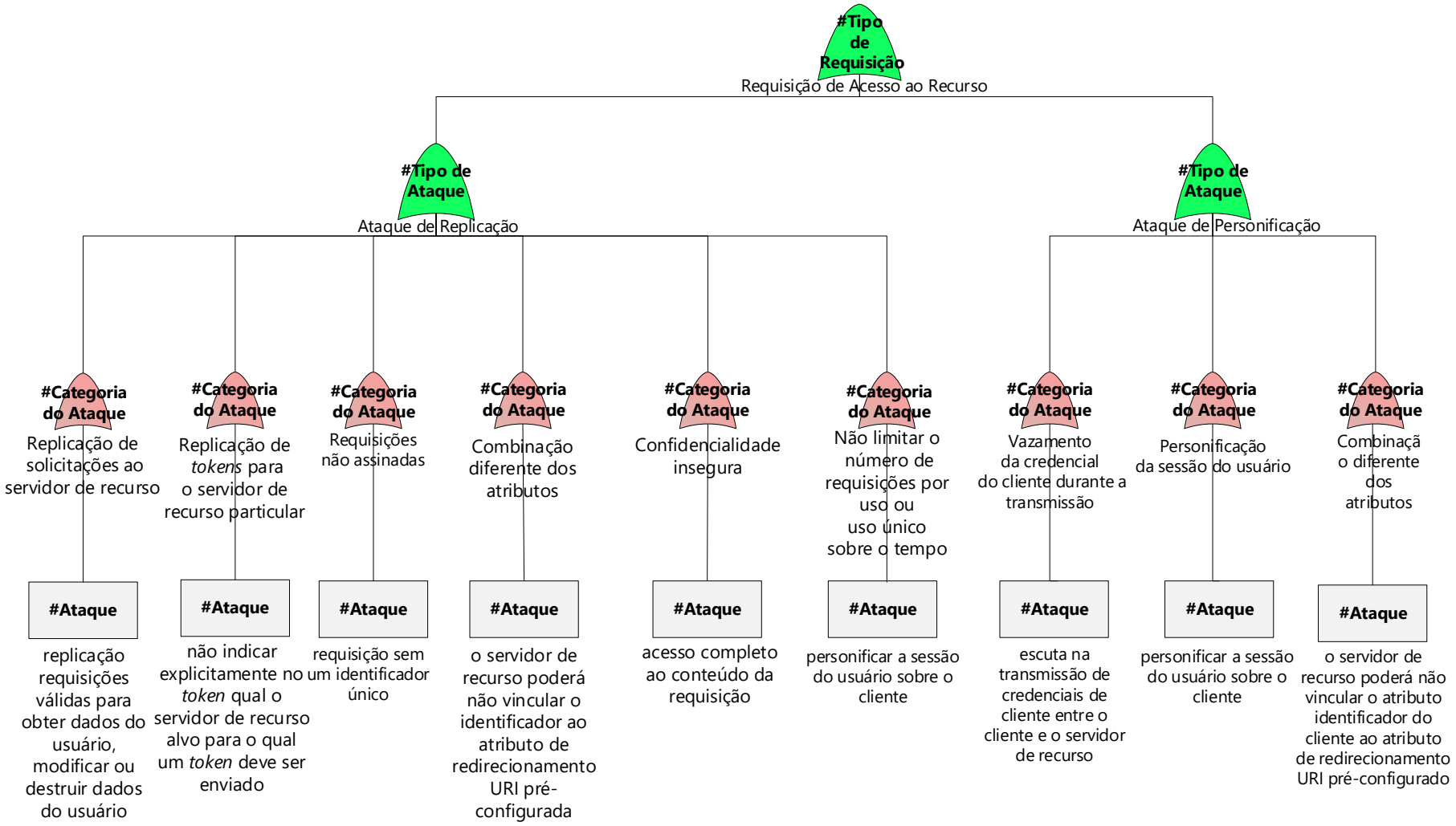

Figura 87 – Subárvore da requisição de acesso ao recurso

Resposta de Acesso ao Recurso. A Figura 88 mostra as possíveis categorias dos ataques e os diversos ataques específicos descritos como:

- Vazamento da credencial do cliente durante a transmissão (escuta na transmissão de credenciais entre o cliente e o servidor de recurso);
- Personificação da sessão do usuário (personificação no lado cliente);
- Confidencialidade insegura (acesso completo do conteúdo da resposta).

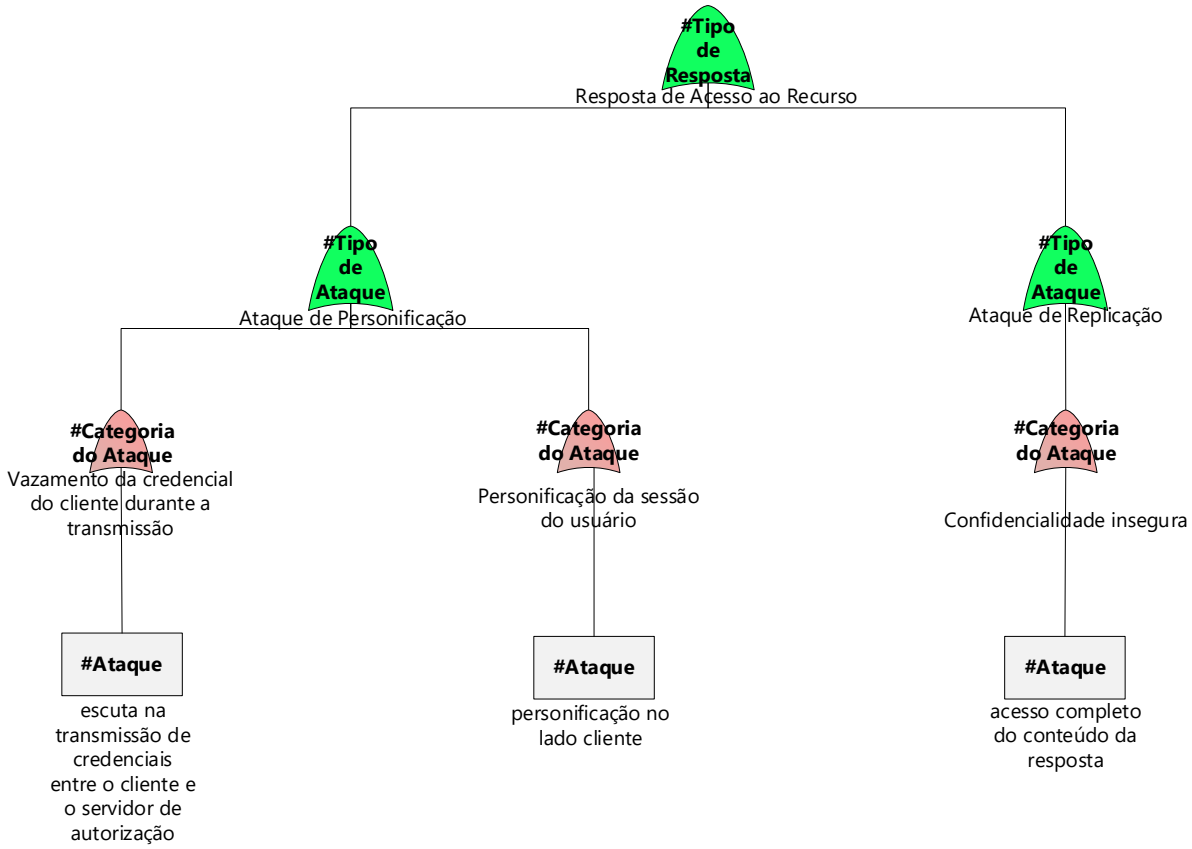

Figura 88 – Subárvore da requisição de acesso ao recurso.

4.6.5 Configuração da Análise de Risco sobre Árvore de Ataque

O cenário da análise do risco do ataque foi baseado em capacidades. As capacidades servem de indicadores para serem computados na árvore de ataque. Os indicadores adotados na análise foram os comportamentais. Os indicadores comportamentais são fatores os quais influenciam no comportamento da pessoa ou sujeito (AMENAZA, 2006). Os seguintes indicadores comportamentais adotadas na análise de risco foram:

- Perceptibilidade: realizar o ataque sem ser reconhecido ou notado pelo sistema;

- Habilidade técnica: experiência e conjunto de habilidades sobre assunto ou área de interesse que o atacante possui;
- Brecha de confiança: falha da entidade responsável por manter algo seguro e violação dos controles por parte do atacante.

Essa atribuição dos valores aos indicadores dos nós da árvore foi realizada de maneira empírica pelo conhecimento do autor sobre o assunto, entretanto, foi levada em consideração a complexidade de realizar o ataque e a experiência que o atacante precisaria para executar o ataque. Assim, as atribuições realizadas para perceptibilidade foram classificadas como baixa (0,3) e alta (0,7) e para conhecimento técnico foram classificadas como básico (30), intermediário (50) e avançado (70).

4.6.6 Caracterização do Agente da Ameaça

Além da atribuição dos indicadores comportamentais. outra tarefa necessária na análise de risco é a representação do agente de ameaça. O agente de ameaça representa a classe de pessoas que tem uma motivação para atacar o sistema (AMENAZA, 2006). Por outro lado, há três razões que levam o agente da ameaça a cometer o ataque, primeiro o sistema alvo a ser atacado tem vulnerabilidades e fraquezas; segundo, o agente da ameaça possui recursos disponíveis que fomentam o ataque; e terceiro, o agente de ameaça está motivado a realizar o ataque e acredita que o ataque lhe trará benefícios (AMENAZA, 2006). De qualquer forma, a Tabela 15 mostra os agentes de ameaças aplicados ao cenário de estudo. Na análise de risco sobre o cenário de investigação, os agentes de ameaças são representados pelo atacante amador e o atacante especialista.

Tabela 15 – Configuração dos perfis dos atacantes.

Perfil	Perceptibilidade	Habilidade Técnica	Brecha de Confiança
Atacante Amador	>=0.5	<=50	Sim
Atacante Especialista	<=0.5	<=70	Sim/Não

4.6.7 Poda da Árvore de Ataque

A podagem da árvore de ataque oferece ao defensor uma estimativa por

eliminação da magnitude do problema de segurança, ao considerar apenas os cenários de ataques que não estejam além da capacidade de um agente de ameaça (INGOLDSBY, 2013). Para podagem da árvore é aplicado o método de cálculo baseado em cenários de ataques que permite examinar todos os caminhos possíveis através nó raiz da árvore e calcular os valores dos nós folhas com base em cada caminho específico que pode ser assumido (AMENAZA, 2006). A partir dos resultados obtidos, Tabela 16, pode-se afirmar que, o fluxo com reforço da solução proposta reduz o cenário de exploração de ataque sobre os perfis dos atacantes em comparação ao fluxo normal:

- Os pontos percentuais de risco do modelo da árvore de ataque da proposta são em torno de 11% menor do que da proposta normal, quando o perfil do atacante é amador;
- Os pontos percentuais de risco do modelo da árvore de ataque da proposta são em torno de 16% menor do que à da proposta normal, quando o perfil do atacante é especialista.

Tabela 16 – Comparação na análise de risco entre as abordagens.

Árvore de Ameaça	Atacante	Cenários de Ataque	Percentual de Exploração de Ataque
Normal	Amador	10	↑ 10,00%
Normal	Especialista	36	↑ 13,88%
Reforço	Amador	9	↓ 11,11%
Reforço	Especialista	31	↓ 16,12%

4.6.8 Prioridade de Risco sobre Árvore de Ataque

A prioridade de risco sobre árvore de ataque levou em consideração ranquear as capacidades como perceptibilidade ou experiência atribuída aos ataques. Ataques com indicadores cujo valor é baixo possuem prioridades alta, já que são menos complexos e têm menor custo para o atacante. Para realizar a priorização de risco, são levados em consideração os indicadores atribuídos na árvore de ataque na seguinte ordem notabilidade, habilidade do ataque e brecha de segurança e o

perfil do atacante baseado nas capacidades.

Os resultados obtidos na priorização do risco de ataques mostram que o tipo de ataque mais recompensador para o atacante é o ataque de replicação no modelo da solução proposta, já que possui nível baixo para os indicadores de perceptibilidade e habilidade técnica. De acordo com o resultado da priorização do risco, as categorias de ataques identificadas com maior chance de serem realizadas foram replicação de autorização na requisição do *token* e não limitação do número de requisições ou do tempo de uso, Figura 89.

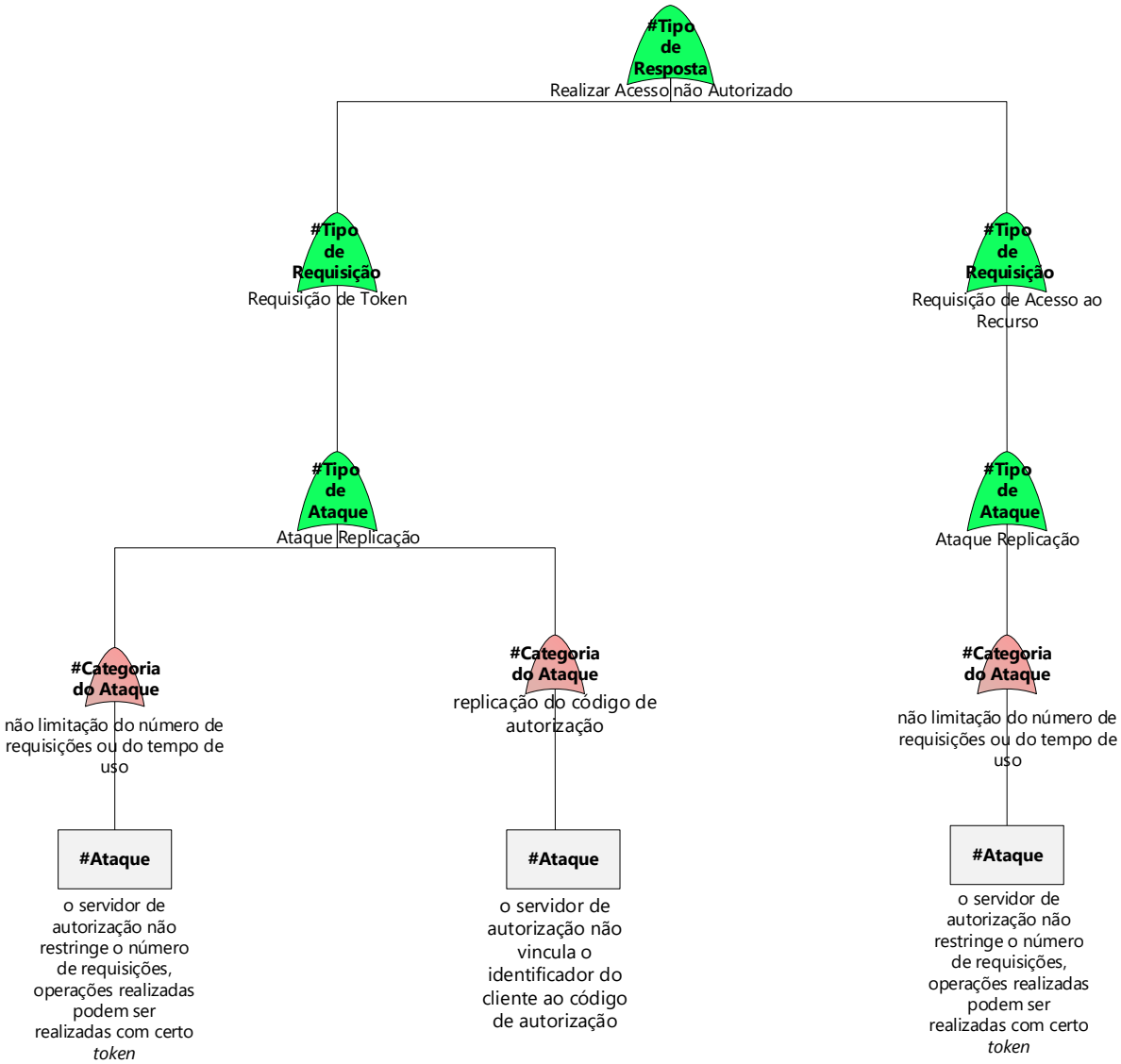

Figura 89 – Ataques atraentes para o atacante amador.

Já quando o perfil do atacante é especialista, o qual tem maior habilidade técnica e menor perceptibilidade, as três primeiras categorias de ataques com maior chance de serem realizadas no modelo da solução proposta foram as categorias de

ataques não utilizar requisições assinadas e não limitação do número de requisições ou do tempo de uso, Figura 90.

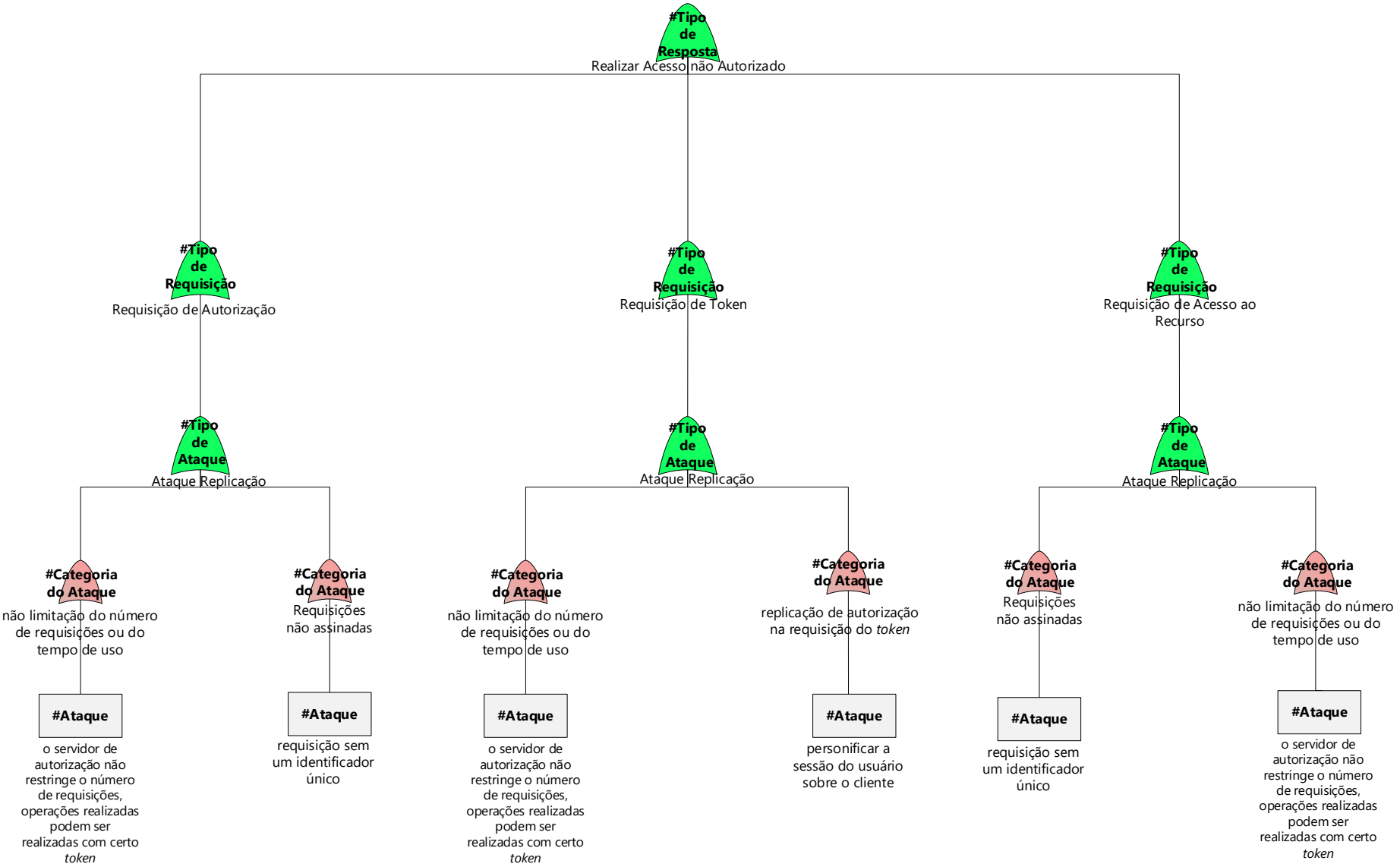

Figura 90 – Ataques de interesse para o atacante especialista.

4.7 IMPLEMENTAÇÃO

Dado o fato de que a solução proposta é uma extensão do ACE-OAuth, a implementação do sistema para solução proposta deste trabalho poderia reutilizar implementações existentes do ACE-OAuth. Entretanto, a implementação existente do ACE-OAuth foi desenvolvida por meio do ambiente JAVA que nos cenários de IoT tem um impacto no desempenho devido ao alto consumo de recursos, como a memória, por exemplo. Embora seja uma tecnologia que oferece ótima interoperabilidade entre os sistemas operacionais e tem forte adoção no desenvolvimento de aplicações Web e móvel.

Neste sentido, para que houvesse essa adaptação às restrições do ambiente de IoT, foi utilizada a linguagem de programação Python, uma linguagem de programação simples e dinâmica, de alto nível, dirigida para cenários experimentais, com várias bibliotecas de terceiros de código fonte aberto disponíveis dos protocolos investigados, além da possibilidade de embarcar o código-fonte em dispositivos de IoT. O código fonte[56], guia de configuração[57] e API RESTful[58] da implementação Python desenvolvidos podem ser encontrados no GitLab. Os fluxogramas das classes implementadas dos três *endpoints* (autorização, token, recurso) no servidor de autorização e no servidor de recurso se encontram no Apêndice A.

Estabelecimento do canal de segurança. Foi utilizado o DTLS para o estabelecimento de canal de segurança na comunicação entre o cliente e o servidor de recursos assim como entre o cliente e o servidor de autorização, já que o tráfego das credenciais deve ser protegido e é fundamental para a efetividade do controle de acesso ao recurso protegido. Para gerar as chaves criptográficas para o estabelecimento do canal de segurança, a ferramenta *openssl*[59] foi utilizada.

Artefatos. Como parte deste projeto, foi desenvolvido um conjunto de classes que suportam o perfil de comunicação e segurança CoAP/DTLS para garantir o controle de acesso aos recursos nas requisições do cliente no servidor de autorização e no servidor de recurso. É importante destacar que projeto da implementação da solução

[56] Código fonte: https://gitcin.cin.ufpe.br/ace-oidc-oauth/source-code
[57] Guia de Instalação: https://gitcin.cin.ufpe.br/ace-oidc-oauth/installation-guide
[58] Documentação da API: https://app.swaggerhub.com/apis-docs/IoT-Flows/ace-oidc-oauth/1.0.0
[59] *Cryptography and* SSL/TLS *Toolkit*: https://www.openssl.org/

proposta foi apoiado em algumas dependências de terceiros.

Dependências de Terceiros. Comumente no desenvolvimento de software, o reuso de bibliotecas de terceiros contribui para alcançar os requisitos do software, Figura 91. Os algoritmos de criptografia usados pela solução são fornecidos pela dependência *cryptography*[60] em Python. Toda a infraestrutura do protocolo CoAP foi construída sobre a dependência Python CoAPThon[61]. O estabelecimento do canal seguro foi realizado utilizando a dependência DTLS 1.2[62]. Para realizar a conversão de mensagens serializadas foi utilizada a dependência Python flunn[63]. E para adicionar da camada de segurança sobre as mensagens CBOR foi utilizada a dependência COSE-PYThon[64].

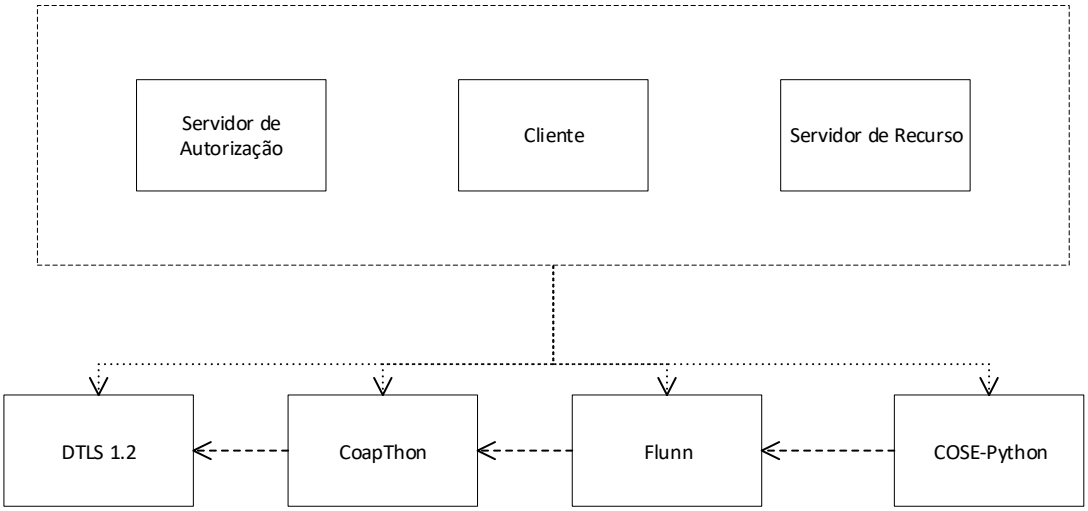

Figura 91 – Bibliotecas de terceiros utilizados na proposta ACE-OAuth.

Implementação. Foram desenvolvidas implementações protocolos utilizado no perfil CoAP/DTLS (Subseção 4.2.4) para todas as três entidades ACE-OAuth, servidor de autorização, servidor de recurso e cliente, Figura 92. Assim como, na solução, estão contidos a lógica do fluxo código de autorização, fluxo desafio-resposta e fluxo de veracidade e prova de posse.

[60] Dependência cryptography: https://github.com/pyca/cryptography
[61] Dependência CoAPthon: https://github.com/Tanganelli/CoAPthon
[62] Dependência pydtls: https://github.com/rbit/pydtls
[63] Dependência flunn: https://github.com/funny-falcon/flunn
[64] Dependência COSE-PYTHON: https://github.com/TimothyClaeys/COSE-PYTHON

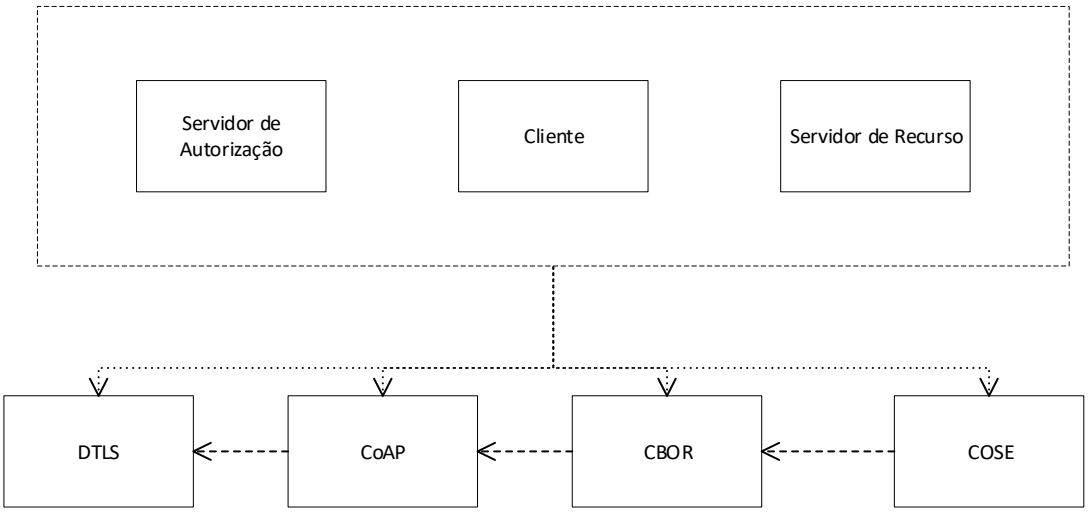

Figura 92 – Protocolos utilizados no perfil CoAP/DTLS para o ACE-OAuth.

CBOR. Por meio da dependência de terceiros flunn, foram modelados os objetos a partir do padrão CBOR. Isto incluiu a codificação e decodificação de mensagens serializadas.

CoAP. Por meio da dependência de terceiros CoAPThon, foram modelados os objetos a partir do padrão CoAP. Isto inclui, chamada a métodos da API RESTful.

COSE. Por meio da dependência de terceiros COSE-Python, foram modelados os objetos a partir do padrão COSE. Isto inclui criptografia, assinatura digital, código de autenticação da mensagem nos objetos CBOR. .

DTLS. Por meio da dependência de terceiros DTLS, foram modelados os objetos a partir do padrão DTLS. Isto inclui os esquemas criptográficos, esquemas de troca de chave e estabelecimento de canal de segurança e de sessão.

Implementação do tipo de resposta ID Token. A implementação da solução de reposta por meio da estrutura ID Token no fluxo de concessão do código de autorização da solução proposta permite verificar a identidade das requisições inspirado nas soluções de resposta do OpenID Connect.

Na Figura 93, é mostrada a estrutura interna dos módulos implementados e a relação com as entidades participantes do fluxo de concessão adotado na solução proposta. A solução proposta usa criptografia simétrica sobre a prova de posse a

partir objeto COSE. A chave simétrica utilizada pelo COSE para criptografar a prova de posse é compartilhada objeto os servidores de autorização e o servidor de recursos. O objeto COSE também pode ocultar ou esconder os dados da credencial de acesso e da credencial de identidade para assegurar a confidencialidade entre as requisições e resposta e dessa forma dificultar investidas de monitoração e de interceptação.

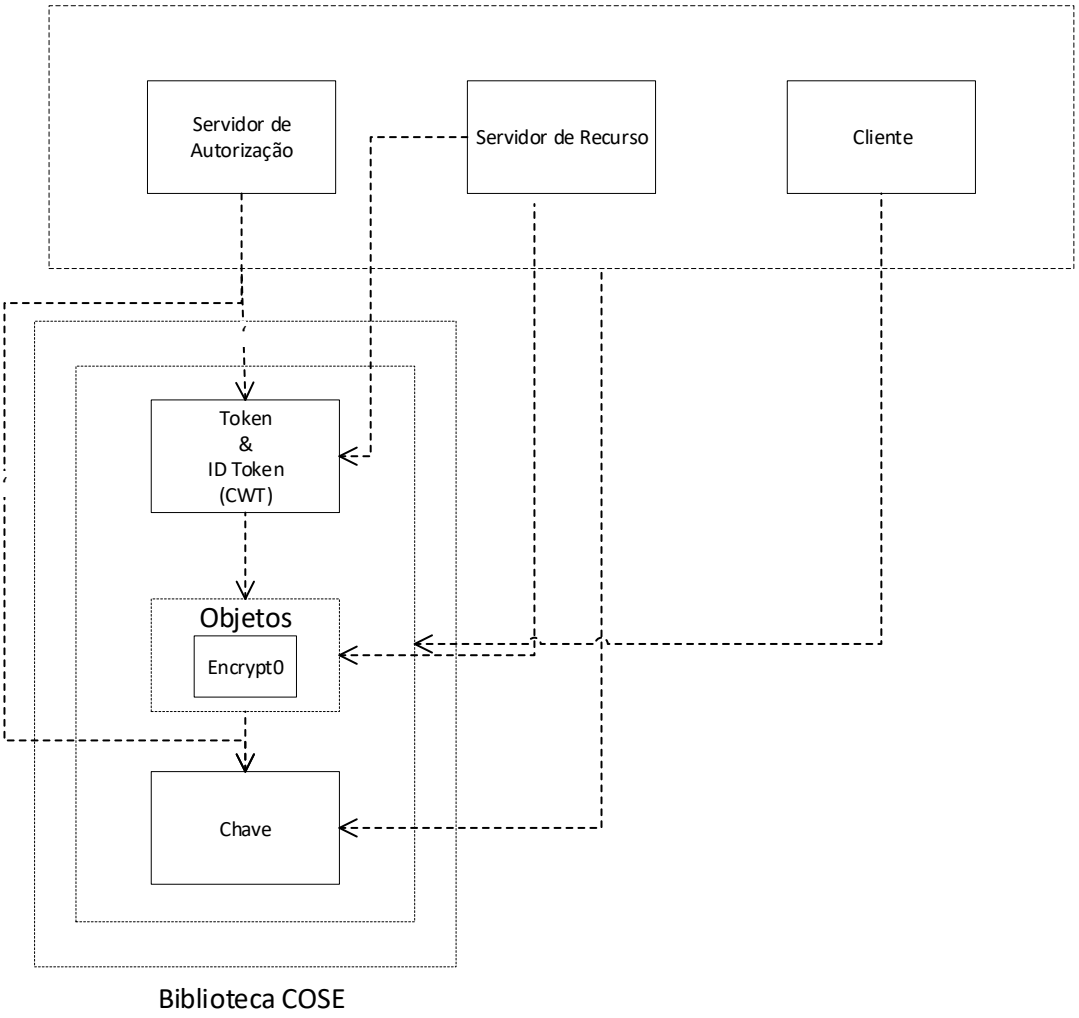

Figura 93 – Estrutura da mensagem que encapsula as credenciais.

5 Avaliação de Desempenho

“A vida é muito curta para ser gasta nutrindo animosidade ou registrando erros.”

Charlotte Bronte.

Neste capítulo, é apresentada a avaliação de desempenho sobre a solução proposta com o objetivo de verificar o impacto na comunicação entre as partes envolvidas, ao considerar os mecanismos de segurança e a plataformas de ambiente. Em relação à plataforma, foram experimentados cenários como locais e em nuvem. Já em relação aos mecanismos de segurança foram configurados com e sem estabelecimento canal de segurança. Além disso, foi adotado no projeto de experimento fatorial 2^k com objetivo de observar quais fatores que mais impactam no resultado em relação ao tempo.

5.1 SISTEMA INVESTIGADO

Esta seção trata dos cenários assumidos pelo sistema, a justificativa para tal escolha do ambiente e os objetivos delineados para essa avaliação de desempenho ocorreram a fim de entender o efeito da combinação dos cenários adotados pelo sistema.

5.1.1 Cenários

Os cenários que compõem o ambiente da avaliação de desempenho teve maior impacto de implantação no lado servidor, com a opção da plataforma servidora local ou plataforma servidora em nuvem. A plataforma local, os servidores de autorização e de recursos se encontram na rede local mais próximo das aplicações clientes e o ambiente pode ter sua distribuição centralizada ou descentralizada, Figura 94.

Figura 94 – Cenário local.

A Figura 95 mostra a plataforma em nuvem onde os servidores de autorização e de recurso estão instanciados em máquinas virtuais em uma ambiente nuvem. Na plataforma em nuvem, os servidores de autorização e de recursos se encontram na rede local mais próximo das aplicações clientes e o ambiente pode ter sua distribuição centralizada ou descentralizada.

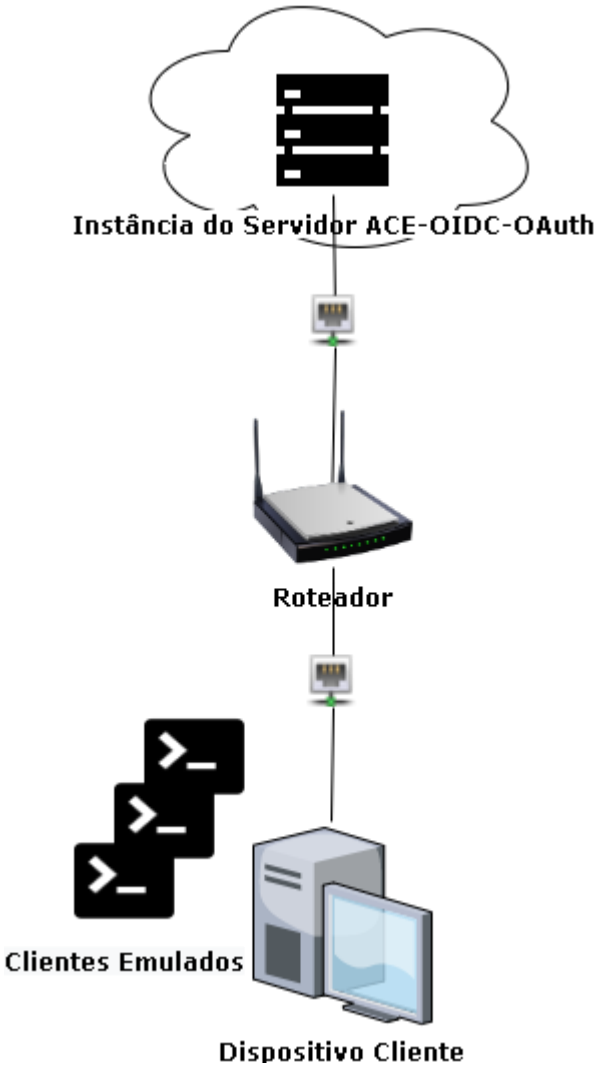

Figura 95 – Cenário em nuvem.

5.1.2 Justificativa

No primeiro cenário investigado, a plataforma em nuvem é caracterizada pela depedência de conexão externa, por meio conectividade com a Internet, e suscetibilidade a queda de comunicação. Além disso, violações de segurança podem revelar dados sensoriados ou registros de contas do usuário que são trafegados para nuvem (TSCHOFENIG, 2016). Além do mais, na plataforma em nuvem, é esperado alta demanda por recursos, maior capacidade de processamento. a plataforma em nuvem o servidor fica mais distante do seu uso e o servidor é hospedado em ambiente ou descentralizado.

Por outro lado, no segundo cenário investigado, a plataforma local não há depedência de conexão externa, e tem uma menor sensibilidade a perca de comunicação. Além do que, a demanda não é tão alta em comparação a plataforma em nuvme.sem contar que a plataforma local é uma das recomendações do *framework* de segurança de IoT (VERMESAN; FRIESS, 2013)

Ambas as plataformas local e em nuvem foram configuradas com estabelecimento de canal de estabelecimento e sem estabelecimento de canal de estabelecimento.

Já no lado cliente a aplicação é representada pelos dispositivos móveis (*smarthphone* e *tablets*), já que comumente atuam como controladores no ambiente de automação residencial de IoT. Desse jeito, simulação da comunicação permite que o cliente solicite os serviços oferecidos pelo servidor na plataforma local ou remoto, com e sem o canal de segurança.

5.1.3 Objetivos

O objetivo principal da avaliação de desempenho é verificar qual a combinação de cenários é mais eficiente sobre o tempo em relação à escolha da plataforma do ambiente e do estabelecimento de canal de segurança. Diante disso, baseado no objetivo principal, os subobjetivos foram definidos, que são:

- Ter conhecimento do impacto no tempo de resposta da aplicação que solicita o serviço;
- Quais os fatores principais e de interações têm impacto considerável na métrica do tempo de resposta;
- Conhecer o impacto no tamanho das mensagens de requisição e de resposta.

5.2 SERVIÇOS E MÉTRICAS

Nesta seção são descritos os serviços fornecidos pelo sistema e as métricas adotadas definidas para avaliação de desempenho.

5.2.1 Serviços

Os serviços oferecidos pelo servidor são acessados pelo cliente por meio das

requisições de autorização, de *token*, de acesso aos recursos direcionados aos respectivos *endpoints* (*authz*, *token*, *well-known*) via ao paradigma REST. Desse modo foi possível simular a comunicação entre o cliente e o servidor e realizar a medição do sistema. A Figura 96 mostra *os endpoints* utilizados que fornecem serviços ao cliente.

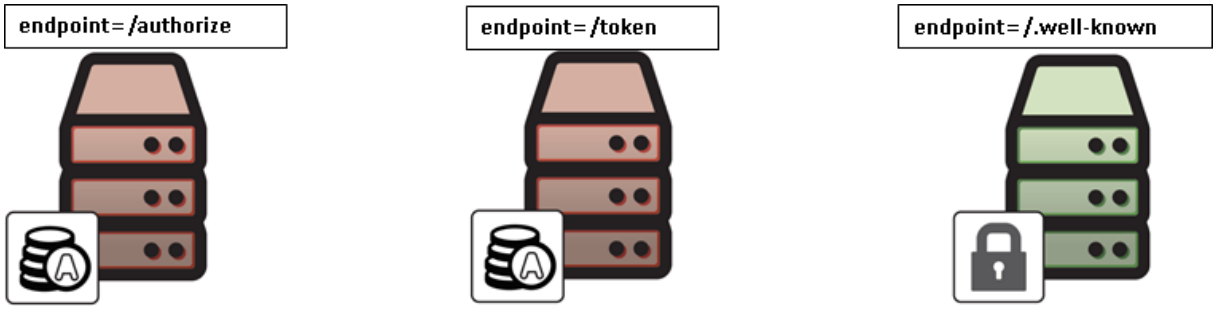

Figura 96 – *Endpoints* disponibilizados pelo cliente na emissão das credenciais e no acesso aos recursos.

5.2.2 Métricas

As métricas definidas para a avaliação de desempenho na comunicação entre as partes do sistema são mostradas na Tabela 17. Nessa situação, a métrica, tamanho da mensagem, permite conhecer qual é impacto na codificação dos dados nos fluxos de mensagens compostos pela solicitação aos *endpoints*. Com essa informação é possível analisar como tamanho da mensagem pode afetar comunicação, já que para ambientes restritos, o gasto de recurso por excesso. Já a métrica relacionada ao tempo identifica o quanto é rápido ou demorado é o sistema para atender as solicitações dos clientes.

Dessa maneira, as escolhas dessas métricas são justificadas, pois, primeiro, elas são comumente utilizadas nos experimentos dos trabalhos relacionados e segundo são apontadas para revelar gargalos no sistema.

Tabela 17 – Métricas adotadas na avaliação de desempenho

Métricas	Cliente	Servidor
Número médio de *bytes* por transação	Requisição	Resposta
Tempo Médio	Latência	Tempo de atendimento

5.3 PARÂMETROS DO SISTEMA E DA CARGA

Nesta seção são descritos os parâmetros do sistema e os parâmetros de carga assim como os fatores e os níveis utilizados na avaliação do sistema.

5.3.1 Parâmetros do Sistema

Os parâmetros do sistema adotados no projeto de avaliação de desempenho foram modelo arquitetura, frequência e quantidade de núcleos do processador, memória, tecnologia de *link* de conectividade, modo de operação da rede, canal operação da rede, sistema operacional, versão do *kernel* e disco.

5.3.2 Parâmetros de Carga

Os parâmetros de carga adotados no projeto de avaliação de desempenho sobre são o tempo de execução, o número de clientes simultâneos emulados, o número de fluxos das requisições por cliente, o intervalo entre os fluxos das requisições, a função de distribuição do intervalo entre as requisições, o conjunto das requisições do fluxo, o canal de segurança e a plataforma do ambiente.

5.3.3 Fatores e Níveis

A Tabela 18 mostra os fatores e níveis adotados nos experimentos da avaliação de desempenho em função dos objetivos. Esses fatores são levados em consideração, pois de acordo com sua variação, podem impactar no desempenho no sistema.

Desse modo, no caso do fator canal de segurança, o estabelecimento do canal de segurança implica a princípio em um aumento computacional ou maior tempo de recurso, o objetivo é assegurar que a configuração do canal de segurança adotado pelo sistema alivie a sobrecarga que já é conhecida.

Já em relação ao fator plataforma do ambiente, a variação de onde o sistema é alocado e o meio de comunicação utilizado também afetam no desempenho do sistema. Desse modo, o objetivo é avaliar em que situação é mais interessante usar a plataforma do ambiente para que se tenha menor interferência na comunicação e, consequentemente, um menor atraso.

Tabela 18 – Fatores e níveis

Fatores	Níveis
Canal de Segurança	Sem Canal de Segurança (*Basic*), Com Canal de Segurança (DTLS)
Plataforma do Ambiente	Local (*wireless*), Remoto (*wired*)

5.4 PROJETO DE EXPERIMENTOS

Em se tratando do projeto de experimentos 2^K, deseja-se saber se a variação entre os dois níveis de um ou mais fatores causa um impacto relevante na resposta (CARPINETTI, 2009). Para identificar a variação de um ou mais fatores dois tipos de efeitos são analisados: efeito principal e efeito de interação. Para isso, foi adotado para os experimentos um projeto fatorial completo 2^2, com replicações.

A Tabela 19 mostra os fatores e níveis escolhidos para a avaliação preliminar do projeto fatorial 2^K (k=2), onde k é o número de fatores. Para cada fator são atribuídos o nível baixo e o nível alto. A análise e avaliação dos resultados para projeto de experimento 2^K são observadas na Subseção Tabela 19.

Tabela 19 – Fatores e níveis para o experimento fatorial 2^k.

Fator	Nível Baixo (-1)	Nível Alto(+1)
Canal Seguro	Sem segurança (Base)	Com Segurança (DTLS)
Ambiente	Local (*wireless*)	Nuvem (*wireless/wired*)

5.5 TÉCNICA DE AVALIAÇÃO

A técnica de avaliação adotada nos experimentos foi a medição. No contexto de medição, geralmente uma parte do sistema é selecionada para realizar a avaliação de desempenho, já que a aplicação da técnica de medição no sistema é complexa e custosa.

5.6 CENÁRIOS UTILIZADOS PARA OS EXPERIMENTOS

A Tabela 20 mostra a combinação dos cenários sobre a solução proposta do sistema que são baseados na variação dos fatores do experimento e que foi submetida à avaliação do desempenho.

Tabela 20 – Cenários utilizados nos experimentos.

Cenários	Plataforma	Canal de Segurança
LAN-Basic	Local	Sem
LAN-DTLS	Local	Com
Remote-Basic	Remoto	Sem
Remote-DTLS	Remoto	Com

Vale destacar que, quando a plataforma é em nuvem, o servidor assume o papel de uma máquina virtual hospedada na Azure[65]. Entretanto, diferentemente de um servidor físico ou de um *datacenter*, por exemplo, que possui grande capacidade de recursos, quando a plataforma do ambiente é situada em nuvem, os recursos são compartilhados e apenas uma instância do sistema é executada. É importante desatacar que, do mesmo modo como aconteceu na nuvem, quando a plataforma do ambiente foi local, o sistema executou também com capacidade reduzida para realizar os experimentos com os mesmos critérios a partir da nivelação dos parâmetros do sistema parâmetros do sistema adotados no projeto de avaliação de desempenho na Subseção 5.3.1. No lado cliente, porém, os clientes foram emulados em um dispositivo *desktop.* Por fim, a Tabela 21 mostra os valores específicos dos componentes ou parâmetros do sistema.

[65]Portal do Azure: https://portal.azure.com

Tabela 21 – Configuração dos dispositivos

Configuração	Processador	Memória	Conectividade	Sistema Operacional	Disco
Servidor (Local)	1 núcleo, 897 MHz, x86, Intel Celeron M	1GB	Wi-Fi Modo da Rede (misto 11 g/b/n) Canal (autosselecionável)	Ubuntu 16.04.6 LTS,4.13.0-46, 32 *bits*	160 GB
Servidor (Remoto)	1 núcleo, 2.40 GHz, X86_64, Intel(R) Xeon(R) CPU E5-2673 v3	1 GB	Ethernet (100 mb/s)	Ubuntu 16.04.6 LTS, 32 *bits*	40 GB
Cliente	2 núcleos, 3.20 GHz, X64, Intel Core i5 650	4 GB	Ethernet (100 mb/s)/ Wi-Fi Modo da Rede (misto 11 g/b/n) Canal (autosselecionável)	Windows 10, 64 *bits*	464 GB

A Tabela 22 mostra os parâmetros de carga para ser submetido ao sistema para avaliação de desempenho. Observa-se que a função de distribuição adotada nos experimentos foi exponencial, pois ela verifica a probabilidade do tempo transcorrido entre duas ocorrências consecutivas, dada uma variável contínua aleatória que conta o número de ocorrências em um dado intervalo de tempo (ZIBETTI, 2016).

Tabela 22 – Parâmetros de carga aplicados ao sistema.

Tempo de Execução	3 minutos
Número de Clientes Emulados Simultâneos	30 clientes
Número de Fluxos de Requisições por Cliente	10 requisições
Intervalo entre os Fluxos das Requisições	2 s
Função de Distribuição do Intervalo entre Requisições	Exponencial
Conjunto de Requisições do Fluxo	Solicitação de autorização (/authz), solicitação de token (/token) e solicitação de recurso (/.well-know)
Canal de Segurança	Com e sem canal de segurança
Plataforma do Ambiente	Local e remoto

5.7 ANÁLISE DOS DADOS E AVALIAÇÃO DOS RESULTADOS

A análise dos dados se baseou nos resultados obtidos por meio do resultado da medição das métricas definidas na Subseção 5.2.2 como o número médio de bytes por transação (o tamanho das mensagens nas requisições do cliente e nas respostas pelo servidor) e no tempo médio (latência no lado cliente e tempo de atendimento no lado servidor). Já a avaliação dos resultados foi feita a partir da comparação entre cenários apresentados na Subseção 5.6.

5.7.1 Tamanho das Mensagens

Para captar as mensagens foi usado o analisador de pacotes *wireshark*[66] com a intenção de avaliar o impacto associado ao tamanho da mensagem CBOR em contraste ao JSON.

Para investigar o impacto da serialização dos dados, o tamanho das mensagens na requisição e na resposta codificadas em CBOR usando CoAP e em JSON usando HTTP foi comparado. Os resultados são sumarizados na Tabela 23.

[66] Ferramenta de captura de pacotes (Wireshark): https://www.wireshark.org

Tabela 23 – Comparativo entre tamanho da mensagem do HTTP e CoAP.

Mensagem	Tamanho do Conteúdo JSON	Tamanho do Conteúdo CBOR	Diferença entre JSON x CBOR	Tamanho da Mensagem no HTTP	Tamanho da Mensagem CoAP	Diferença entre HTTP e CoAP
Requisição de Autorização	255	264	↑9,00	475	323	↓152,00
Resposta de Autorização	135	224	↑89,00	499	273	↓226,00
Requisição de Token	444	414	↓30,00	683	469	↓214,00
Resposta do Token	323	664	↑341,00	674	713	↑39,00*
Requisição do Recurso	469	646	↓177,00	714	707	↓7,00
Resposta do Recurso	41	32	↓9,00	390	81	↓309,00
Somatório				3423	2566	↓908,00
Diferença do tamanho da mensagem CoAP em relação ao HTTP (%)				$\sum(HTTTP - CoAP) / \sum(HTTP)$		26,433%
Tamanho da mensagem que ocupa tanto em HTTP e quanto em CoAP (%)				100% – 26,433%		73,566%

Para cada mensagem trocada durante cada requisição do fluxo proposto, foi medido o tamanho da mensagem da estrutura CBOR, assim como o tamanho da mensagem CoAP. Os resultados preliminares demonstram que menos da metade de *bytes* trocados entre as entidades pode ser atribuído ao CoAP. Desse modo, então se concluiu que CBOR afeta consideravelmente no tamanho da mensagem.

Além do mais, de acordo os resultados preliminares na Tabela 23, ao se comparar o tamanho do conteúdo entre JSON e CBOR, percebe-se que a

mensagem CBOR impacta muito mais do que o JSON no tamanho da mensagem, mesmo o CBOR usando codificação serializada. Entretanto, ao se comparar o tamanho da mensagem entre no formato JSON transferido sobre HTTP e no formato CBOR transferido sobre CoAP, percebe-se que o HTTP tem uma influência significativa no tamanho final da mensagem muito mais do que CoAP, isso devido ao HTTP utilizar uma série de cabeçalhos adicionais nas mensagens. Dessa forma, conclui-se que, o uso do CoAP com CBOR reduz o impacto no tamanho da mensagem em 26,433% em comparação HTTP com JSON.

5.7.2 Latência

A Figura 97 mostra o tempo de latência médio na execução dos experimentos sobre o conjunto de requisições do fluxo proposto.

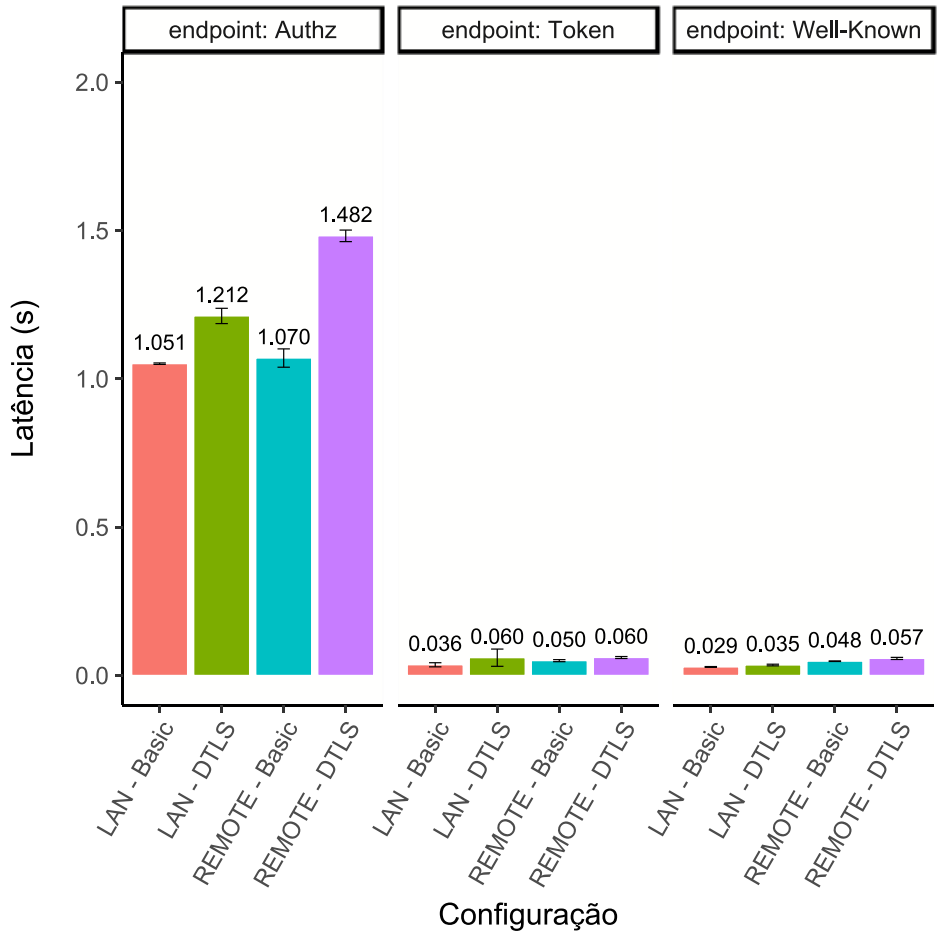

Figura 97 – Latência.

Na Figura 97, nota-se que, quando o fator de canal de segurança foi considerado e o nível variou de [sem canal de segurança] para [com canal de

segurança], a latência aumentou. Do mesmo modo, constatou-se que, quando o fator plataforma variou [local] para [remoto], houve incremento na latência.

Dessa forma, o acréscimo no tempo de atendimento apresentou uma margem visualmente muito próxima, sendo necessário verificar se apresenta uma diferença significativa entre os mesmos. Para confirmar a hipótese de quão diferente são os valores assumidos na variação dos fatores, foi realizado o *test*-t.

O resultado obtido do *test*-t mostrou que, tanto na variação do fator plataforma quanto na variação do fator canal de segurança, houve diferença não significativa, quando se considerou o mesmo nível de confiança. Essa diferença não significativa pode ter decorrido de uma situação particular de proximidade do serviço em nuvem e da subutilização dos recursos envolvidos.

5.7.3 Tempo de Atendimento

O tempo de atendimento no lado servidor consiste no tempo computado desde o recebimento da requisição do cliente, passando pelo seu processamento, que pode ser finalizado ou não, até o envio da resposta para o cliente. A Figura 98 mostra o tempo de atendimento médio para o conjunto de requisições do fluxo.

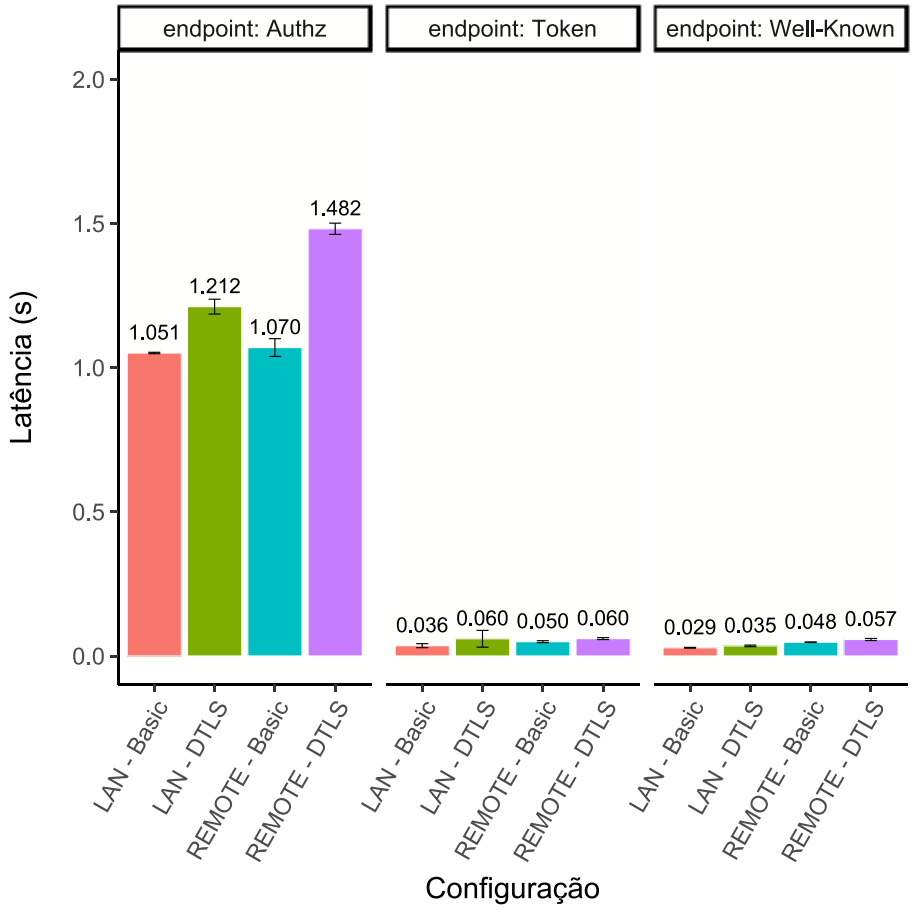

Figura 98 – Tempo de atendimento médio pelo o servidor

Na Figura 98 nota-se que quando o fator de canal de segurança é considerado, o nível variou de sem canal de segurança para com canal de segurança, a latência teve um aumento considerável quando o ambiente foi local. No entanto, quando o fator de canal de segurança foi considerado em um ambiente remoto, a latência teve um pequeno aumento em comparação ao ambiente local. Já quando o fator plataforma é considerado, o tempo de atendimento diminuiu quando variou de local para remoto. Essa diminuição no tempo de atendimento pode ser justificada pelo fato de que o servidor remoto tem maior capacidade de processamento (2,40 GHz) em relação ao servidor local (897 MHz).

Dessa forma, o acréscimo no tempo de atendimento apresentou uma margem visualmente muito próxima, apesar de que o tempo de atendimento da plataforma em nuvem é um pouco mais rápido do que a plataforma local, mas não apresentou uma diferença significativa do ponto de vista dos valores assumidos. Para confirmar a hipótese de quão diferentes são os valores assumidos na variação dos fatores, foi realizado o *test*-t.

O resultado obtido do *test*-t mostrou, tanto na variação do fator ambiente quanto na variação do fator canal de segurança, que houve diferença não significativa, quando se considera o mesmo nível de confiança. Essa diferença não significativa pode ter decorrido de uma situação particular de proximidade do serviço em nuvem, meio de comunicação utilizado, configuração dos componentes dos dispositivos e da subutilização dos recursos envolvidos.

5.7.4 Impacto dos Fatores

Nesta subseção são mostrados os resultados da análise do efeito principal e do efeito de interação na avaliação do projeto de experimentos 2^k.

Efeito Principal. O efeito principal de um fator é dado pela variação média da resposta causada pela variação do fator de -1 para +1 (CARPINETTI, 2009).

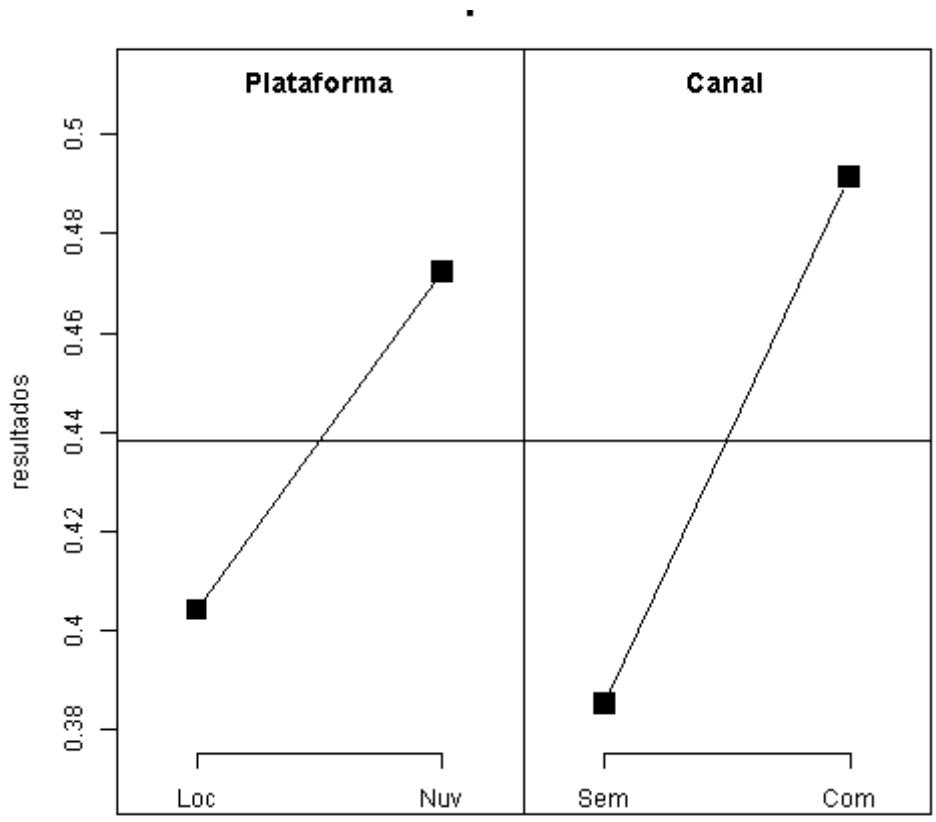

Figura 99 – Efeito principal sobre a latência.

A Figura 99 mostra a variação da latência média sobre os testes dos efeitos principais. A diferença entre os valores da plataforma [local, nuvem] e canal de segurança [com, sem] corresponde ao efeito principal A diferença entre os valores

da plataforma [local, nuvem] e canal de segurança [com, sem] corresponde ao efeito principal. Na Figura 99 a subida da reta sinalizou, para efeito principal da plataforma, o aumento no impacto sobre a latência quando variou de local para nuvem. Já quando o efeito principal do canal de segurança é analisado, a subida da reta sinalizou o aumento no impacto sobre a latência, quando variou de sem canal de segurança para com canal de segurança.

Efeito de Interação. Identifica o quão representativo é o impacto nos resultados para as métricas consideradas na variação dos fatores devida à interação e a influência entre eles.

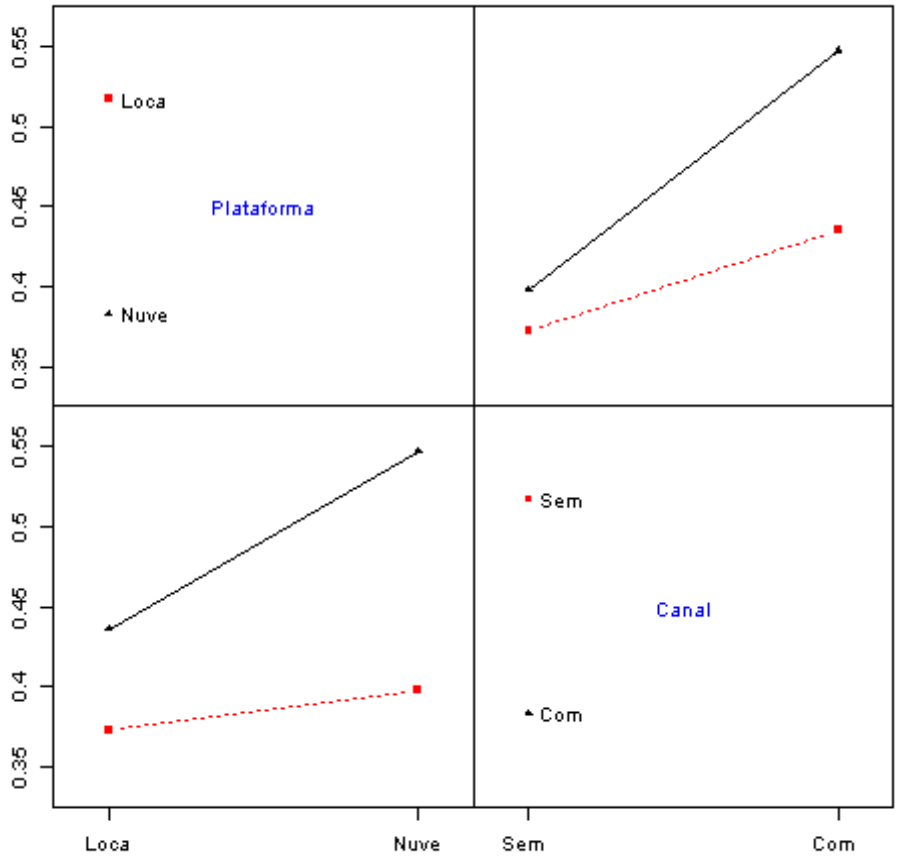

Figura 100 – Efeito de interação sobre a latência.

A Figura 100 mostra os resultados do efeito de interação sobre resultado da latência. Para realizar os testes do efeito de interação observa-se o quanto a variação do efeito principal do primeiro fator aumentou ou diminuiu sobre o resultado medido em função da variação do efeito principal do segundo fator, representado pela subida ou pela descida da reta. A diferença entre as retas consiste no efeito de interação que indica quanto o efeito principal do segundo fator influenciou o

comportamento do efeito do principal do primeiro fator sobre a variável medida.

Dessa forma, os resultados dos testes do efeito de interação, constataram que o efeito principal do primeiro fator (canal de segurança) no resultado da latência foi influenciado pelo efeito principal do segundo fator (plataforma) quando variou de local para a nuvem e que o efeito principal do primeiro fator (plataforma) no resultado da latência foi influenciado pelo o efeito principal do segundo fator (canal de segurança) quando variou de sem estabelecimento de canal segurança para com estabelecimento de canal de segurança.

6 Discussão e Conclusões

“Por mais que a vida pareça difícil, há sempre algo que você pode fazer e em que pode ter sucesso.”

Stephen Hawking.

Neste trabalho foi proposta uma solução para o reforço do controle de acesso em um ambiente IoT. A solução proposta visa mitigar, principalmente, ataques de personificação e de replicação no fluxo de concessão código de autorização utilizado no ACE-OAuth. Como contramedidas a esses ataques, foi desenvolvida uma solução API RESTful, por meio do perfil de comunicação e de segurança CoAP/DTLS em conjunto com os fluxos de desafio-resposta e de veracidade de identificação.

Na avaliação de segurança sobre a solução proposta, houve redução em 30 pontos percentuais de não conformidade a ameaças em relação ao fluxo original e na avaliação de risco, houve diminuição dos riscos associados ao ataque do perfil amador em cerca de 11 pontos percentuais e os riscos associados ao perfil do especialista em torno de 16 pontos percentuais.

Já na avaliação de desempenho, os fatores plataforma da solução e estabelecimento de canal de segurança foram avaliados para verificar o impacto no tempo de resposta do sistema da solução proposta. Resultados preliminares mostraram que o tempo de resposta médio da solicitação do cliente ficou abaixo de 2s em média. Já em relação ao tamanho das mensagens das requisições e das respostas no fluxo, o CoAP reduziu em cerca de 74%, em média, o tamanho da mensagem em comparação ao HTTP

Por fim, são destacadas as principais contribuições deste trabalho e as propostas de trabalhos futuros.

6.1 CONTRIBUIÇÕES

A seguir estão apresentadas as contribuições realizadas e potenciais deste trabalho.

ACE-OAuth. A implementação da solução proposta pode servir como modelo/base para o desenvolvimento de outras soluções baseadas no perfil de comunicação e de segurança CoAP/DTLS para os domínios de IoT em que a especificação ACE-OAuth seja aplicável como, por exemplo, o da automação residencial.

Segurança. Esse trabalho contribui no reforço ao controle de acesso contra os ataques de personificação e replicação. Além disso, enfatizou as preocupações quanto ao ambiente IoT, no cenário de automação residencial, relacionado à autenticação, à autorização e à gestão de identidade.

Pesquisa. Este trabalho contribui no avanço do estado da arte sobre o problema investigado do reforço de segurança no controle de acesso em IoT. Além disso, foram esclarecidos os aspectos de segurança e, também, foram endereçados os desafios no uso de credenciais e no controle de acesso aos recursos sobre as aplicações de terceiros. Além disso, a implementação da solução pode ser reaproveitada por pesquisadores nessa linha de investigação para início do desenvolvimento de suas próprias API e para fins de comparação na avaliação de desempenho e na avaliação de segurança.

6.2 TRABALHOS FUTUROS

A seguir foram enumeradas sugestões de trabalhos futuros como desdobramentos ou evolução deste trabalho:

Registro dinâmico da credencial. O processo de registro dinâmico da credencial no controle de acesso é fundamental para mitigar ameaças e ataques de personificação e replicação. No Apêndice A são apresentados dois diagramas de sequência que mostram o processo do registro dinâmico da credencial proposto para evolução deste trabalho que faz uso de *tokens* e de QrCode e que podem servir como referência nas investigações e propostas futuras. Ademais, no trabalho de Echeverría et al. (2019) encontra-se a implementação de uma solução alternativa do registro dinâmico no controle de acesso de IoT. Contudo, para aumentar a eficácia do uso do registro dinâmico da credencial na mitigação contra as ameaças sobre ambiente em IoT, pode-se agregar abordagens de gerenciamento de acesso e de

identidade através de soluções existentes como Open ID Connect, autenticação com segundo fator (M2A) ou FAST *Identity Online* (FIDO), esta última solução abordada no trabalho de Fremantle & Aziz (2018).

Fluxo de concessão. Como o foco deste trabalho se dirigiu, exclusivamente, ao fluxo de concessão código de autorização, ainda é preciso avançar na investigação para outros tipos de fluxos como, implícito, credenciais de cliente e credenciais dono do recurso, já que cada solução em IoT pode exigir um fluxo específico a partir de requisitos próprios em função do ambiente ou restrições impostas, além do contexto da aplicação, como por exemplo, na monitoração de saúde pessoal no ambiente hospitalar.

Autenticação e integridade. Destacam-se, para o aprofundamento no estudo sobre a segurança de comunicação e de autenticação em IoT, aspectos como autenticação mútua, segurança do conteúdo objeto ou do recurso protegido, autorização e controle de acesso descentralizado, distribuição e gerenciamento de chaves, criptografia, dentre outros (TSCHOFENIG; BACCELLI, 2019). Para atender os aspectos de integridade e de autenticação, recomenda-se o uso de assinatura digital[67] ou MAC[68] na mensagem (SCHAAD, J; CELLARS, 2017), autenticação de mensagem baseada em chave *hash* (KRAWCZYK; BELLARE; CANETTI, 1997), esquema de troca de chaves e derivação de chave usando algoritmo Diffie-Hellman[69] (SELANDER; MATTSSON; PALOMBINI, 2019) para reforçar as fases de geração da assinatura e verificação das mensagens trocadas entre o cliente e servidor para fins de confirmação da origem no processo de autenticação. Já no trabalho de Lagutin et al. (2019) são usados identificadores descentralizados que independem de autoridade central que gerencia a identidade e credenciais verificáveis que provam o direito de uso de algum serviço.

Controle de Acesso. No trabalho de EL-HAJJ et al. (2019) é realizado um levantamento sobre os mecanismos de controle de acesso utilizados nas plataformas de IoT, em sua grande maioria em nuvem, sobre as aplicações de

[67] *Signature Algorithms*: https://tools.ietf.org/html/rfc8152#section-8
[68] MAC *Objects*: https://tools.ietf.org/html/rfc8152#section-6
[69] *Ephemeral Diffie-Hellman Over* COSE (EDHOC): https://tools.ietf.org/html/draft-selander-ace-cose-ecdhe-14

terceiros. Neste estudo foram destacados mecanismos como criptografia assimétrica, *gateways* seguros, assinatura digital baseada em certificado, sincronização nos esquemas de troca de chave, protocolo remoto e gerenciamento de acesso e de identidades. Esses mecanismos apresentados podem servir como gatilho nas investigações e propostas futuras para reforçar a segurança do ACE-OAuth.

Modelo do fluxo de comunicação e árvore de decisão de ataques. Para alcançar um nível de maturidade nos resultados adquiridos a partir da avaliação de não conformidade a ameaças e da avaliação de risco, é necessário desenvolver modelos do fluxo de comunicação dos dados e da árvore de decisão de ataques mais robustos para que contramedidas de contenção sejam eficazes e efetivas no combate as ameaças e os ataques tratados neste trabalho. Além disso, é necessário aplicar restrições e condições bem fundamentadas na avaliação de não conformidade a ameaças e definir pesos com critérios mais objetivos aos ataques na avaliação de risco. Para isso, exige-se experiência e habilidade específica do domínio pelo modelador. Contudo, a implantação de sistema de detecção de invasão ou teste de invasão podem confirmar ou refutar se contramedidas elaboradas a partir da avaliação de não conformidade a ameaças e avaliação de riscos são eficazes na mitigação das ameaças e dos ataques.

Experimentos e tecnologias em IoT. As experimentações realizadas em um *testbed* com uma melhor infraestrutura de IoT com uso de tecnologias mais recentes como WSN, IPv6, 6LowPan IEEE 802.15.4 (DEEP; ZHENG; HAMEY, 2019) são mais adequadas para realizar uma avaliação comparativa das soluções de IoT sobre aspectos como escalabilidade, interoperabilidade e desempenho.

Investigação dos casos de uso para autenticação e autorização em IoT. O desenvolvimento de soluções de IoT precisa estar ciente dos casos de uso[70] que permeiam os domínios de IoT, como automação residencial, monitoração de saúde pessoal, medidores inteligente e sistemas de controle residencial para verificar se aplicação de mecanismos de segurança já existentes satisfazem os casos de uso

[70] *Use Cases for Authentication and Authorization in Constrained Environments* https://tools.ietf.org/html/rfc7744

desses domínios ou se há necessidade de adaptação a eles, caso contrário seria necessária a criação de novos mecanismos de segurança. Por exemplo, na automação residencial há casos em que a comunicação é direta entre o cliente e o dispositivo de IoT, sem a participação da autoridade central de autorização para fins de autorização (HOVSMITH, 2017). Nessa ocasião, se o controle de acesso for realizado pelo ACE-OAuth, o fluxo de comunicação tem que ser modificado para atender esse caso de uso, já que o ACE-OAuth traz em sua essência o papel centralizado do servidor de autorização para realizar processo de autorização. Como no trabalho de Jung & Jung (2017) em que os autores propõem uma adaptação do OAuth para IoT em que a lógica de autorização é delegada para o lado cliente e ocorre de maneira descentralizada.

REFERÊNCIAS

ABOMHARA, Mohamed; KØIEN, Geir M. **Security and Privacy in the Internet of Things: Current Status and Open Issues**, 2014. Disponível em: <https://ieeexplore.ieee.org/document/6970594>

AKAMAI. **Financial Services — Hostile Takeover Attempts**, 2020. Disponível em: <https://www.akamai.com/us/en/multimedia/documents/state-of-the-internet/soti-security-financial-services-hostile-takeover-attempts-report-2020.pdf>

ALI, Bako; AWAD, Ali Ismail. Cyber and physical security vulnerability assessment for IoT-based smart homes. **Sensors (Switzerland)**, [s. l.], v. 18, n. 3, p. 1–17, 2018.

ALI, Waqar et al. IoT based smart home: Security challenges, security requirements and solutions. **ICAC 2017 - 2017 23rd IEEE International Conference on Automation and Computing: Addressing Global Challenges through Automation and Computing**, [s. l.], n. September, p. 7–8, 2017.

AMENAZA. **Understanding Risk Through Attack Tree Analysis**, 2003. Disponível em: <https://www.amenaza.com/downloads/docs/Methodology.pdf>

AMENAZA. **BurgleHouse Tutorial**, 2006. Disponível em: <https://www.amenaza.com/downloads/docs/Tutorial.pdf>

ANGRISHI, Kishore. Turning Internet of Things(IoT) into Internet of Vulnerabilities (IoV) : IoT Botnets. [s. l.], p. 1–17, 2017. Disponível em: <http://arxiv.org/abs/1702.03681>

ARAGON et al. ACE of Spades in the IoT Security Game: A Flexible IPsec Security Profile for Access Control. **Conference on Communications and Network Security (CNS)**, [s. l.], n. IEEE, 2018.

BARBETTINI, N. **OAuth and OpenID Connect(in plain english)**, 2017.

BELTRAN; SKARMETA. **Overview of Device Access Control in the IoT and its Challenges**, 2018. Disponível em: <https://ieeexplore.ieee.org/document/8470752>

BELTRAN, Victoria; SKARMETA, Antonio F. An overview on delegated authorization for CoAP: Authentication and authorization for Constrained Environments (ACE). **2016 IEEE 3rd World Forum on Internet of Things, WF-IoT 2016**, [s. l.], p. 706–710, 2017.

BERTIN, Emanuel et al. **Access control in the Internet of Things: a survey of existing approaches and open research questions**, 2019. Disponível em: <https://www.researchgate.net/publication/331475112_Access_control_in_the_Internet_of_Things_a_survey_of_existing_approaches_and_open_research_questions>

BORGIA, Eleonora. The internet of things vision: Key features, applications and open issues. **Computer Communications**, [s. l.], v. 54, p. 1–31, 2014. Disponível em: <http://dx.doi.org/10.1016/j.comcom.2014.09.008>

BORMANN, Carsten; HOFFMAN, Paul. **Concise Binary Object Representation**

(CBOR), 2013. Disponível em: <https://tools.ietf.org/html/rfc7049>

BOTYRIUTE. **Access to Online Resources**, 2018.

CAMPBELL, B. et al. **Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants**, 2015. Disponível em: <https://tools.ietf.org/html/rfc7521>

CARPINETTI. **Planejamento e Análise de Experimentos**. [s.l: s.n.].

CLAEYS, Timothy; ROUSSEAU, Franck; TOURANCHEAU, Bernard. Securing Complex IoT Platforms with Token Based Access Control and Authenticated Key Establishment. **2017 International Workshop on Secure Internet of Things (SIoT)**, Oslo, Norway, p. 1–9, 2017. Disponível em: <https://ieeexplore.ieee.org/document/8394976>

COELHO, Flavia; ARAÚJO, Luiz; BEZERRA, Edson. **Segurança da Informação NBR 27001 e NBR 27002**Escola Superior de Redes, , 2014. Disponível em: <https://pt.scribd.com/doc/136758554/Gestao-da-Seguranca-da-Informacao-NBR-27001-e-NBR-27002>

DEEP, Samundra; ZHENG, Xi; HAMEY, Len. **A survey of security and privacy issues in the Internet of Things from the layered context**, 2019. Disponível em: <https://arxiv.org/pdf/1903.00846.pdf>

ECHEVERRÍA, Sebastian; KLINEDINST, Dan; SEITZ, Ludwig. **Authentication and Authorization for IoT Devices in Disadvantaged Environments**, 2019.

EL-HAJJ, Mohammed et al. A survey of internet of things (IoT) authentication schemes. **Sensors (Switzerland)**, [s. l.], v. 19, n. 5, p. 1–43, 2019.

EUROPOL. **First report of the observatory function on encryption**EUROPOÇ, , 2019. Disponível em: <https://www.europol.europa.eu/sites/default/files/documents/final_report_of_the_observatory_function.pdf>

FREMANTLE, Paul; AZIZ, Benjamin. Cloud-based federated identity for the Internet of Things. **Annals of Telecommunications**, [s. l.], v. 73, p. 415–427, 2018.

GERBER, Urs. **Authentication and Authorization for Constrained Environments (ACE)**. 2018. [s. l.], 2018. Disponível em: <https://www3.unifr.ch/inf/softeng/en/assets/public/files/research/students_projects/master/Master_Gerber_Urs.pdf>

GIL. **Como elaborar projetos depesquisa**. 4. ed. São Paulo: Atlas, 2002.

GONG, L. **Variations on the themes of message freshness and replay-or the difficulty in devising formal methods to analyze cryptographic protocols**, 1993. Disponível em: <https://ieeexplore.ieee.org/document/246633>

HARDT, Dick. **The OAuth 2.0 Authorization Framework**, 2012. Disponível em: <https://tools.ietf.org/html/rfc6749>

HILT, Stephen et al. **Cybersecurity Risks in Complex IoT Environments : Other Structures**TrendMicro, , 2019.

HOVSMITH, Skip. **Adapting OAuth2 for Internet of Things (IoT) API Security**, 2017. Disponível em: <https://hackernoon.com/adapting-oauth2-for-internet-of-things-iot-api-security-59044d2472d>

INGOLDSBY. **Attack Tree Threat Risk Analysis**, 2013. Disponível em: <https://www.amenaza.com/downloads/docs/AttackTreeThreatRiskAnalysis.pdf>

JONES, M. et al. **Proof-of-Possession Key Semantics for CBOR Web Tokens (CWTs)**, 2019. Disponível em: <https://tools.ietf.org/html/draft-ietf-ace-cwt-proof-of-possession-08>

JUNG, Seung Wook; JUNG, Souhwan. Personal OAuth authorization server and push OAuth for Internet of Things. **International Journal of Distributed Sensor Networks**, [s. l.], v. 13, n. 6, p. 1–11, 2017.

KAWASAKI. **Diagrams of All The OpenID Connect Flows**, 2017.

KRAWCZYK, Hugo; BELLARE, Mihir; CANETTI, Ran. **HMAC: Keyed-Hashing for Message Authentication**, 1997. Disponível em: <https://tools.ietf.org/html/rfc2104>

KUMAR, Suraj. **OAuth 2.0 Beginner's Guide**, 2019. Disponível em: <https://dzone.com/articles/oauth-20-beginners-guide>

LAGUTIN, Dmitrij et al. Enabling Decentralised Identifiers and Verifiable Credentials for Constrained IoT Devices using OAuth-based Delegation. [s. l.], n. February, 2019.

LAKATOS. **Fundamentos de metodologia científica**. 5. ed. São Paulo: Atlas, 2003.

LEVIN, Guy. Four Most Used REST API Authentication Methods. [s. l.], p. 1–6, 2019.

LI, Shuai. **A survey on security analysis of OAuth 2.0 framework**, 2017. Disponível em: <https://pages.cpsc.ucalgary.ca/~shuai.li1/blog/survey-security-analysis-Shuai>

LIN, Jie et al. A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications. **IEEE Internet of Things Journal**, [s. l.], v. 4662, n. c, p. 1–1, 2017. Disponível em: <http://ieeexplore.ieee.org/document/7879243/>

LODDERSTEDT, Torsten; MCGLOIN, Mark; HUNT, Phil. **OAuth 2.0 Threat Model and Security Considerations**, 2013. Disponível em: <OAuth 2.0 Threat Model and Security Considerations>

MADUREIRA. **Modelo de Gestão de Identidade e Acessos**, 2010. Disponível em: <https://repositorio-aberto.up.pt/bitstream/10216/67888/1/000150424.pdf>

MAHALLE, P.; BABAR, S. **Identity management framework towards internet of things (iot): Roadmap and key challenges**, 2010.

MIDRACK. **What Is a Third-Party App ?**, 2019. Disponível em: <https://www.lifewire.com/what-is-a-third-party-app-4154068>

ORACLE. **An Introduction To OpenID Connect**, 2017. Disponível em: <https://www.youtube.com/watch?v=6DxRTJN1Ffo>

OWASP. **Authorization**. 2019. Disponível em: <https://cheatsheetseries.owasp.org/cheatsheets/Access_Control_Cheat_Sheet.html>.

PIRETTI. **CoAP over DTLS TinyOS Implementation and Performance Analysis**, 2013.

PREVITALI. **The Business Viewpoint of Securing the Industrial Internet Executive Overview**, 2016. Disponível em: <https://www.iiconsortium.org/pdf/IIC-Security-WP.pdf>

RAZA, Shahid et al. Lithe: Lightweight secure CoAP for the internet of things. **IEEE Sensors Journal**, [s. l.], v. 13, n. 10, p. 3711–3720, 2013.

RESCORLA, E. **Guidelines for Writing RFC Text on Security Considerations**IETF, , 2005. Disponível em: <https://tools.ietf.org/html/rfc3552>

RESCORLA, Eric; MODADUGU, Nagendra. **Datagram Transport Layer Security Version 1.2**, 2012. Disponível em: <https://tools.ietf.org/html/rfc6347>

RICHER, Justin et al. **OAuth 2.0 Dynamic Client Registration Protocol**, 2015. Disponível em: <https://tools.ietf.org/html/rfc7591>

RICHER, Justin; SANSO, Antonio. **OAuth 2 in Action**. [s.l.] : Manning Publications Co, 2017.

SAKIMURA, Nat et al. **OpenID Connect Core**, 2014. Disponível em: <https://openid.net/specs/openid-connect-core-1_0.html>

SAKIMURA, Nat. **The Basic Concepts of OAuth**, 2017. Disponível em: <https://www.youtube.com/watch?v=5zfIFpp3AFY>

SAKIMURA, Nat; BRADLEY, John; AGARWAL, Naveen. **Proof Key for Code Exchange by OAuth Public Clients**, 2015. Disponível em: <https://www.rfc-editor.org/rfc/rfc7636.html>

SAYANA, Laxman Sayana; JOSHI, Binnet Kumar. Security Issues and Challenges in Internet of Things. [s. l.], 2016.

SCHAAD, J; CELLARS, A. **CBOR Object Signing and Encryption (COSE)**, 2017. Disponível em: <https://tools.ietf.org/html/rfc8152>

SCHIEFER, Michael. Smart Home Definition and Security Threats. **Proceedings - 9th International Conference on IT Security Incident Management and IT Forensics, IMF 2015**, [s. l.], p. 114–118, 2015.

SCHNEIER. **Attack Trees**, 1999. Disponível em: <https://www.schneier.com/academic/archives/1999/12/attack_trees.html>

SCIANCALEPORE et al. OAuth-IoT: An access control framework for the Internet of Things based on open standards. **Symposium on Computers and Communications (ISCC),** [s. l.], n. IEEE, 2017.

SCULLY, Neil; MCDONALD, John; MCEVOY, Kathleen. Library Analytics: Shaping the Future — Data, Privacy and the User Experience. [s. l.], 2018.

SEITZ, L. et al. **Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth)**IETF, , 2020. Disponível em: <https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-33>

SEITZ, Ludwig; GERDES, Stefanie. **Use Cases for Authentication and Authorization in Constrained Environments**, 2016. Disponível em: <https://tools.ietf.org/html/rfc7744>

SELANDER, Goeran; MATTSSON, John; PALOMBINI, Francesca. **Ephemeral Diffie-Hellman Over COSE (EDHOC)**, 2019. Disponível em: <https://tools.ietf.org/html/draft-selander-ace-cose-ecdhe-14>

SHARMA. **Deep Dive Into OAuth2.0 and JWT (Part 2 OAuth2.0)**, 2019. Disponível em: <https://dzone.com/articles/deep-dive-to-oauth20-amp-jwt-part-2-oauth20>

SHELBY, Z.; HARTKE, K.; BORMANN, C. **The Constrained Application Protocol (CoAP)**, 2014. Disponível em: <https://tools.ietf.org/html/rfc7252>

SHIREY, Robert W. **Internet Security Glossary, Version 2**, 2007.

STALLINGS, Williiam. **Cryptographt and network security**. [s.l: s.n.].

STOJKOSKA, Biljan; TRIVODALIE, Kire V. A review of Internet of Things for smart home: Challenges and solutions. **Journal of Cleaner Production**, [s. l.], v. 140, p. 1454–1464, 2017. Disponível em: <http://dx.doi.org/10.1016/j.jclepro.2016.10.006>

TSCHOFENIG, Hannes. Fixing User Authentication for the Internet of Things (IoT). **Datenschutz und Datensicherheit - DuD**, [s. l.], v. 40, n. 4, p. 222–224, 2016.

TSCHOFENIG, Hannes; BACCELLI, Emmanuel. **Cyberphysical Security for the Masses A Survey of the Internet Protocol Suite for Internet of Things Security**, 2019. Disponível em: <https://ieeexplore.ieee.org/document/8764344>

VERMESAN, O.; FRIESS, P. **Internet of Things: Converging Technologies for Smart Environments and Integrated Ecosystems**, 2013.

WESTPHALL, Johann. Desenvolvimento de uma Aplicação com dispositivo IoT usando Protocolos DTLS e CoAP. [s. l.], 2018.

ZIBETTI, Andre. **Distribuição Exponencial**, 2016. Disponível em: <https://www.inf.ufsc.br/~andre.zibetti/probabilidade/exponencial.html>

APÊNDICE A — FLUXOGRAMAS DA IMPLEMENTAÇÃO

ENDPOINT DE AUTORIZAÇÃO

O fluxo do *endpoint* de autorização no servidor de autorização é apresentado na Figura 1.

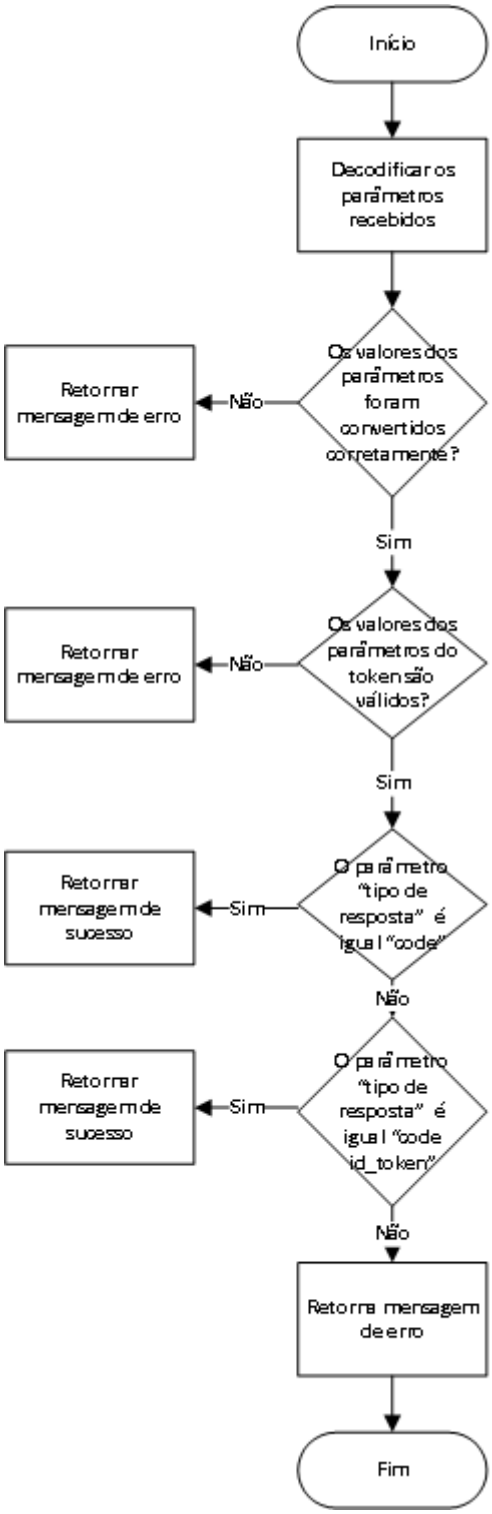

Figura 1 – Fluxograma de execução do endpoint de autorização.

ENDPOINT DO TOKEN

O fluxo do *endpoint* de token no servidor de autorização é apresentado na Figura 2.

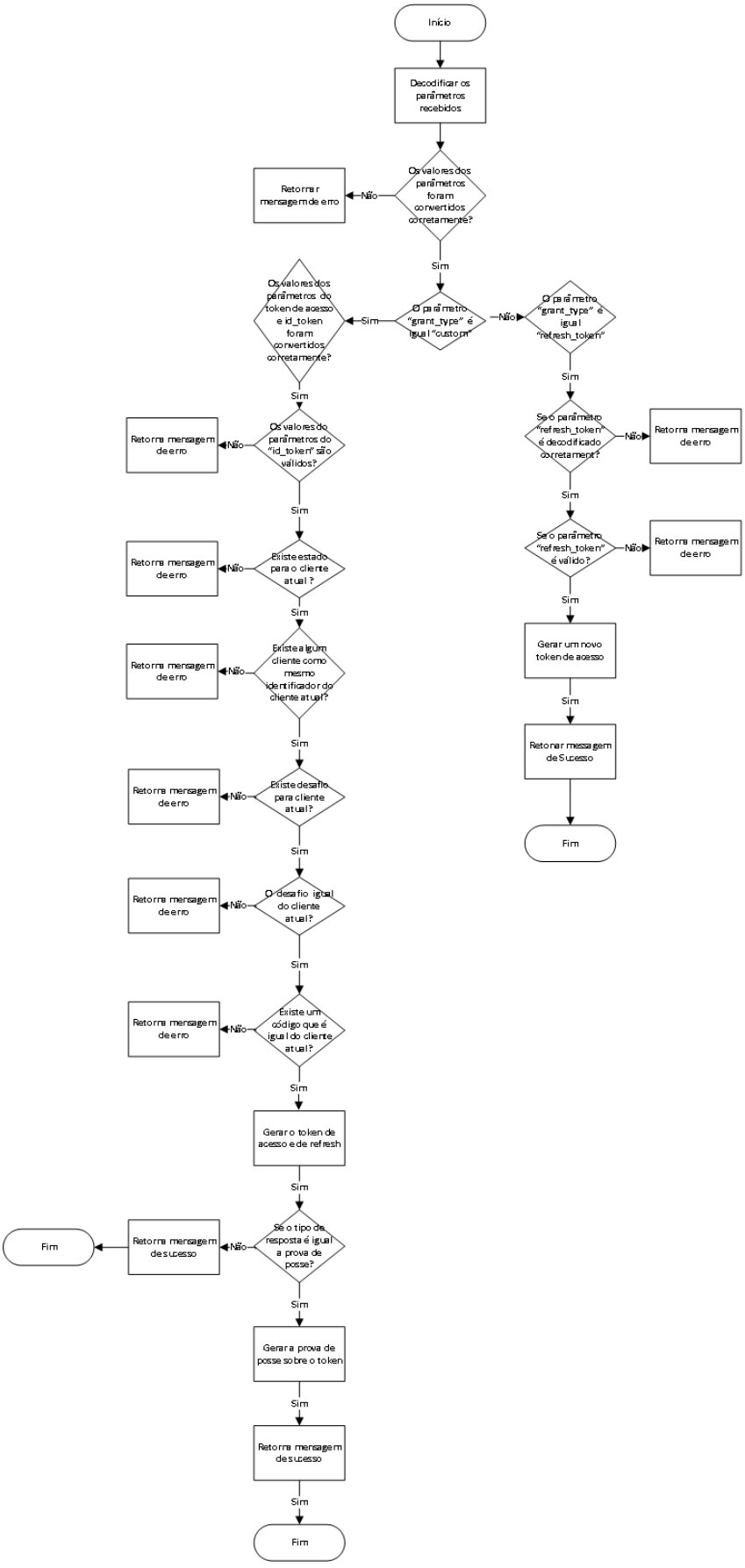

Figura 2 – Fluxograma da execução do endpoint de token.

ENDPOINT DE RECURSO

O fluxo do *endpoint* de recurso no servidor de recurso é apresentado na Figura 3.

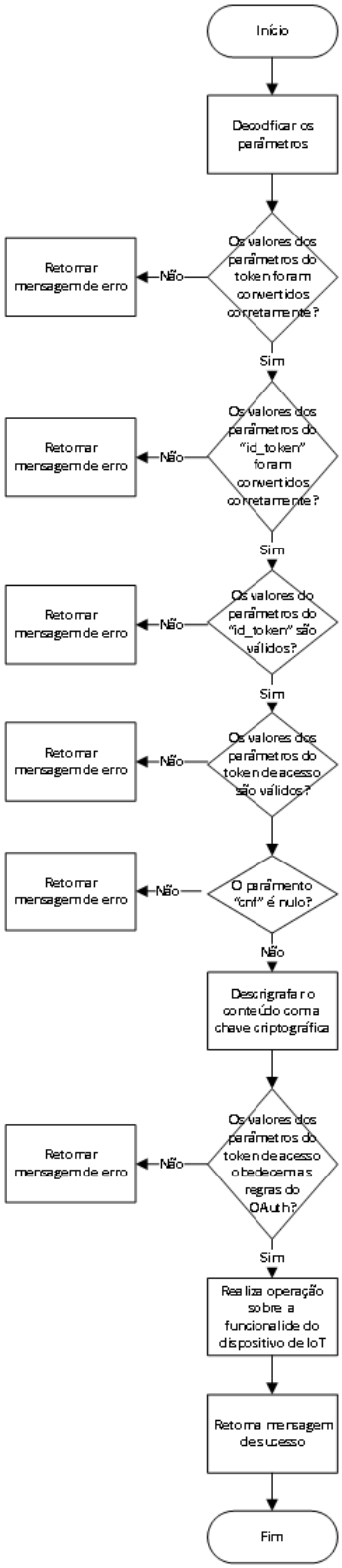

Figura 3 – Fluxograma da execução do endpoint de recurso.

APÊNDICE B — GERENCIAMENTO DINÂMICO DA CREDENCIAL

REGISTRO DE DINÂMICO DA CREDENCIAL

Na Figura 4 utiliza uma das propostas projetadas, mas sem concessão de autenticação sobre registro das credenciais e para uso da credencial, em seguida, para solicitação da concessão de autorização.

Figura 4 – Fluxo sem concessão de autenticação para solução do registro da credencial.

REGISTRO DINÂMICO DA CREDENCIAL COM SOLICITAÇÃO DE CONCESSÃO DE ACESSO

Na Figura 5, uma segunda proposta para o problema de credencial, mas com concessão de autenticação sobre registro das credenciais e para uso da credencial, em seguida, para solicitação da concessão de autorização.

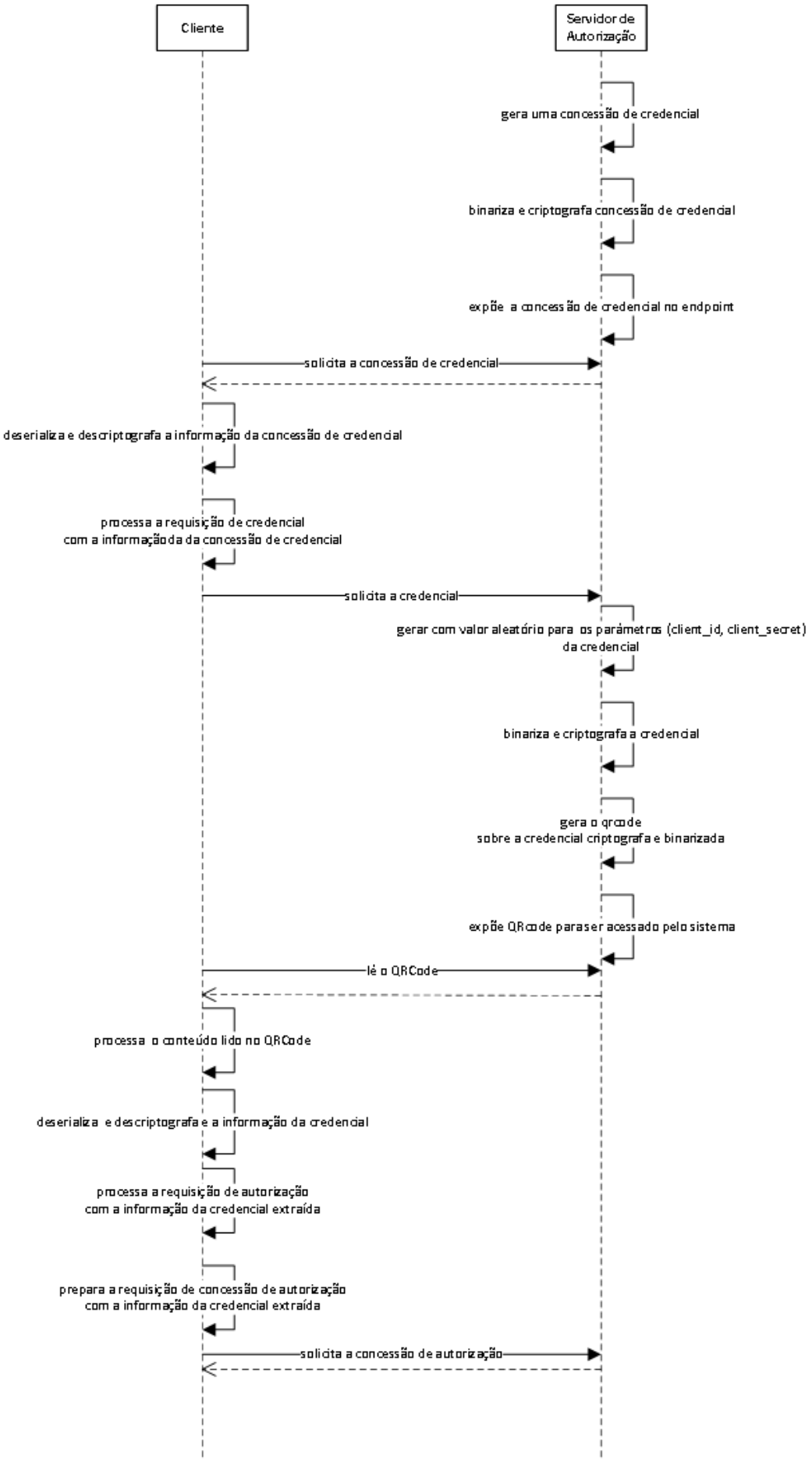
Cliente
Servidor de Autorização
gera uma concessão de credencial
binariza e criptografa concessão de credencial
expõe a concessão de credencial no endpoint
solicita a concessão de credencial
deserializa e descriptografa a informação da concessão de credencial
processa a requisição de credencial
com a informação da da concessão de credencial
solicita a credencial
gerar com valor aleatório para os parâmetros (client_id, client_secret)
da credencial
binariza e criptografa a credencial
gera o qrcode
sobre a credencial criptografa e binarizada
expõe QRcode para ser acessado pelo sistema
lê o QRCode
processa o conteúdo lido no QRCode
deserializa e descriptografa e a informação da credencial
processa a requisição de autorização
com a informação da credencial extraída
prepara a requisição de concessão de autorização
com a informação da credencial extraída
solicita a concessão de autorização

Figura 5 – Fluxo com concessão de autenticação para solução do registro da credencial.

APÊNDICE C — MENSAGENS

PARÂMETROS NA REQUISIÇÃO DO CLIENTE PARA ENDPOINT DE AUTORIZAÇÃO

Na Tabela 1 Utilizou os seguintes parâmetros para usados na requisição realizada pelo cliente ao *endpoint* (/authorization) do servidor de autorização, o símbolo (*) significa parâmetro é obrigatório:

Tabela 1 – Parâmetros para chamada ao endpoint (/authorization)

Chave CBOR	Parâmetro	Valor	Tipo do Valor CBOR
24	client_id*	ace-oidc-oauth-client-xxx	Caracteres de texto
27	redirect_uri*	coaps://localhost:9000/calback	Caracteres de texto
9	scope*	[“foo”, “bulb”, “openid”]-	Vetor de caracteres de texto em bytes
-1	code_challenge*	Xxxxxx	Caracteres de texto
-2	code_challenge_method*	Xxxxxxxxx-	Caracteres de texto
28	state*	xxxxxxx-	Caracteres de texto
26	response_type*	“code id_token”	Caracteres de texto

PARÂMETROS NA REQUISIÇÃO DO CLIENTE PARA ENDPOINT DE TOKEN

Na Tabela 2, utilizou os seguintes parâmetros para usados na requisição realizada pelo cliente ao *endpoint* (/token) do servidor de autorização, o símbolo (*) significa parâmetro é obrigatório:

Tabela 2 – Parâmetros para chamadas ao endpoint (/token)

Chave CBOR	Parâmetro	Valor	Tipo do Valor CBOR
24	client_id*	ace-oidc-oauth-client-xxx	Caracteres de texto
27	redirect_uri*	coaps://localhost:9000/calback	Caracteres de Texto
28	scope*	[“foo”, “bulb”, “openid”]	Caracteres de Texto
-3	code_verifier*	Xxxxxxxxx -	Caracteres de Texto
33	grant_type*	-1	Inteiro negativo
29	code*	Xxxxxxxxx	Caracteres de Texto
28	state*	Xxxxxxxxx	Caracteres de Texto
-5	id_token*	{ iss:’’,sub:’’ ,aud:’’,exp:’’, nbf: ’’,iat:’’,cti:’ ’}	Mapa com chave e valor
26	response_type*	“pop”	Caracteres de texto

PARÂMETROS NA REQUISIÇÃO DO CLIENTE PARA ENDPOINT DE RECURSO

Na Tabela 3, utilizou os seguintes parâmetros para usados na requisição realizada pelo cliente ao *endpoint* (/.well-known) do servidor de autorização, o símbolo (*) significa parâmetro é obrigatório:

Tabela 3 – Parâmetros para chamadas ao endpoint (/.well-known)

Chave CBOR	Parâmetro	Valor	Tipo do Valor CBOR
24	client_id*	ace-oidc-oauth-client-xxx	Caracteres de texto

9	scope*	[“foo”, “bulb”, “openid”]	Array de bytes
2	expires_in*	15000	Inteiro positivo
5	cnf*	cose_encrypt0{ header_unprotected{} header_protected{} chipertext{}, }	Mapa com chave e valor
-5	id_token*	{ iss:’’,sub:’’ ,aud:’’,exp:’’, nbf: ’’,iat:’’,cti:’ ’}	Mapa com chave e valor

PARÂMETROS DAS MENSAGENS

Algumas das chaves do CBOR utilizadas precisaram ser criadas pela implementação e foram definidas como números inteiros negativos, com o objetivo de teste, para não haver colisão como outros parâmetros já padronizados. Já que, para usar novos números para representar o atributo é necessário fazer pedido de registro no IANA[71]. Além disso, um novo valor do tipo de concessão ("`grant_type`") foi definido como inteiro negativo, pelo mesmo motivo, para evitar colisão com as chaves do mapa CBOR que já tenham sido registradas anteriormente. Outro ponto a ser destacado é que tanto a chave quanto os valores do mapa em CBOR são codificados na representação hexadecimal. Nesse sentido, o formato de dados ou do conteúdo transferidos na comunicação entre as partes são abordados na implementação, Tabela 4.

Tabela 4 – Formato de dados para transferir informação do recurso

Formato de Dados	Tipo de Mídia IANA	Formato do Conteúdo Numérico
cbor	application/cbor	60

[71] https://www.iana.org/about

APÊNDICE D — FLUXOS DO OAUTH

FLUXO IMPLÍCITO

O fluxo implícito[72][73] é otimizado para clientes conhecidos para operar sobre um a redireção particular através de URI. Este tipo de autenticação não envolve autenticação do cliente. Os clientes são implementados em aplicações móveis ou Web usando uma linguagem *script* como Javascript. Nesse tipo de concessão o cliente recebe o *token* de acesso como resultado da requisição de autorização Figura 6.

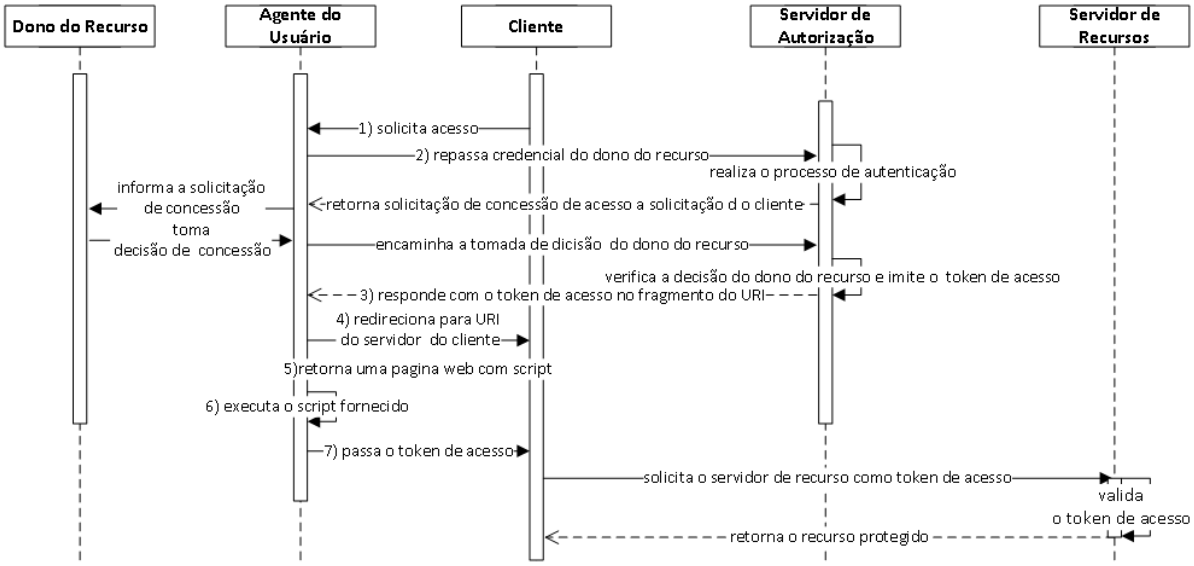

Figura 6 – Fluxo implícito . Fonte: adaptado de (HARDT, 2012).

De acordo com as etapas realizadas para o fluxo do tipo de concessão implícito são descritas a seguir:

1) O cliente inicia o fluxo pelo direcionamento do agente do usuário do dono do recurso para autorização. O cliente inclui seus próprios parâmetros para o servidor de autorização e que serão enviados de volta para o agente do usuário uma vez que o acesso seja concedido ou proibido.

2) O servidor de autorização autentica o dono do recurso (via agente do usuário) e estabelece caso o dono do recurso conceda ou proíba a solicitação de acesso do cliente.

3) Assumindo o que o dono do recurso conceda o acesso, o servidor de autorização redireciona de volta do agente do usuário para o cliente usando a URI

[72] *Implicit Grant Type*: https://tools.ietf.org/html/rfc6749#section-4.2
[73] *Implicit Grant Type Flow*: https://tools.ietf.org/html/rfc6749#page-32

de redirecionamento oferecida mais cedo. O redirecionamento da URI inclui o *token* de acesso no fragmento URI.

4) O agente do usuário permite que as instruções de redirecionamento possa fazer uma solicitação para recurso do cliente armazenado web. O agente do usuário retém localmente a informação fragmentada.

5) O recurso do cliente hospedado na web retorna uma página web capaz de acessar o redirecionamento URI total incluindo o fragmento retido pelo agente do usuário e extrai o *token* de acesso

6) O agente do usuário executa o script fornecido pelo recurso do cliente hospedado na web localmente, qual extrai o *token* de acesso.

7) O agente do usuário passa o *token* de acesso para o cliente

FLUXO DE CREDENCIAIS COM SENHA DO DONO DO RECURSO

O fluxo de concessão com senha do dono do recurso74 é otimizado para clientes confidenciais. Nesse tipo a concessão é adequada nos casos o dono de recurso tem um relacionamento de confiança com o cliente. Credenciais do usuário (aplicações confiáveis).

Essa concessão é usada tipicamente em formulários interativos e para migração de esquemas de autenticação baseado em HTTP básico ou sumarização (*digest)*. A Figura 7 que representa o fluxo da concessão de código de autorização.

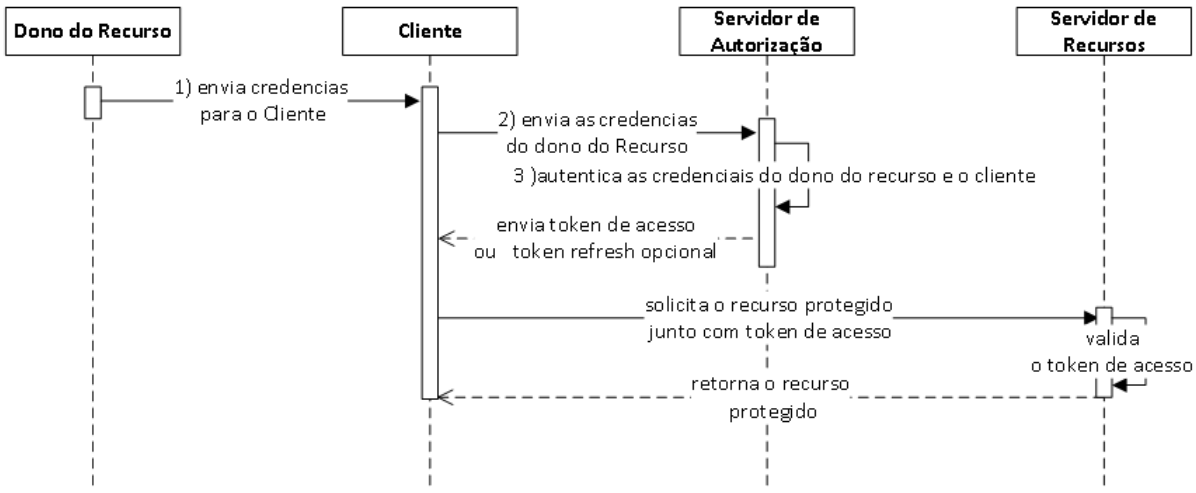

Figura 7 – Fluxo de credenciais com credenciais do dono do recurso. Fonte: adaptado de (HARDT, 2012).

As etapas realizadas para o tipo de concessão de credencial dono do recurso

[74] *Resource Owner Password Credentials Flow* : https://tools.ietf.org/html/rfc6749#section-4.3

são descritas a seguir:

1) O dono do recurso oferece ao cliente as suas credenciais
2) O cliente solicita um *token* de acesso do servidor de autorização pela inclusão das credenciais recebidas pelo dono do recurso. Quando é feito a solicitação, o cliente autentica com o servidor de autorização.
3) O servidor de autorização autentica o cliente e valida as credenciais do dono do recurso, e se é válido, emite um *token* de acesso.

FLUXO DE CREDENCIAIS DO CLIENTE

No fluxo de concessão credencial do cliente a própria aplicação acessa informações sobre sua conta via API. A concessão das credenciais do cliente[75] é dada sobre o cliente que seu próprio controle e apenas usa suas próprias credenciais para fazer autenticação. A Figura 8 adaptada que representa o fluxo[76] da concessão credenciais do cliente.

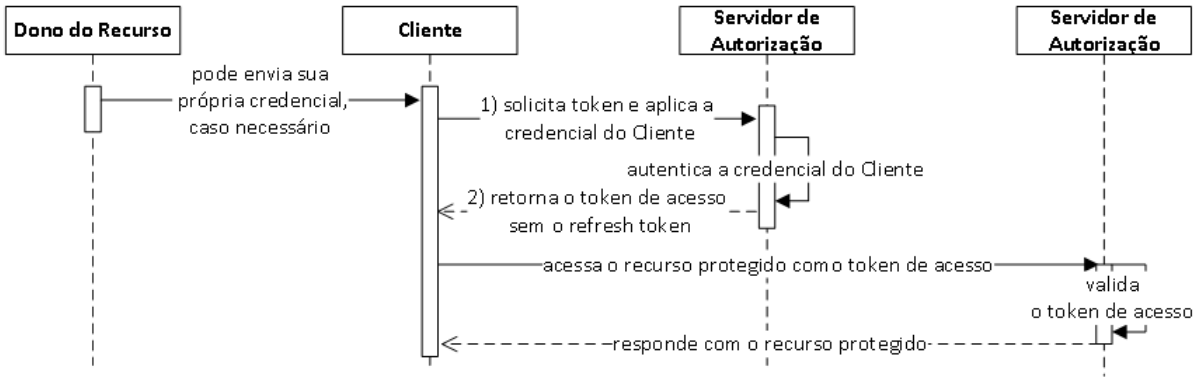

Figura 8 – Fluxo de credenciais do cliente. Fonte: adaptado de (HARDT, 2012).

Na Figura, segue as etapas realizadas para o tipo de concessão de credenciais do cliente são descritas a seguir:

1) O cliente solicita um *token* de acesso, utilizando da própria credencial do cliente para se autenticar.
2) O servidor de autorização autentica o cliente e valida a credencial do cliente. Se for válido, emite um *token* de acesso.

[75] *Client Credentials Grant*:: https://tools.ietf.org/html/rfc6749#section-4.4
[76] Client Credentials *Flow* : https://tools.ietf.org/html/rfc6749#page-41

FLUXO DE DISPOSITIVO

O fluxo do dispositivo[77] é uma variação ou extensão dos fluxos de autorização por meio do OAuth 2.0. O fluxo de dispositivo[78] atende os dispositivos sem navegador e com acesso restrito (ex. smart TV, impressoras, consoles de mídia etc.). O fluxo de dispositivos adaptado é mostrado na Figura 9.

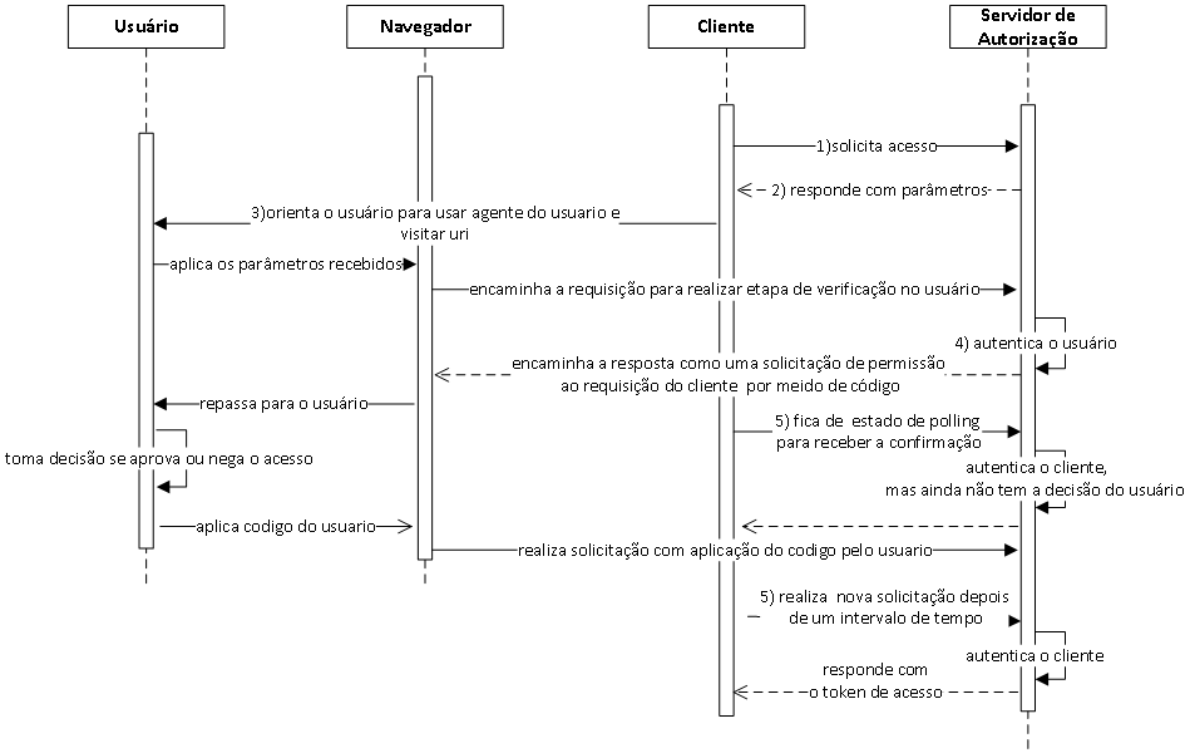

Figura 9 – Fluxo do dispositivo. Fonte: Adaptado de (TSCHOFENIG et al., 2019).

Na Figura 9, as etapas realizadas para o tipo de concessão de dispositivo são descritas a seguir:

1) o cliente solicita acesso passando como parâmetro o (`device_id`)
2) o servidor de autorização responde com os parâmetros (`device_code`, `user_code`, `verification_uri`). O usuário aplica os parâmetros (`user_code`) e (`verification_uri`) na solicitação para o agente do usuário. O agente do usuário encaminha a requisição para realizar etapa de verificação no usuário no servidor de autorização.
3) O servidor de autorização recebe a solicitação de verificação do usuário, autentica o usuário e encaminha a solicitação de permissão enviada pelo o

[77] Draft v15 do “OAuth 2.0 Device Authorization Grant”: https://tools.ietf.org/html/draft-ietf-oauth-device-flow-15
[78] Fluxo de dispositivo: https://tools.ietf.org/html/draft-ietf-oauth-device-flow-15#page-4

cliente. O agente do usuário repassa a solicitação do servidor de autorização para o usuário.

4) Depois disso, o cliente fica de estado de polling para receber a confirmação passando os parâmetros de (`verification_code`, `client_id`). O servidor de autorização autentica o cliente, mas espera receber decisão de confirmação realizada pelo o usuário. Em dado momento, o usuário toma decisão se aprova ou nega o acesso. No caso onde o usuário aprova o acesso, o usuário aplica código do usuário e repassa para o agente do usuário que trata de realizar a solicitação com aplicação do código pelo usuário para o servidor de autorização.
5) Depois de passar um determinado tempo desde última tentativa, o cliente realiza uma nova solicitação com os parâmetros (“`verification_code`, `client_id`). O servidor de autorização autentica o cliente e como ele recebeu a decisão de confirmação do usuário dessa vez, ele, o servidor de autorização responde com o *token* de acesso para o cliente.

APÊNDICE E — CONTROLES DE ACESSO NAS PLATAFORMAS DE IOT

No trabalho de EL-HAJJ et al. (2019) é realizado um levantamento sobre os controles de acesso utilizados nas plataformas de soluções de IoT, os quais podem ser aplicados sobre as aplicações (*third-party*), na Tabela 5.

Tabela 5 - Controles de acesso utilizados nas plataformas de soluções de IoT. Fonte: (EL-HAJJ et al., 2019).

Plataforma	Estratégia	Método ou Técnica
AWS	Autenticação do dispositivo de IoT Autenticação mútua de todos os pontos	Certificados X.509: Chave Pública e certificados digitais ;Gerenciamento de Acesso de Identidade (IAM): Identidade Usuário, Grupo e papéis; HTTP e WebSocket
ARM mbed IoT	Não uma maneira específica de se aplicar de IoT, mas várias opções que fica a critério do desenvolvedor	Padrões criptográficos; Mecanismos de troca de chave assinaturas base certificados digitais. Ex.: certificados X.509.
Google Wave	Descoberta, provisionamento e autenticação de dispositivos e usuários	Protocolo de OAuth 2.0 junto com certificados digitais são usados para autenticação Embora o servidor cloud habilitado pela Wave seja escolhido pelo o usuário, Google oferece servidor de autenticação a parte
Calvin	Autenticação	Autenticação local a qual tem um valor hash da credencial que são armazenadas em arquivo

		JSON Máquina externa que age como um servidor de autenticação e realiza Usando um servidor RADIUS
Apple Home kit	Assinatura Digital	Baseados sobre assinatura de chave pública ED15519 As chaves são armazenadas em cadeias protegidas e sincronizadas entre os dispositivos As chaves são trocadas usando protocolo de senha remota, o qual tem 8 dígitos é oferecido pelo fabricante e deve ser inserido pelo o usuário via Interface Gráfica do dispositivo IoS
KURA	Canal de Segurança e criptografia assimétrica	Sockets seguros oferecidos pelo ambiente Java Runtime Inclui criptografia de chave pública para autenticar a comunicação como dispositivos remotos e gateways.
Samsung Smart Things	Usa protocolo OAuth 2.0 para autenticação do dispositivo inteligente (SmartDevice)	Dispositivos conectados na nuvem ou local seguem um procedimento diferente para autenticação devido o uso de outros protocolos de comunicação para passar pelo o gateway e conectar diretamente na nuvem

		Principal função é o gerenciador de serviço que gerencia a autenticação dos serviços de cloud de terceiros A identificação de SmartDevice através de vasta gama de dispositivos suportados de vários vendedores Fatores que incluem identificadores únicos, número serial, endereço IMAC e IP

APÊNDICE F — AMEAÇAS E ATAQUES DE PERSONIFICAÇÃO E REPLICAÇÃO

Grande parte das ameaças enfrentados pelo ACE-OAuth são herdadas do OAuth 2.0. Desse modo, as ameaças que podem ocasionar os ataques de replicação e personificação. A Figura 6 mostra as ameaças baseadas no modelo de ameaça do OAuth 2.0.

Tabela 6 – Modelos de ameaças sobre credencial. Fonte: (LODDERSTEDT; MCGLOIN; HUNT, 2013)

Tipo de Ameaça	Ameaça	Impacto
Obtenção dos segredos do cliente	a) O atacante poderia tentar obter acesso ao segredo de um cliente particular a fim de replicar seus *token*s de refresh e códigos de autorização; b) Obter *token*s em nome do cliente atacado com os privilégios desse “cliente_id” agindo como uma instância do cliente.	a) autenticação do cliente de acesso ao servidor de autorização pode passar despercebido b) *token*s de refresh roubados ou códigos de autorização podem ser replicados.
Divulgação das credenciais do cliente durante a transmissão	Um invasor pode tentar interceptar a transmissão do cliente credenciais entre o cliente e o servidor durante o processo de autenticação do cliente ou durante solicitações de *token*.	Revelação de uma credencial do cliente que permite realizar a pesca (phishing) ou personificação de um serviço ao cliente
Pescaria do código de autorização	Um atacante pode se passar por um site cliente e obter acesso ao "código" de autorização. Isso pode ser alcançado usando falsificação DNS ou ARP.	Isso afeta aplicativos que se utilizam do parâmetro de redirecionamento URI pode levar à divulgação de "códigos" de autorização e, potencialmente, o

	Isso se aplica aos clientes, que usam redirecionamento de URI de redirecionamento não é local para o host de origem que efetua a requisição e no navegador do usuário está executando.	correspondente acesso e atualização de *token*s
Personificando a sessão do usuário	Uma atacante pode se passar por um site cliente e obter acesso na sessão do usuário. Isso pode ser alcançado usando falsificação DNS ou ARP. Isso se aplica aos clientes, que usam redirecionamento de URI que não é local para o host de origem que efetua a requisição e o navegador do usuário está executando	Um invasor que intercepta o "código" de autorização assim que é enviado para o *endpoint* de retorno de chamada (*callback*) pode obter acesso a recursos protegidos enviando o "código" de autorização ao cliente. O cliente trocará o "código" de autorização por um *token* de acesso e usará o *token* de acesso para acessar recursos protegidos em benefício do invasor, fornecendo recursos protegidos ao invasor ou modificando os recursos protegidos conforme as instruções do invasor

APÊNDICE G — GERENCIAMENTO DE IDENTIDADE

O gerenciamento de identidade contribui para o gerenciamento e o controle de acesso seguro aos recursos e à informação (ABOMHARA; KØIEN, 2014) e, em conjunto com serviço de autenticação, identifica objetos a fim de estabelecer a comunicação confiável entre as partes (MAHALLE; BABAR, 2010). E, em um cenário hostil como é IoT, o gerenciamento de identidade é essencial realizar para o controle de aplicações, dispositivos e usuários através de serviços confiáveis.

De qualquer modo, segundo (MADUREIRA, 2010) a gestão de identidades pode ser definida como a informação ou o conjunto de fluxos que são suficientes para identificar quem é a entidade que tem acesso ao sistema de informação. Trata-se de um processo que aplica identidades aos sujeitos e que deve ser definida e gerenciada por meio do ciclo de vida (criação, suspensão, alteração e remoção) do uso da identidade.

OPENID CONNECT

OpenID Connect (OIDC)[79] é um camada de autenticação e/ou identidade que atua sobre o OAuth 2.0 (SAKIMURA et al., 2014). O (OIDC) é uma solução aberta, gratuita e descentralizada e que tem como objetivo atacar os seguintes problemas: modelo de autenticação tradicional baseado em senha, falta do suporte à aplicações nativas e incompatibilidade no formato na troca de dados (XML).

De qualquer forma, para entender as características e o funcionamento do OIDC é necessário tomar ciência dos conceitos básicos tais como, os participantes, a estrutura da credencial de concessão de acesso, afirmações (*claims*), escopos, ID Token e a diferença entre OAuth 2.0 e OIDC.

COMPARAÇÃO ENTRE OAUTH 2.0 E OPENID CONNECT

O OIDC permite que aplicações clientes verifiquem a identidade do usuário final baseado na autenticação realizada por um servidor de autorização, mas diferentemente do OAuth 2.0 que fornece autorização com pseudo-autenticação. Com OAuth 2.0, a autenticação e confirmação do usuário são apresentados para servidor de autorização, mas o único propósito disso é de criar e conceder ou permitir a autorização para aplicação cliente. Na realidade, OAuth 2.0 oferece uma

[79] https://openid.net/connect/

concessão ou licença para acessar os recursos, muito mais do que oferecer informação sobre a própria autenticação.

A pseudo-autenticação do OAuth 2.0 é como síndico de um edifício que dá a chave temporária para alguém que seja dono do apartamento, mas o síndico não vai querer saber a identidade de quem aluga o espaço ou de quem seja de confiança do proprietário. A chave implica apenas no direito de entrar no apartamento para uma fração específica do tempo. Isto não implica que o indivíduo é o dono do apartamento. Portanto, não há implicação de identidade nesta ocasião.

A pseudo-autenticação do OAuth 2.0 não fornece também outras informações como, por exemplo, oferecer quando, onde, e como a autenticação atualmente ocorreu. Além disso, dentro das demais limitações, OAuth 2.0 não permite *Single Sign On* (SSO) federado. Entretanto, OIDC, em aplicações baseada em nuvem pode obter informação da identidade para recuperar detalhes sobre evento da autenticação. O OIDC realiza a troca de informação usando a API REST.

Dessa maneira OIDC pode adicionar as seguintes características em comparação ao OAuth 2.0: identificação por *token* (*Identity Token*, ID Token); *endpoint* de informação do usuário para conseguir mais informações específicas do usuário, um conjunto de padrões de escopos e uma implementação padronizada. A troca de mensagens do OIDC entre os participante se dá por meio do REST e o formato da estrutura da mensagem é o JSON (BARBETTINI, 2017). Na Tabela 7 é feita uma comparação entre a diferença entre o OAuth 2.0 e o OIDC (BARBETTINI, 2017):

Tabela 7– Comparação entre OAuth 2.0 e OIDC.

OAuth 2.0	OIDC
Concede acesso para sua API	Login do usuário
Consegue acesso para os recursos do usuário	Faz um conta ser autenticada em mais de um sistema
Autorização	Autenticação

ENTIDADES DO OPENID CONNECT

As entidades envolvidas no processo OIDC são: o Provedor de Identidade (*Identity Provider*, IdP ou OP) e a Parte Confiante (*Rely Party*, RP) e o Usuário Final

(*Final User*, FU), Figura 10.

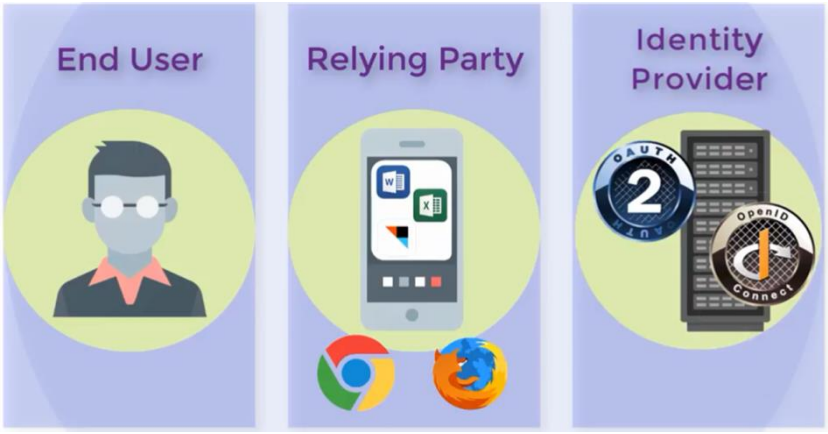

Figura 10 – Entidades do OpenID Connect. Fonte: (ORACLE, 2017).

Provedor de identidade. Assegura que o usuário final é autenticado e fornece *claims* sobre o usuário final e informa eventos de autenticação para a Parte Confiante.

Parte Confiante. É aplicação cliente que delega a função de autenticação do usuário para Provedor de Identidade.

Usuário Final. É a entidade que representa uma informação de sua identidade a qual Parte Confiante requisita para o Provedor de Identidade.

TOKEN DE IDENTIDADE E CLAIMS

O Provedor de Identidade fornece informação, conhecida como *Token* de Identidade (*Identity Token*, ID Token), para a Parte Confiante a respeito da identidade do usuário final. O ID Token é similar ao um identificador ou passaporte. O ID Token contém um conjunto de atributos requeridos ou *claims* sobre o usuário final e também como o usuário final foi autenticado. De qualquer maneira, ID Token é codificado como JSON Web *Token* ou JWT, Figura 11.

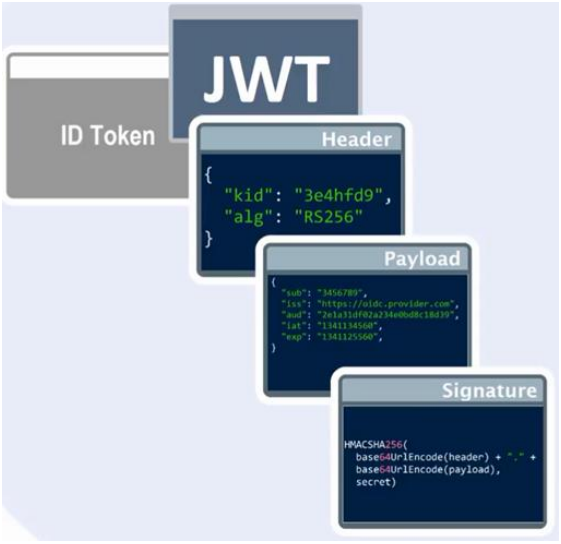

Figura 11 – Estrutura do ID Token. Fonte: (ORACLE, 2017).

Na Figura 11, o ID Token pode ser digitalmente assinado usando a Assinatura Web JSON (*JSON Web Signature*, JWS)[80] para conseguir integridade e não-repúdio. Além disso, o cabeçalho (*header*), conteúdo da mensagem (payload) e assinatura (*signature*) são combinados dentro de um JWT e também podem ser criptografados com Criptografia Web JSON (JSON *Web Encryption*, JWE)[81] para confidencialidade.

O conjunto de atributos, *claims*, forma o corpo da mensagem ID Token. No corpo da estrutura ID Token está contido os atributos Autoridade emissora (*Issuer Authorithy*), Público-Alvo (*Audience*), Data de Emissão (*Issue Date*) e Data de Expiração (Expired Date).

Figura 12 – *Claims* do ID Token. Fonte: (ORACLE, 2017).

Na Figura 12, há também um número de *claims* opcionais que podem ajudar

[80] https://tools.ietf.org/html/rfc7515
[81] https://tools.ietf.org/html/rfc7516

parte confiável a validar o ID Token tempo de autenticação (*Authentication Time*) e *nonce*. Além disso, o ID Token pode também contar com atributos *claims* adicionais como nome e endereço de e-mail.

- **Sujeito**: Atribuído para um usuário pelo provedor de Identidade;
- **Autoridade emissora**: Provedor de identidade que emite o *token*;
- **Público-Alvo**: Identifica a parte confiante a quem pode usar este o *token*;
- **Data de Emissão**: Data e tempo que o *token* foi emitido;
- **Data de Expiração**: Data e tempo que o *token* foi expirado;
- **Tempo de Autenticação**: Mostra o tempo que aplicação cliente foi autenticada;
- ***Nonce***: Valores que podem ser usados para mitigar ataques de replicação.

ESCOPOS

Além de usar *claims* para estrutura a informação da identidade, o OIDC também podem usar escopos. O OIDC contém quatro escopos padrões que são usados para fornecer à aplicação cliente com informação detalhado do usuário que são (“`openid`”), (“`profile`”), (“`email`”), (“`address`”), (“`phone`”). Por exemplo, as requisições de acesso para usuários finais são utilizando, por padrão, o escopo (“`profile`”).

Por meio de uma requisição inicial de autenticação, a aplicação cliente solicita os escopos para serem retornados no ID Token. Alternativamente, o ID Token pode ser solicitado usando uma credencial de acesso, *token* de acesso, através de chamadas API REST para o *endpoint* de informação do usuário.

ENDPOINT

O provedor de identidade oferece os *endpoints* de autorização, de *token* e de informação do usuário em que o cliente ou parte confiante podem realizar as suas requisições:

Autorização. É onde o usuário final é perguntado para autenticar e conceder acesso à aplicação cliente. Uma vez o consentimento é dado, o *endpoint* de autorização passa de volta um código de autorização para o cliente.

Token. Realiza a troca do código de autorização que veio a partir do *endpoint* de Autorização tendo como resposta um ID Token, um *token* de acesso e *refresh token*.

Informação do Usuário. Retorna a informação do usuário ou conjunto de *claims* para aplicação cliente, baseado no escopo de acesso em conjunto com *token* de acesso válido para que a informação do usuário ou conjunto de *claims* seja apresentada.

SAML

Linguagem de marcação de verificação segura (*Security Assertion Markup Language*, SAML) é um protocolo de autenticação que muito comum usado dentro de intranet de websites no mercado empresarial (OWASP, 2019). SAML contém as seguintes características: é baseada em XML, permite SSO que não se inicia pelo provedor de serviço.

FRAMEWORK DE AUTENTICAÇÃO UNIVERSAL

O Framework de Autenticação Universal (*Universal Authentication Framework,* UAF) é um protocolo Identidade criada pela Aliança de Identidade Rápida Online (FAST *Identity Online*, FIDO) baseado sobre modelo desafio-resposta de criptografia de chave pública (OWASP, 2019). O UAF foca na autenticação sem *password*. O UAF toma as vantagens de sensores impressão digital (*fingerprint*), câmeras (biometria da face), microfones (biometria da voz) entre outros. UAF trabalha tanto com aplicações nativas como com aplicações web

FRAMEWORK DE SEGUNDO FATOR

O Framework de Autenticação Universal (*Universal Second Factor ,* U2F) é um protocolo Identidade criada também pela Aliança de Identidade Rápida Online(FAST *Identity* Online, FIDO) baseado sobre modelo desafio-resposta de criptografia de chave pública (OWASP, 2019).

U2F permite autenticação baseado em senha usando *token* em hardware (ex. tipicamente USB) que armazena chaves criptográficas de autenticação e as usa para assinar. O usuário pode usar o mesmo *token* como segundo fator para aplicações

múltiplas. U2F trabalha bem com aplicações Web.

APÊNDICE H — RELAÇÃO NÃO CONFORMIDADE A AMEAÇAS

Na Tabela 8, é mostrado a as restrições do domínio das ameaças relacionadas aos ataques de personificação e replicação para verificar a conformidade sobre o modelo de comunicação desenvolvido das soluções.

Tabela 8 – Restrições de ameaças relacionadas sobre os ataques de personificação e replicação

Ataque	Nome da Ameaça	Conformidade
Replicação	Obtendo segredo do cliente	(flow is [Recupere Credencial Autorizacao] or flow is [Solicita Credencial Autorizacao] or flow is [Solicita Credencial Identificacao] or flow is [Recupere Credencial Identificacao] or flow is [Solicita Credencial Acesso] or flow is [Recupere Credencial Acesso]) and ((flow.[Token] is 'Yes') or (flow.[Code] is 'Yes') or (flow.[Refresh Token] is 'Yes')) and ((flow.[Source Authenticated] is 'No') or (flow.[Provides Confidentiality] is 'No')) and ((flow.[CWT ID_TOKEN] is 'No') or (flow.[COSE Layer] is 'No'))
Replicação	Replicação de solicitações ao servidor de recurso e o servidor de autorização	(flow is [Solicita Credencial Autorizacao] or flow is [Solicita Credencial Identificacao] or flow is [Solicita Credencial Acesso]) and ((flow.[Token] is 'Yes'

		) or (flow.[Code] is 'Yes') or (flow.[Refresh Token] is 'Yes')) and ((flow.[Source Authenticated] is 'No') or (flow.[Provides Confidentiality] is 'No')) and ((flow.[CWT ID_TOKEN] is 'No') or (flow.[COSE Layer] is 'No')) and ((flow.[CWT ID_TOKEN] is 'No') or (flow.[Chalenge Response] is 'No'))
Replicação	Confidencialidade insegura	(flow is [Recupere Credencial Autorizacao] or flow is [Solicita Credencial Autorizacao] or flow is [Solicita Credencial Identificacao] or flow is [Recupere Credencial Identificacao] or flow is [Solicita Credencial Acesso] or flow is [Recupere Credencial Acesso]) and ((flow.[Token] is 'Yes') or (flow.[Code] is 'Yes') or (flow.[Refresh Token] is 'Yes')) and ((flow.[Source Authenticated] is 'No') or (flow.[Provides Confidentiality] is 'No')) and ((flow.[CWT ID_TOKEN] is 'No') or (flow.[COSE Layer] is 'No'))
Replicação	Usar maior tempo de expiração	(flow is [Transfere Token Acesso]) and (flow.[Token] is 'Yes') and (flow.[CWT

		ID_TOKEN] is 'No')
Replicação	Não limitar o número de requisições por uso ou uso único sobre o tempo	(flow is [Solicita Credencial Autorizacao] or flow is [Solicita Credencial Identificacao] or flow is [Solicita Credencial Acesso]) and (flow.[Token] is 'Yes') and (flow.[CWT ID_TOKEN] is 'No')
Replicação	Replicação de tokens para o servidor de recurso particular	(flow is [Solicita Credencial Acesso]) and (flow.[Token] is 'Yes') and (flow.[CWT ID_TOKEN] is 'No')
Replicação	Emitir segredos de cliente específicos da instalação	(flow is [Solicita Credencial Autorizacao] or flow is [Emite Credencial Acesso]) and (flow.[Client Credential] is 'Yes') and (flow.[CWT ID_TOKEN] is 'No')
Replicação	Replicação do código de autorização	(flow is [Solicita Credencial Autorizacao] or flow is [Emite Credencial Autorizacao]) and (flow.[Code] is 'Yes') and (flow.[CWT ID_TOKEN] is 'No')
Replicação	Requisições não assinadas	(flow is [Solicita Credencial Autorizacao] or flow is [Solicita Credencial Identificacao] or flow is [Solicita Credencial Acesso]) and (flow.[Signed Communication] is 'No')
Personificação	Vazamento da credencial do cliente durante a	(flow is [Recupere Credencial Autorizacao] or

	transmissão	flow is [Solicita Credencial Autorizacao] or flow is [Solicita Credencial Identificacao] or flow is [Recupere Credencial Identificacao] or flow is [Solicita Credencial Acesso] or flow is [Recupere Credencial Acesso]) and ((flow.[Token] is 'Yes') or (flow.[Code] is 'Yes') or (flow.[Refresh Token] is 'Yes')) and ((flow.[Source Authenticated] is 'No') or (flow.[Provides Confidentiality] is 'No')) and ((flow.[CWT ID_TOKEN] is 'No') or (flow.[COSE Layer] is 'No'))
Personificação	Personificação Sessão do usuário	(flow is [Solicita Credencial Acesso] or flow is [Recupere Credencial Acesso]) and ((flow.[Token] is 'Yes') or (flow.[Code] is 'Yes') or (flow.[Refresh Token] is 'Yes')) and ((flow.[Source Authenticated] is 'No') or (flow.[Provides Confidentiality] is 'No')) and ((flow.[CWT ID_TOKEN] is 'No') or (flow.[COSE Layer] is 'No'))
Personificação	Personificação Dono do Recurso	(flow is [Solicita Consentimeno Dono Recurso] or flow is [Retorna Consentimento Dono do

		Recurso])
Personificação	Combinação diferente dos atributos	(flow is [Solicita Credencial Autorizacao] or flow is [Solicita Credencial Identificacao] or flow is [Solicita Credencial Acesso]) and flow.[Client Bind Redirect URI] is 'No'

APÊNDICE I — CASOS DE USO EM AUTOMAÇÃO RESIDENCIAL

As ameaças e os ataques apresentados afetam soluções baseadas no OAuth 2.0 e o desenvolvimento do ACE-OAuth as quais podem ser encontradas no ambiente de automação residencial em IoT. Dessa forma, soluções baseada em ACE-OAuth precisam estar em conformidade com os casos de uso de automação residencial. Seitz & Gerdez (2016) propõem casos de uso para ambiente de automação residencial em IoT. Esses casos de uso podem contribuir no entendimento dos requisitos de segurança para construção de aplicações adequadas para ambiente de automação de IoT. Seitz & Gerdez (2016) elenca os seguintes caso de uso:

- Configurar autorização de acesso remoto
- Aplicar diferentes direitos de acesso para diferentes usuários
- Conceder acesso de permissão para alguém durante uma janela de tempo
- Dispositivos precisam ser capazes de se comunicar seguramente com os dispositivos de controle
- Usuários autorizados querem ser capazes de acesso com mínimo esforço
- Prevenir o acesso não autorizado de entidades capazes de deduzir os perfis comportamentais dos dispositivos na rede local

Depois de observados os casos de uso para automação residencial é importante detalhar as propriedade desse ambiente. Neste caso, sabe-se que numa residência, vários dispositivos (aquecimento, refrigeração, ventilação e luminosidade) podem ser controlados localmente pelos proprietários como Alice e Bob, Figura 13. Os proprietários podem acionar seus dispositivos, desde que provem sua identidade para o controle de acesso e uma vez liberado também confirmem seu o direito de acesso para acionar as funcionalidades do dispositivo.

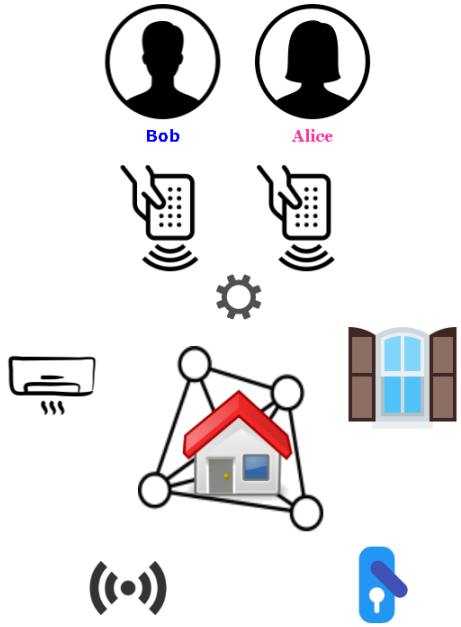

Figura 13 - Acesso local dos dispositivos no ambiente de automação residencial em IoT.

Contudo, outro caso pode ser percebido, é quando os proprietários da casa desejam remotamente acessar e controlar os dispositivos dentro da residência, Figura 14. Nota-se que residência pode ficar vulnerável ao acesso externo, se não usar algum mecanismo de controle de acesso para impedir a invasão e o ataque externo sobre os dispositivos de IoT.

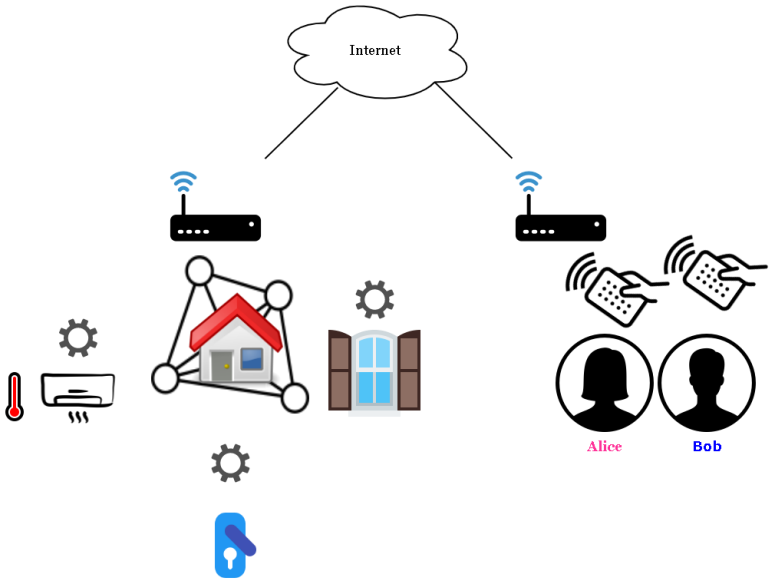

Figura 14 – Acesso remoto dos dispositivos do ambiente de automação residencial.

Dessa forma, na Figura 15, para aplicar o controle de acesso os usuários da residência é importante atribuir perfis aos usuários que possam ter autorização aos dispositivos (SEITZ; GERDES, 2016). Dessa forma, atribuição dos perfis ao usuário pode ser dividida em duas classes de usuários, a classe regular, habitantes, não moradores confiáveis e classe heterogênica e dinâmica, como visitantes (operadores de serviço e conhecidos) (SEITZ; GERDES, 2016). Dessa forma Figura 15 é possível estabelecer níveis de confiança e de direitos de acesso a depender da categoria.

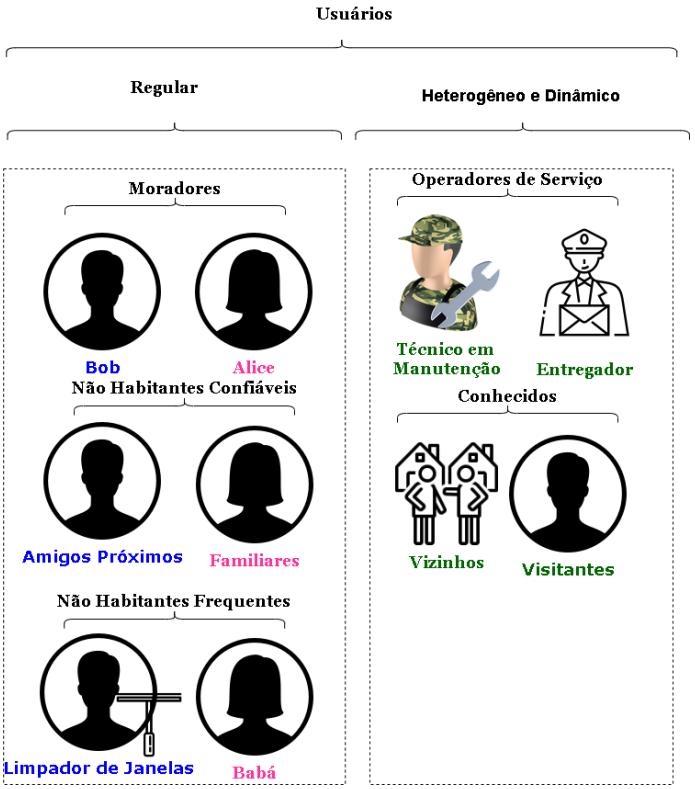

Figura 15 – Perfil dos usuários no ambiente de automação residencial.

Entretanto um terceiro caso de uso ocorre quando os parentes de um dos moradores que não tem autorização desejam acionar os dispositivos, Figura 16. Na Figura 16, para permitir a entrada dos seus parentes, Alice concede autorização para que os seus pais possam usufruir das funcionalidades dos dispositivos de IoT.

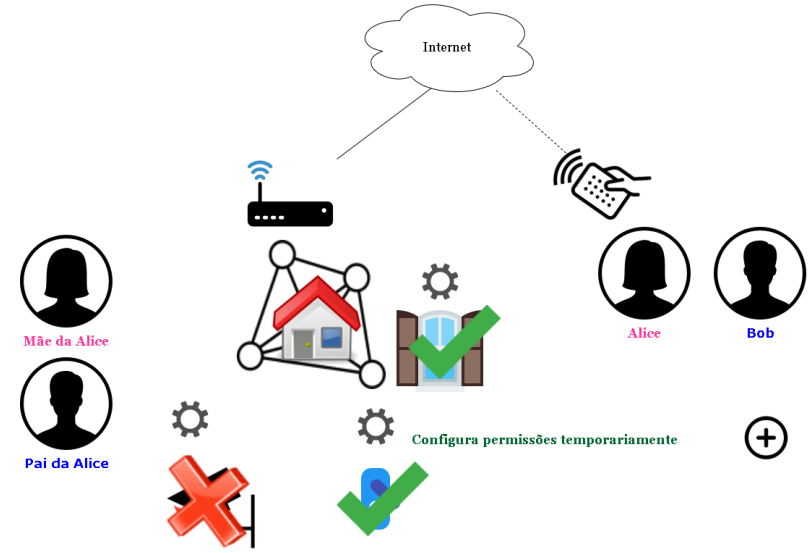

Figura 16 – Permissão realizada por Alice para seus pais.

Todavia, um quarto caso de uso, pode haver situações que necessitem revogação autorização para realizar alguma tarefa de manutenção em casa ou realizar algum outro tipo de serviço, Figura 17. Na Figura 17, como o técnico de manutenção tem um menor grau de confiança baseada no seu perfil, desse modo o técnico de manutenção não pode acionar todas as funcionalidades dos dispositivos.

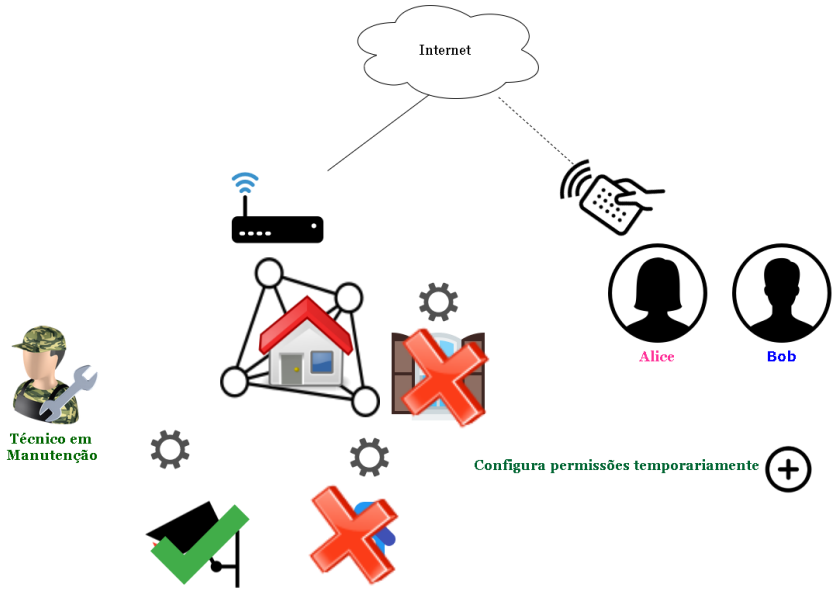

Figura 17– Usuário técnico de manutenção acessa o dispositivo.

A Figura 18 mostra o modelo de ameaça em que um atacante em potencial se aproveita das vulnerabilidades do ambiente de automação residencial. Por exemplo, Além disso, o atacante pode espionar ou interceptar a comunicação e adquirir a

credencial de acesso para se passar por um dos proprietários e acionar os dispositivos consentimento dos proprietários, se não houver controle de acesso mais rígido.

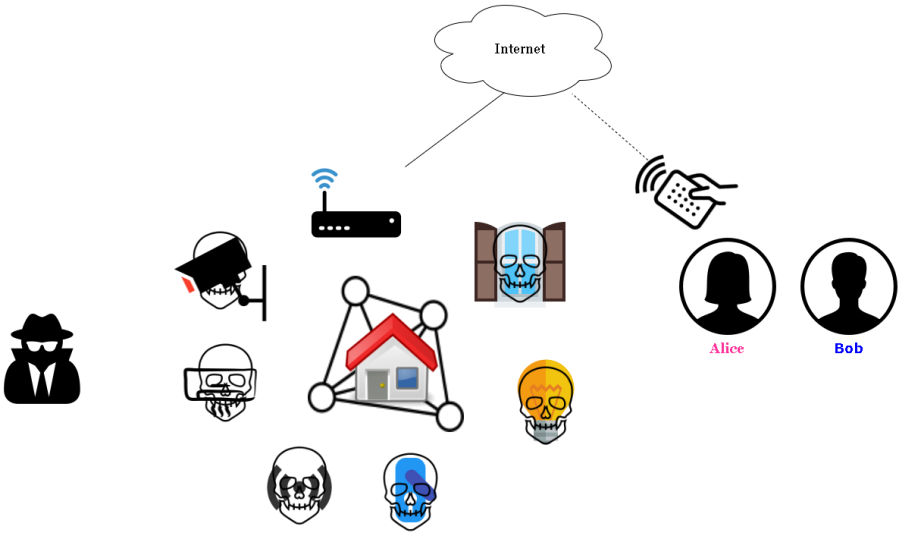

Figura 18– Investida do atacante no ambiente em IoT.