UNIVERSIDADE FEDERAL FLUMINENSE

DANILO FERNANDES DE ASSIS

Uma análise da segurança do padrão IEEE 2030.5

NITERÓI

2021

UNIVERSIDADE FEDERAL FLUMINENSE

DANILO FERNANDES DE ASSIS

Uma análise da segurança do padrão IEEE 2030.5

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

Orientador:
Diego Gimenez Passos

NITERÓI

2021

Ficha catalográfica automática - SDC/BEE
Gerada com informações fornecidas pelo autor

A848a Assis, Danilo Fernandes de
Uma análise da segurança do padrão IEEE 2030.5 / Danilo Fernandes de Assis ; Diego Gimenez Passos, orientador. Niterói, 2021.
147 f.

Dissertação (mestrado)-Universidade Federal Fluminense, Niterói, 2021.

DOI: http://dx.doi.org/10.22409/PGC.2021.m.00429303165

1. Segurança da informação. 2. Rede elétrica inteligente. 3. Internet das coisas. 4. Produção intelectual. I. Passos, Diego Gimenez, orientador. II. Universidade Federal Fluminense. Instituto de Computação. III. Título.

CDD -

Bibliotecário responsável: Debora do Nascimento - CRB7/6368

DANILO FERNANDES DE ASSIS

Uma análise da segurança do padrão IEEE 2030.5

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Computação da Universidade Federal Fluminense como requisito parcial para a obtenção do Grau de Mestre em Computação. Área de concentração: Ciência da Computação.

Aprovada em março de 2021.

BANCA EXAMINADORA

Prof. Diego Gimenez Passos, D.Sc. – Orientador, UFF

Profª. Débora Christina Muchaluat Saade, D.Sc. – UFF

Prof. Marcelo Gonçalves Rubinstein, D.Sc. – UERJ

Niterói

2021

A minha amada mãe – Vilma Fernandes.

Agradecimentos

Agradeço a Deus por me dar força e coragem para iniciar e terminar essa caminhada. Por não permitir que eu fraquejasse e por me dar forças para sustentar até o fim.

Agradeço a minha mãe por sempre acreditar em mim e na minha capacidade. Tudo que sou e tenho é graças a ela. Sua fé e dedicação em proporcionar sempre o melhor moldou a pessoa que hoje sou. A ti, minha mãe, meu imenso muito obrigado.

Agradeço muito a minha amada esposa Nayara por permanecer ao meu lado em todos os momentos dessa caminhada. Obrigado por compreender a minha ausência devido à dedicação ao mestrado e pelas palavras de apoio e incentivo nos momentos mais difíceis. Sem você, tudo seria muito mais difícil.

Agradeço ao meu orientador Diego pela paciência, dedicação e pelos muitos ensinamentos proporcionados. Professor detentor de didática e conhecimento ímpar, é um exemplo a ser seguido. Obrigado por acreditar em mim e por aceitar o desafio da orientação.

Agradeço ao professor Célio pela conversa inicial sobre o programa de pós-graduação e por ter me recebido na UFF de forma amistosa.

Por fim, gostaria de agradecer à Marinha do Brasil por conceder a oportunidade de cursar o mestrado.

"O período de maior ganho em conhecimento e experiência é o período mais difícil da vida de alguém."

(Dalai Lama)

"Be patient and tough; someday this pain will be useful to you."

(Ovid - Elegy XI)

Resumo

As rápidas evolução e disseminação da Internet das Coisas (IoT) e das fontes de energia elétrica distribuídas — *Distributed Energy Resources* (DER) — estão trazendo ao mercado diversos dispositivos elétricos inteligentes, tanto no âmbito residencial quanto industrial, principalmente medidores e inversores.

Como resultado, surge também a necessidade de uma padronização a fim de promover interoperabilidade, segurança e comunicação eficiente entre os diferentes tipos e fabricantes de dispositivos. Nesse contexto, surge o padrão IEEE 2030.5-2018 que visa atender a essas demandas bem como à crescente demanda das concessionárias de energia elétrica e dos consumidores em gerenciarem a produção e o consumo de eletricidade de forma mais eficiente.

O padrão IEEE 2030.5-2018 define uma camada de aplicação no topo de uma pilha TCP/IP para permitir o gerenciamento do ambiente elétrico do usuário. Além disso, ele também define os mecanismos a serem utilizados para a troca de mensagens e os recursos de segurança para proteger tais mensagens. Dentre os recursos de segurança, destaca-se uma infraestrutura de chave pública própria cujos certificados digitais possuem a característica específica de não expiração e não revogação.

Sendo a segurança um requisito fundamental para o gerenciamento dos dispositivos e para o tráfego de dados sensíveis, este trabalho tem por finalidade realizar uma análise deste aspecto do padrão IEEE 2030.5-2018, onde destacamos potenciais lacunas de segurança — em especial, a decorrente da falta de suporte para a revogação e expiração dos certificados digitais provenientes da infraestrutura de chave pública própria. Também é realizada uma prova de conceito onde são demonstrados na prática possíveis ataques explorando tal lacuna de segurança.

Além disso, este trabalho também descreve o padrão fornecendo uma visão geral apresentando aspectos relacionados a arquitetura, topologia e a forma de comunicação entre os dispositivos.

Palavras-chave: *Distributed Energy Resources*; IEEE 2030.5; interoperabilidade; Internet das coisas; IoT; segurança.

Abstract

The fast evolution and dissemination of the Internet of Things (IoT) and of Distributed Energy Resources (DER) are bringing to the market several smart electrical devices, both in the residential and industrial spheres, mainly meters and inverters.

As a result, there is also a need for standardization in order to promote interoperability, security and efficient communication between different types and manufacturers of devices. In this context, the IEEE 2030.5-2018 standard emerges, which aims to meet these demands as well as the growing demand from electricity utilities and consumers to manage electricity production and consumption more efficiently.

The IEEE 2030.5-2018 standard defines an application layer on top of a TCP/IP stack to allow management of the user's electrical environment. In addition, it also defines the mechanisms to be used for the exchange of messages and the security features to protect such messages. Among the security features, the standard's public key infrastructure stands out by working with certificates that do not expire and cannot be revoked.

Since security is a fundamental requirement for the management of devices and for the traffic of sensitive data, this work aims to perform an analysis of this aspect of the IEEE 2030.5-2018, highlighting potential security gaps — in particular, due to the lack of support for the revocation and expiration of digital certificates from the public key infrastructure itself. A proof of concept is also carried out where possible attacks are demonstrated in practice exploiting such a security gap.

In addition, this work also describes the standard providing an overview presenting aspects related to architecture, topology and how devices communicate.

Keywords: Distributed Energy Resources; IEEE 2030.5; interoperability; Internet of Things; IoT; security.

Lista de Figuras

Lista de Tabelas

Lista de Abreviaturas e Siglas

AC	*Alternating Current*	2
ACL	*Access Control List*	45
CA	*Certificate Authority*	48
CRL	*Certificate Revoked List*	49
CSIP	*Common Smart Inverter Profile*	2
DC	*Direct Current*	2
DER	*Distributed Energy Resource*	1
DNS	*Domain Name System*	27
DNS-SD	DNS *Service Discovery*	27
DoS	*Denial of Service*	60
DRLC	*Demand Response and Load Control*	40
EXI	*Efficient XML Interchange*	22
GFEMS	*Generating Facility Energy Management System*	17
HAN	*Home Area Network*	3
HEMS	*Home Energy Management System*	14
HTTP	*Hypertext Transfer Protocol*	21
IDS	*Intrusion Detection System*	12
IoT	*Internet of Things*	5
LFDI	*Long Form Device Identifier*	43
MCA	*Manufacturer CA*	50
mDNS	*Multicast* DNS	27
MICA	*Manufacturer Issuing CA*	50
OCSP	*Online Certificate Status Protocol*	49
PKI	*Public Key Infrastructure*	48

Sumário

Capítulo 1

Introdução

1.1 Contextualização

O uso de fontes renováveis de energia nas residências, principalmente a solar, vem alterando a forma de produção e distribuição da energia elétrica e aumentando significativamente a chamada *Distributed Energy Resource* (DER) [1]. Com o uso de fontes renováveis de energia, as residências passam a gerar energia, disponibilizando o excedente para a rede elétrica, ao invés de apenas consumi-la. Nesse cenário, o fluxo de energia elétrica não é mais unidirecional (da rede elétrica da concessionária para a residência). Em vez disso, torna-se multidirecional: da rede da concessionária para a residência, da residência para a rede da concessionária, ou até mesmo de residência para residência. Como um exemplo prático desse conceito, o estado da Califórnia atualmente lidera nos Estados Unidos a produção de energia distribuída com 1.201.064 projetos solares capazes de produzir um total de 9.106 Megawatts (MW)[1] [11].

A estrutura atual do sistema elétrico é projetada para fluxo de energia unidirecional [1, 56]. Os fluxos de energia multidirecionais requerem uma interconexão de dados entre a rede da concessionária e a DER. Essa interconexão deve ser segura, eficiente e deve permitir um fluxo bidirecional de informação de forma a possibilitar o controle e o gerenciamento do fluxo de energia em um determinado momento para evitar falhas e sobrecargas.

Entre os vários programas de interconexão para geração distribuída de eletricidade na Califórnia está o *California's Rule 21* [9]. Ele estabelece um conjunto de requisitos de interconexão, operação e medição para fontes geradoras distribuídas se conectarem

[1] Dados de 30-11-2020.

à rede elétrica [8]. Esta interconexão requer o uso de *inversores* — dispositivos que convertem corrente contínua (*Direct Current* (DC)) em corrente alternada (*Alternating Current* (AC)). Além de realizar a conversão DC para AC, alguns inversores também podem oferecer funcionalidades adicionais e avançadas. Esses inversores são chamados de inversores inteligentes (*smart inverters*). Essas funcionalidades, acessadas via interface de comunicação, permitem o gerenciamento preciso da produção, consumo e disponibilidade do excedente de energia. Elas permitem que as DERs modifiquem seus parâmetros e se adaptem de acordo com as condições observadas localmente [12].

Visando o desenvolvimento das funcionalidades avançadas nos inversores, foi criado o *Smart Inverter Working Group* (SIWG) — uma colaboração entre a Comissão de Energia da Califórnia e a Comissão de Serviços Públicos da Califórnia. O SIWG identificou o desenvolvimento das funcionalidades avançadas como uma importante estratégia para mitigar o impacto da alta penetração das DERs [10]. O plano de desenvolvimento das funcionalidades foi dividido em três fases [10]:

- **Fase 1 – Funções Autônomas**: definir as funções que o inversor conectado a DER deveria executar.
- **Fase 2 – Protocolos de Comunicação**: definir o protocolo de comunicação a ser utilizado entre as concessionárias e as DERs.
- **Fase 3 – Funcionalidades Avançadas**: considerar funcionalidades avançadas a serem implementadas nos inversores.

O *California's Rule 21* definiu que a comunicação entre as concessionárias e as DERs deve utilizar o padrão IEEE 2030.5. O padrão pode ser utilizado para prover a comunicação de uma ampla gama de dispositivos e serviços tais como termostatos inteligentes, programas de resposta à demanda, medidores inteligentes, carregamento de veículos elétricos e inversores inteligentes. O particular uso do padrão pelo *California's Rule 21* foi consolidado no *Common Smart Inverter Profile* (CSIP) *IEEE 2030.5 Implementation Guide for Smart Inverters* [15].

O IEEE 2030.5 faz parte do IEEE 2030 que é um guia que fornece abordagens alternativas e melhores práticas para alcançar a interoperabilidade em redes inteligentes do sistema de energia elétrica [27]. Ele apresenta um modelo de referência de interoperabilidade de redes inteligentes cujo objetivo é fornecer às partes interessadas um entendimento comum dos critérios de interoperabilidade do ponto de vista do sistema elétrico, da tecnologia de comunicação e da tecnologia da informação [27]. Alguns exemplos das séries

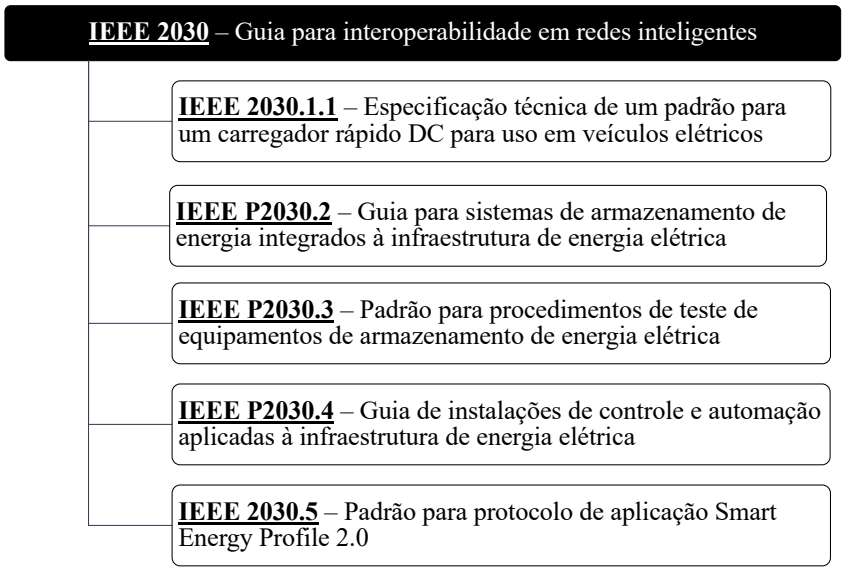


Figura 1.1: Exemplo de projetos no contexto do IEEE 2030.

de projetos do IEEE 2030 para interoperabilidade de redes inteligentes podem ser vistos na Figura 1.1.

O padrão IEEE 2030.5 é uma evolução do *ZigBee Smart Energy Profile 1.x* (SEP 1) conforme mostrado na Figura 1.2. O *Smart Energy Profile* (SEP) foi inicialmente desenvolvido pela ZigBee Alliance em 2007 para prover interoperabilidade entre os dispositivos ZigBee presentes em uma rede inteligente residencial — frequentemente chamada de *Home Area Network* (HAN). Desde então, passou por várias revisões e atualizações. No entanto, o SEP 1 está limitado ao uso da tecnologia e da pilha de protocolos ZigBee (incluindo o IEEE 802.15.4). Em 2008, iniciou-se o desenvolvimento do *Smart Energy Profile 2.0*

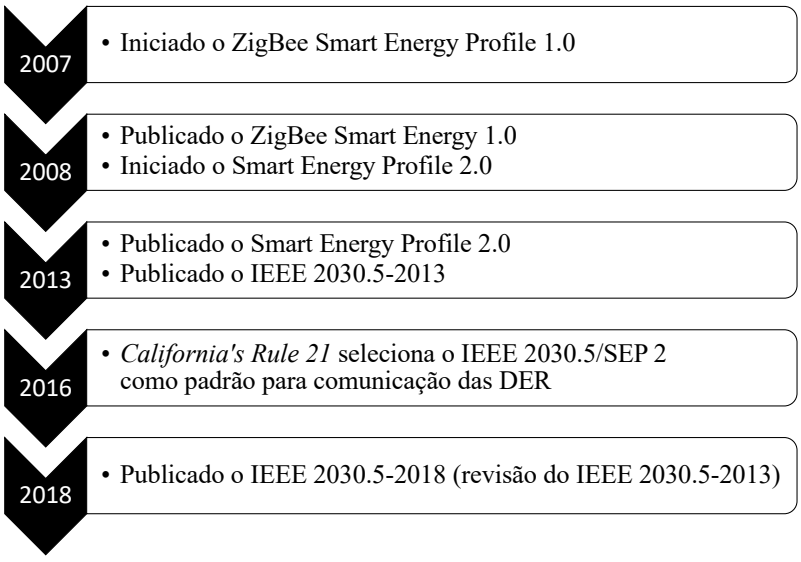


Figura 1.2: Evolução do padrão IEEE 2030.5.

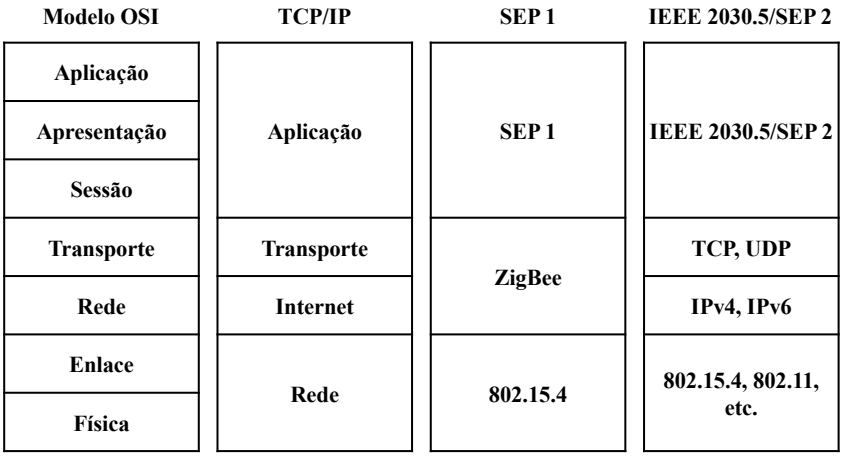


Figura 1.3: Comparação entre as pilhas de rede ISO/OSI, TCP/IP, SEP 1 e IEEE 2030.5/SEP 2 (adaptado de [63]).

(SEP 2) que, evoluindo do SEP 1, foi projetado para fazer uso de tecnologias de comunicação amplamente difundidas que suportam o *Internet Protocol* (IP) tornando-se assim agnóstico à tecnologia de comunicação subjacente, conforme visto na Figura 1.3. Em 2011, HomePlug Alliance, Wi-Fi Alliance, HomeGrid Forum e ZigBee Alliance formaram o Consórcio para Interoperabilidade do *Smart Energy Profile 2* (CSEP) com o objetivo de criar testes de conformidade para promover a certificação dos dispositivos que fazem uso do SEP 2. Em 2013, o SEP 2 foi formalmente publicado pela ZigBee Alliance e HomePlug Powerline Alliance. No mesmo ano, o IEEE adotou o SEP 2 como o padrão IEEE 2030.5-2013, mantendo toda a especificação original. Em 2016, o *California's Rule 21* definiu o IEEE 2030.5 (SEP 2) como o padrão a ser utilizado na comunicação das DERs. Posteriormente, o IEEE começou a trabalhar em uma revisão do IEEE 2030.5-2013 visando dar suporte total ao IEEE 1547 (padrão para interconectar recursos distribuídos com sistemas de energia elétrica), além de eliminar erros e ambiguidades no IEEE 2030.5-2013. A revisão deu atenção especial às atividades do *California's Rule 21* [33]v e foi publicada em 2018. Nesse ponto, o padrão foi formalmente adotado pelo IEEE como o IEEE 2030.5-2018.

Mesmo com a adoção do IEEE 2030.5 pelo *California's Rule 21* e sua recente revisão, as referências ao mesmo encontradas na literatura científica são escassas e fornecem apenas uma abordagem superficial do padrão, sem promover um estudo detalhado quanto a segurança, estrutura e arquitetura. Tais referências possuem como foco principal outros tópicos relacionados, como, por exemplo, os trabalhos [39, 23] que realizam uma implementação do padrão para fins de avaliação de desempenho da comunicação. Por outro lado, o impacto do padrão na sociedade moderna deve ser muito representativo, pois

ele altera substancialmente a relação entre clientes e concessionárias de fornecimento de energia elétrica.

1.2 Objetivo

A adoção do IEEE 2030.5 no projeto de comunicação das DERs na Califórnia confirma sua notoriedade e relevância frente à comunicação das DERs e também na comunicação dos dispositivos da chamada *Internet of Things* (IoT) [3] presentes em uma HAN. No entanto, há uma lacuna de conhecimento que existe pela falta de um estudo científico que promova uma análise de segurança e aborde questões relacionadas ao modo como os recursos de segurança são concebidos e utilizados para prover a comunicação e o gerenciamento dos dispositivos.

O objetivo deste trabalho é justamente preencher essa lacuna de forma a promover um estudo amplo, detalhado e atualizado quanto à segurança e à confiabilidade na comunicação do padrão IEEE 2030.5-2018. O estudo visa analisar os recursos de segurança utilizados pelo padrão e trazer à tona potenciais lacunas de segurança advindas principalmente pela falta de suporte para revogação e expiração de certificados digitais provenientes da infraestrutura de chave pública própria do IEEE 2030.5. Uma prova de conceito é realizada com o intuito de apresentar de forma prática possíveis ataques relacionados com a lacuna de segurança. O trabalho também aborda de forma geral aspectos arquiteturais, estruturais e de topologia do padrão a fim de fornecer uma visão geral do mesmo.

1.3 Organização

O resto do texto está organizado da seguinte forma. O Capítulo 2 faz uma revisão da literatura sobre os trabalhos relacionados ao padrão IEEE 2030.5. O Capítulo 3 apresenta possíveis aplicações e topologias de comunicação usadas para interconectar os dispositivos das concessionárias e dos consumidores finais. O Capítulo 4 apresenta e descreve os elementos que constituem o padrão. No Capítulo 5, são discutidas as técnicas de segurança adotadas pelo padrão, com atenção especial aos aspectos de autenticação, incluindo uma descrição da própria infraestrutura de chave pública do IEEE 2030.5. Nele, também é realizada uma demonstração prática da falha referente à lacuna de segurança existente na infraestrutura de chave pública do IEEE 2030.5. Por fim, no Capítulo 6 são apresentadas

as considerações finais.

Capítulo 2

Revisão da Literatura

Esse capítulo tem por finalidade fornecer uma visualização das tendências de estudo no âmbito do padrão IEEE 2030.5. Além disso, também permite identificar possíveis lacunas e aspectos menos discutidos, o que possibilita o desenvolvimento de novas pesquisas.

Para isso, é necessária uma pesquisa na literatura científica. Tal pesquisa busca selecionar os trabalhos mais pertinentes sobre o padrão IEEE 2030.5 a fim de que eles forneçam os subsídios e os dados necessários. De posse dos trabalhos, e aplicando os devidos filtros, é possível extrair as informações que irão ajudar a consolidar o conhecimento desejado.

Para selecionar os trabalhos científicos mais pertinentes, foram utilizadas as bases de conteúdos acadêmicos da *Scopus*, *IEEE Xplorer* e *Google Scholar*. Em cada base, foi realizada uma pesquisa fazendo uso da *string* de busca *"IEEE 2030.5"*. A escolha de tal *string* se deve ao fato de que é a nomenclatura oficial utilizada pelo IEEE em relação ao padrão. Os resultados, com os respectivos quantitativos de trabalhos retornados por cada base, são apresentados na Tabela 2.1.

Pela leitura da tabela é perceptível a discrepância no quantitativo de trabalhos retornados pela base do *Google Scholar* em relação às outras duas bases pesquisadas. Visando dar mais equidade quanto aos resultados retornados e procurando eliminar possíveis falhas quanto à pesquisa realizada, foram definidos filtros de seleção e exclusão a serem

Tabela 2.1: Número de trabalhos retornados na pesquisa.

Base de busca	***String* de busca**	**Trabalhos retornados**
Scopus	"IEEE 2030.5"	19
IEEE *Xplorer*	"IEEE 2030.5"	12
Google Scholar	"IEEE 2030.5"	198

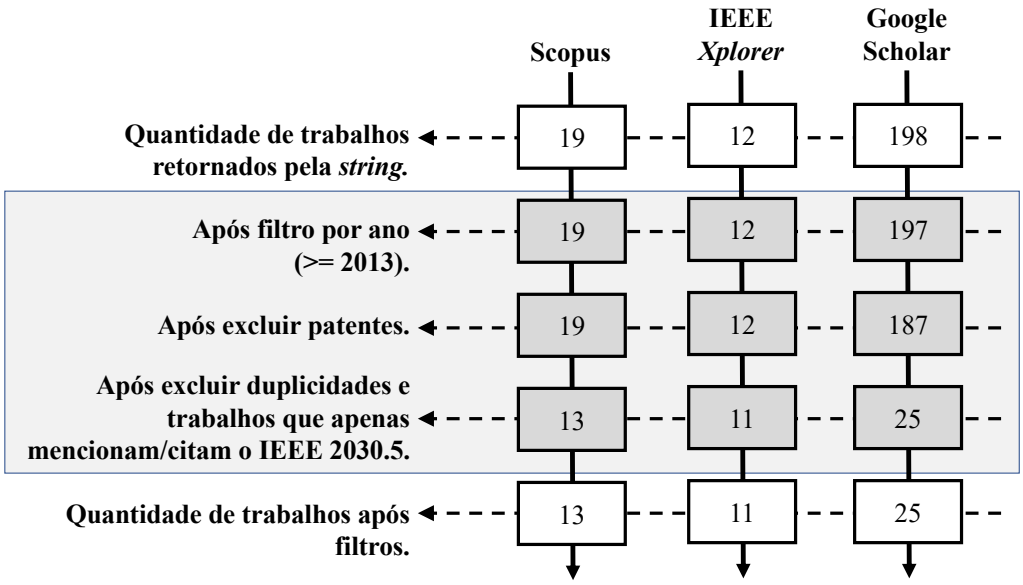


Figura 2.1: Aplicação de filtros nos trabalhos envolvendo o padrão IEEE 2030.5.

aplicados nos trabalhos.

O primeiro filtro aplicado é referente ao ano de publicação dos trabalhos. Tendo em vista que o IEEE 2030.5 foi formalmente adotado com essa nomenclatura pelo IEEE em 2013, um filtro foi aplicado a fim de selecionar trabalhos produzidos a partir desse ano.

O segundo filtro aplicado refere-se às patentes. Uma vez que tais trabalhos não tratam especificamente do padrão em si, nem fazem uma abordagem mais detalhada ao se referenciarem ao mesmo, esses foram excluídos.

Por fim, o terceiro filtro aplicado procura eliminar duplicidades no resultado retornado por cada base e também eliminar trabalhos e apresentações que apenas mencionam/citam o IEEE 2030.5. Nesse segundo caso, alguns trabalhos e apresentações que possuem temas correlatos, ao mencionar/citar o IEEE 2030.5, sequer explicitam ou definem sua estrutura, arquitetura ou funcionamento. Um exemplo são trabalhos e apresentações sobre *Smart Grids* que, ao tratar do tópico de comunicação, mencionam/citam o IEEE 2030.5 juntamente com os demais padrões/protocolos utilizados para tal fim, sem sequer apresentar ou trazer minimamente qualquer informação sobre a estrutura e funcionamento do mesmo.

A variação do quantitativo de trabalhos das bases, após a aplicação de cada um dos filtros, é apresentada na Figura 2.1. Devido ao grande número de duplicidades e trabalhos/apresentações que apenas mencionam/citam o IEEE 2030.5, o quantitativo de trabalhos retornado pela base do *Google Scholar* sofreu uma redução considerável em comparação às demais bases, as quais sofreram pequenas variações.

Tabela 2.2: Artigos selecionados

Título	Ano	Tipo de Trabalho
IEEE Adoption of Smart Energy Profile 2.0 Application Protocol Standard (*Draft*) [29]	2013	*Draft*
IEEE Adoption of Smart Energy Profile 2.0 Application Protocol Standard [28]	2013	Padrão
Model-driven development of a standard-compliant Customer Energy Manager [52]	2015	Secundário
Cyber security requirements and recommendations for CSI RD&D solicitation# 4 distributed energy resource communications [25]	2015	Segurança
Standardization of Smart Grid Customer Interfaces [7]	2015	Secundário
Advanced inverter functions and communication protocols for distribution management [48]	2016	Secundário
Exploring emerging cybersecurity risks from network-connected DER devices [61]	2017	Secundário
Cyber security primer for DER vendors, aggregators and grid operators [40]	2017	Segurança
IEEE Draft Standard for Smart Energy Profile Application Protocol [30]	2017	Draft
Implementation of a smart grid communication system compliant with IEEE 2030.5 [23]	2018	Implementação
Upper-middleware development of smart energy profile 2.0 for demand-side communications in smart grid [43]	2018	Secundário
General Requirements for Designing and Implementing a Cryptography Module for Distributed Energy Resource (DER) Systems [4]	2018	Segurança
IEEE Approved Draft Standard for Smart Energy Profile Application Protocol [31]	2018	Draft
IEEE Standard for Smart Energy Profile Application Protocol [33]	2018	Padrão
Application Prospect of Edge Computing in Power Demand Response Business [41]	2018	Secundário
Recommendations for trust and encryption in DER interoperability standards [49]	2019	Segurança
Cybersecurity Risk Assessment for California's Smart Inverter Functions [65]	2019	Segurança
Communications, cybersecurity, and the internet of things for microgrids [53]	2019	Secundário
Simulation and analysis of OpenADR agents using VOLTTRON platform [62]	2019	Secundário
IPv6-Based Smart Grid Communication over 6LoWPAN [67]	2019	Secundário
DER-TEE: Secure Distributed Energy Resource Operations Through Trusted Execution Environments [60]	2019	Secundário
Evolution of Distributed Energy Resource Grid Interconnection Standards for Integrating Emerging Storage Technologies [35]	2019	Secundário
Cyber Attack and Defense for Smart Inverters in a Distribution System [66]	2019	Segurança
Recommended functionalities for improving cybersecurity of distributed energy resources [16]	2019	Secundário
PV Cybersecurity Final Report [38]	2019	Segurança
Communication protocols for the IoT-based smart grid [24]	2019	Secundário
Considerations on Communication Infrastructures for Cooperative Operation of Smart Inverters [17]	2019	Secundário
Transactive Demand Response Operation at the Grid Edge using the IEEE 2030.5 Standard [21]	2020	Implementação
Cyber-Physical Security and Resiliency Analysis Testbed for Critical Microgrids with IEEE 2030.5 [59]	2020	Segurança
Assessing DER network cybersecurity defences in a power-communication co-simulation environment [37]	2020	Segurança
CReST-VCT System Integration Framework [51]	2020	Secundário
Distributed energy resource aggregation using customer-owned equipment: A review of literature and standards [50]	2020	Secundário

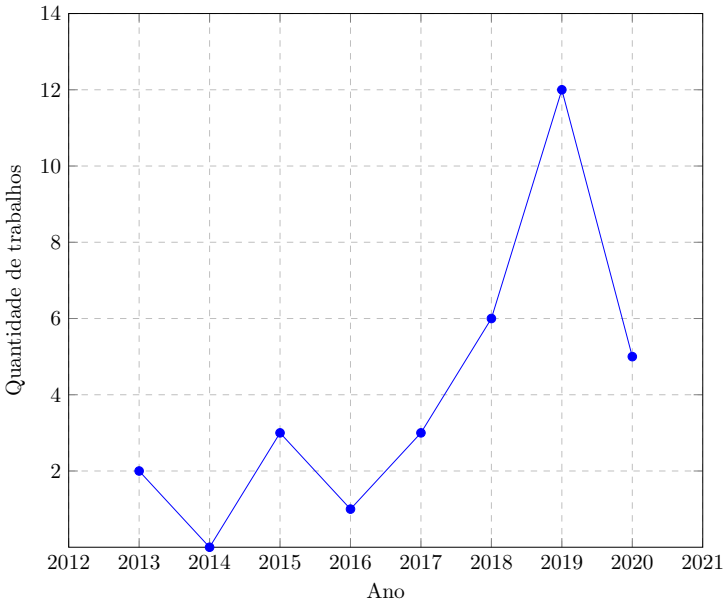


Figura 2.2: Evolução da quantidade de trabalhos envolvendo o padrão IEEE 2030.5.

Com o resultado obtido após a aplicação dos filtros, uma comparação entre os trabalhos resultantes das diferentes bases foi feita a fim de eliminar duplicidades entre as mesmas. A consolidação final dos trabalhos resultantes das três bases é apresentada na Tabela 2.2. Ela é ordenada de forma crescente de acordo com o ano de publicação de cada trabalho. Nela, também é apresentada uma classificação com relação ao tipo de abordagem feita por cada trabalho em relação ao IEEE 2030.5. As classificações utilizadas são:

- *Draft* – versão prévia não ratificada do padrão;
- Padrão – versão final ratificada do padrão;
- Secundário – o trabalho possui um tema correlato e trata, definindo minimamente estrutura ou funcionamento, o IEEE 2030.5 de forma secundária;
- Implementação – o trabalho faz uma implementação do IEEE 2030.5 a título de avaliação de desempenho ou conforme necessidade do trabalho;
- Segurança – o trabalho aborda aspectos de segurança não só do IEEE 2030.5 como também de outros padrões/protocolos;

De posse das informações presentes na Tabela 2.2, é possível verificar que desde 2013 houve um aumento na quantidade de trabalhos que abordam o IEEE 2030.5. A Figura 2.2 apresenta essa evolução no quantitativo de trabalhos ao longo dos anos até o presente momento da pesquisa[1]. Possivelmente, o fato do *California's Rule 21* ter adotado, em 2016,

[1]Dados de 04-11-2020

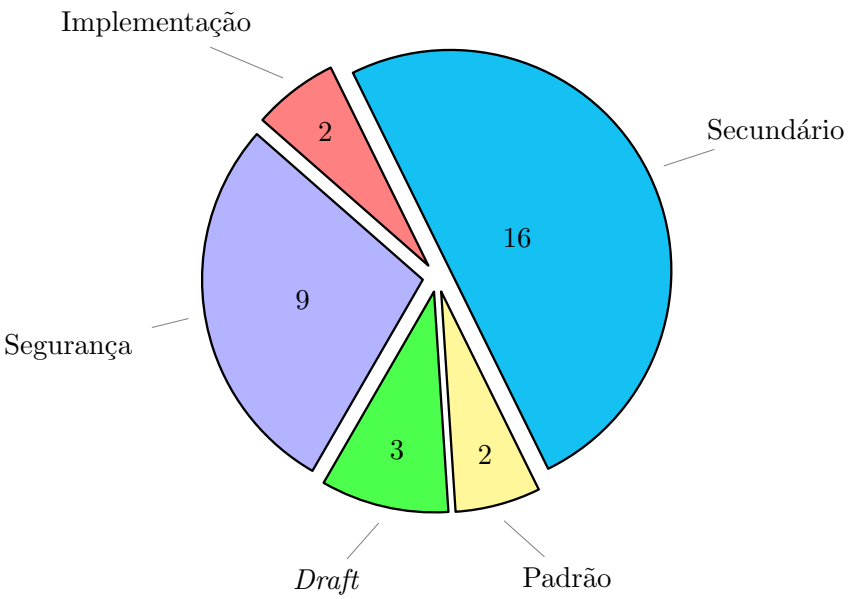


Figura 2.3: Tipo de trabalho abordando o IEEE 2030.5

o padrão, bem como a crescente necessidade de controle sobre os dispositivos inteligentes nas *smart grids* e nas HANs, ajudou no aumento da produção de trabalhos científicos.

No entanto, conforme demonstrado na Figura 2.3, nota-se que a maioria dos trabalhos abordam o padrão de forma secundária. Não foram encontrados trabalhos na literatura científica que abordem o IEEE 2030.5 na íntegra, fazendo uma completa revisão quanto à sua estrutura, desenvolvimento, arquitetura de implementação, segurança na comunicação e confiabilidade dos entes envolvidos.

Os trabalhos [29, 30, 31] são classificados como "*drafts*", ou seja, são versões prévias não ratificadas do padrão. São minutas de proposta da versão final. Já os trabalhos [28, 33] são as versões finais ratificadas do padrão publicadas nos anos de 2013 e 2018, respectivamente.

Os trabalhos classificados como "Segurança", em sua maioria, realizam estudos, análises e avaliações de segurança nos protocolos/padrões utilizados na comunicação das DERs. Eles descrevem possíveis riscos, vulnerabilidades e também apresentam alternativas para aprimorar a segurança na comunicação de modo geral. O trabalho [25] descreve o ambiente DER elencando diversas ameaças de forma geral, trazendo recomendações de segurança da mesma forma, com foco em um módulo específico de conversão de outros protocolos para o protocolo Modbus. O trabalho [40] traz um resumo simplificado dos recursos de segurança utilizado por cada protocolo específico do ambiente DER em comparação com os requisitos necessários para tal ambiente. O trabalho [4] busca prover a base para futuros projetos e implementações de um módulo criptográfico dedicado a promover segurança

na comunicação de sistemas DERs. No entanto, possui uma abordagem bem superficial focando apenas na localização de implementação do módulo levando-se em conta apenas alguns requisitos do sistema. O trabalho [49] aborda recomendações de forma geral para prover segurança na comunicação das DERs e também enumera algumas medidas alternativas que podem ser usadas, tal como o uso de módulos de segurança e *blockchain*, sem aprofundar em detalhes. O trabalho [65] faz um resumo das funcionalidades dos inversores inteligentes no programa *California's Rule 21* e modela as ameaças de segurança usando o modelo STRIDE (*Spoofing, Tampering, Repudiation, Information Disclosure, Denial of Service, and Elevation of Privilege*). O trabalho [66] desenvolve um *Intrusion Detection System* (IDS) baseado em assinaturas para detecção de atividades maliciosas. O trabalho [38] apresenta um panorama geral comparativo da segurança dos protocolos no ambiente DER com recomendações de segurança. Por fim, o trabalho [59] propõe um ambiente de teste e simulação para estudo e análise da resiliência da comunicação enquanto [37] faz uma análise do impacto na qualidade do serviço (QoS) proveniente da segmentação da rede e do uso de criptografia na comunicação das DERs baseado em um ambiente virtualizado de desenvolvimento próprio chamado SCEPTRE.

O trabalho [23] implementa o padrão visando promover uma avaliação de desempenho da comunicação, enquanto [21] implementa o padrão para dar suporte a uma proposta de esquema de resposta à demanda para uma HAN. Ambos são trabalhos classificados como "Implementação".

Os demais trabalhos, que são classificados como "Secundários", não possuem como foco principal o IEEE 2030.5. Eles apenas o utilizam secundariamente para dar suporte ao tema principal abordado. Por exemplo, o trabalho [52] descreve um modelo de gerenciamento de energia do cliente com suporte ao IEEE 2030.5. O trabalho [43] realiza a implementação de um *middleware* com suporte ao IEEE 2030.5 a fim de sobrepor restrições próprias dos dispositivos IoT, tais como restrições de *hardware*. Já o [67] realiza a implementação do IPv6 sobre o 6LoWPAN para conectar dispositivos em conformidade com o IEEE 2030.5.

Trabalhos tais como [17, 50] chegam a abordar o tema de revisão de padrões de comunicação. O primeiro elenca os pontos mais relevantes dos principais padrões de comunicação que são aplicados em *smart inverters*, realizando um breve comparativo e uma breve discussão sobre as vantagens e desvantagens de cada um. Já o segundo tem como foco uma revisão da literatura das soluções que as concessionárias usam para controlar os problemas causados pela adoção em larga escala das DERs. Ele também dá

enfoque na revisão dos padrões de comunicação usados para gerenciar tais soluções. O trabalho faz uma descrição sucinta de cada padrão elencando alguns detalhes em específico de cada um. No entanto, a abordagem de revisão feita por ambos os trabalhos é genérica e não possui o IEEE 2030.5 como foco em específico. Além disso, o padrão é apenas um dentre os demais padrões analisados. A revisão não aborda todos os pontos que compõem a estrutura, arquitetura e segurança do IEEE 2030.5.

Sendo assim, torna-se perceptível a ausência de um trabalho completo, principalmente no meio acadêmico, que revise todos os pontos arquiteturais, estruturais e de segurança do IEEE 2030.5-2018. Tal ausência amplia ainda mais a importância deste trabalho, que busca preencher tal lacuna além de trazer à tona, para discussão, as possíveis questões em aberto em relação à segurança.

Capítulo 3

Aplicação e Topologia Lógica do IEEE 2030.5-2018

O IEEE 2030.5-2018, além de possibilitar a interoperabilidade de diversos dispositivos inteligentes, também permite expandir a gerência do ambiente de energia dos usuários finais de tal forma que as concessionárias possam exercer tal gerência. Isso inclui gerenciamento de geração de energia distribuída, suporte a veículos elétricos e controle de carga e resposta à demanda. Para isso, os usuários finais usam nas residências dispositivos inteligentes que possuem uma interface de comunicação que permite o gerenciamento remoto.

Embora a aplicação ao setor de energia seja frequentemente citada, o padrão também pode ser empregado em outros nichos (*e.g.*, água, vapor e gás natural). O monitoramento e o controle centralizado dos diversos dispositivos inteligentes, dos mais variados nichos presentes em uma HAN, pode ser feito utilizando um *Home Energy Management System* (HEMS). O HEMS possibilita uma visão geral de todos os dispositivos inteligentes do ambiente residencial. Com isso, é possível obter um maior controle e otimizar o consumo tanto da energia elétrica quanto das demais *commodities* dos outros nichos.

Basicamente, um HEMS pode ser tanto uma solução de *software* quanto uma solução integrada de *hardware* e *software*. Sendo assim, pode ser um equipamento dedicado exclusivamente para esse fim ou pode ser implementado em outros dispositivos (*e.g.*, em medidores elétricos inteligentes). Ele pode agir ativamente enviando comandos aos dispositivos inteligentes para que os mesmos alterem seus parâmetros conforme a necessidade, ou pode agir passivamente apenas colhendo informações dos dispositivos inteligentes para serem exibidas em um monitor de informações. Por exemplo, agindo passivamente, o HEMS pode monitorar um aparelho de ar-condicionado, lâmpadas e máquina de lavar

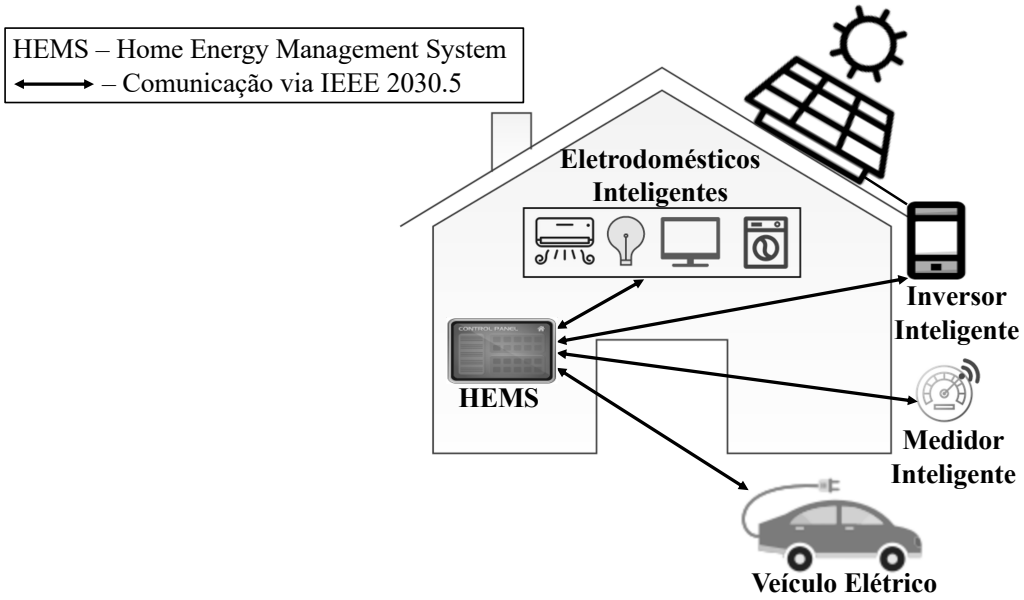


Figura 3.1: Topologia de comunicação entre HEMS e dispositivos usando IEEE 2030.5.

apresentando em um monitor de informações (*e.g.,* aplicativo ou tela) a quantidade de energia consumida por cada equipamento, horário de maior consumo, entre outras informações relevantes. Agindo ativamente, o HEMS pode desabilitar a recarga de um veículo elétrico caso não seja detectada, pelo monitoramento do inversor inteligente, a produção de energia solar.

O padrão IEEE 2030.5-2018 pode ser usado tanto em uma HAN não integrada aos servidores da concessionária como em uma integrada. A Figura 3.1 apresenta uma HAN controlada por um HEMS. Nela, é possível notar os mais diversos dispositivos inteligentes inclusive uma DER representada pelo painel solar juntamente com o inversor inteligente (o veículo elétrico quando utilizado como uma fonte produtora de energia também pode ser considerado uma DER). O HEMS se comunica com os dispositivos inteligentes via IEEE 2030.5-2018. Nessa topologia, em específico, todo o controle e monitoramento dos dispositivos inteligentes ficam restritos ao ambiente residencial, não sendo possível o controle pela concessionária, uma vez que a HAN não está integrada aos servidores da concessionária.

No entanto, os exemplos de aplicações mais recentes concentram-se na comunicação que extrapola o ambiente residencial, mais especificamente a comunicação envolvendo servidores da concessionária de energia elétrica. Nesse caso, o HEMS pode funcionar como um *gateway* da HAN de forma a promover a integração da mesma com os servidores da concessionária conforme apresentado na Figura 3.2. Nela, o HEMS é apresentado em um *hardware* dedicado. No entanto, ele poderia ser implementado em um medidor elétrico

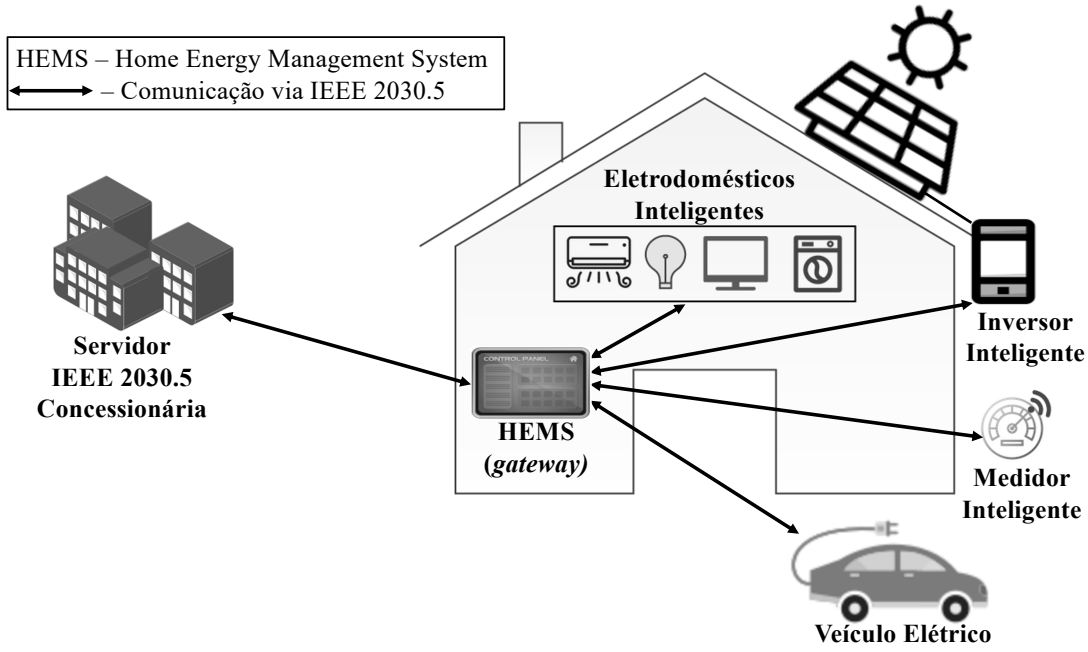


Figura 3.2: Topologia de comunicação entre HEMS e dispositivos usando IEEE 2030.5.

inteligente e este, por sua vez, também poderia agir como o *gateway* da HAN.

Tanto a comunicação do HEMS com os dispositivos inteligentes da HAN, quanto a comunicação do HEMS com os servidores da concessionária são realizadas utilizando o IEEE 2030.5-2018. Com isso, uma vez integrada aos servidores da concessionária, a HAN pode trocar informações e receber comandos da concessionária. Um exemplo prático é a possibilidade da concessionária informar o preço da energia elétrica em determinado momento para que o consumidor final tenha ciência do valor que está sendo gasto. Outro exemplo é a concessionária ter a capacidade de desativar a recarga de um veículo elétrico em caso de sobrecarga de consumo na região. Ela também pode direcionar o excedente de energia produzido pelos painéis solares de uma residência para a rede de distribuição, a fim de compensar parte da demanda em outra HAN/região. Em troca, o proprietário dos painéis solares poderia receber créditos em sua fatura mensal.

Essa interação entre concessionária e HAN propicia uma grande otimização no consumo de energia. Ela possibilita que as concessionárias antevejam regiões de grande demanda de consumo de energia, permitindo que as mesmas reajam de forma a suprir tal demanda evitando que falhas e sobrecargas possam comprometer a rede como um todo.

Nada impede também que os dispositivos inteligentes se integrem aos servidores da concessionária sem fazer o uso de um HEMS. Um exemplo é uma residência que tenha apenas painéis solares e um inversor inteligente. O inversor inteligente pode ser monitorado e controlado por meio da interface de comunicação fazendo uso de um aplicativo de

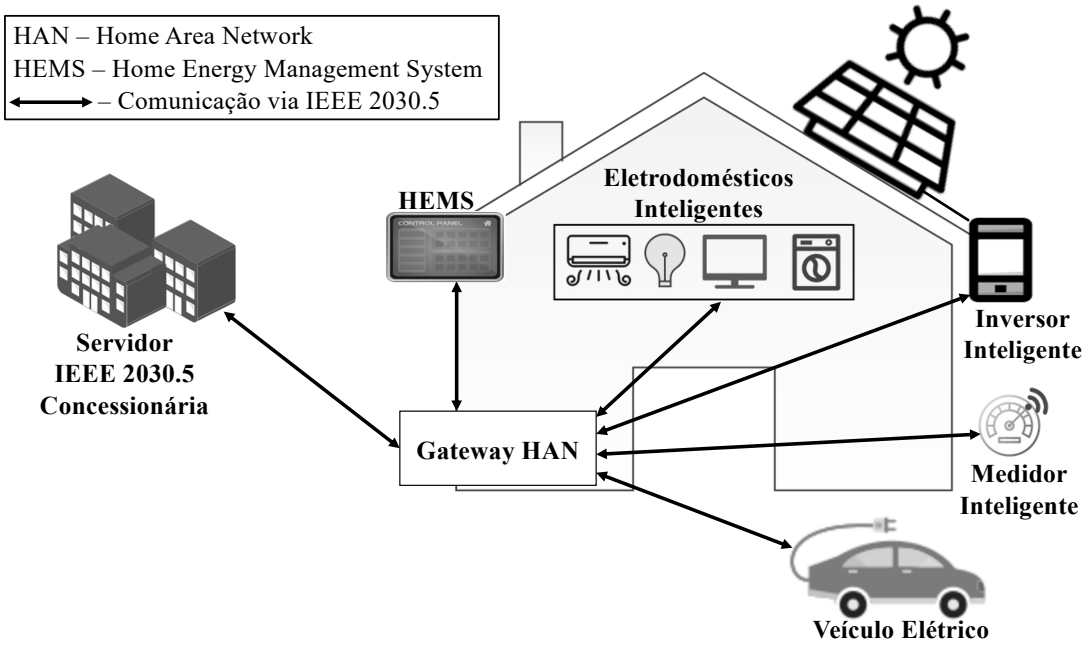


Figura 3.3: Comunicação entre concessionária e dispositivos usando IEEE 2030.5.

smartphone provido pela fabricante do inversor, por exemplo. Por meio da mesma interface de comunicação do inversor inteligente também é possível promover a comunicação direta dele com a concessionária para que ambos troquem informações e seja possível o envio de comandos da concessionária ao inversor.

Portanto, diversas são as possibilidades de topologia envolvendo a interação entre a concessionária e os dispositivos inteligentes presentes na HAN. A Figura 3.3 apresenta uma topologia mais genérica dessa interação envolvendo o IEEE 2030.5-2018 na comunicação.

Já em se tratando de DERs, especificamente no âmbito do *Californa's Rule 21*, o CSIP prevê dois tipos de topologias para a comunicação entre a concessionária e as DERs: comunicação direta e comunicação via agregador.

A comunicação direta é usada quando a concessionária precisa interagir diretamente com a DER para controlar as operações do sistema [15]. Para isso, pode-se fazer uso de um *Smart Inverter Control Unit* (SMCU)[1] ou *Generating Facility Energy Management System* (GFEMS)[1].

No primeiro caso, a concessionária exerce a gerência e o controle sobre uma única unidade DER utilizando o SMCU que é um componente que provê a comunicação fazendo uso do IEEE 2030.5-2018. O SMCU pode ser integrado à DER formando um conjunto único conforme mostrado na Figura 3.4-(A). Ele também pode ser externo. Sendo externo,

[1]SMCU e GFEMS são termos utilizados no *California's Rule 21*

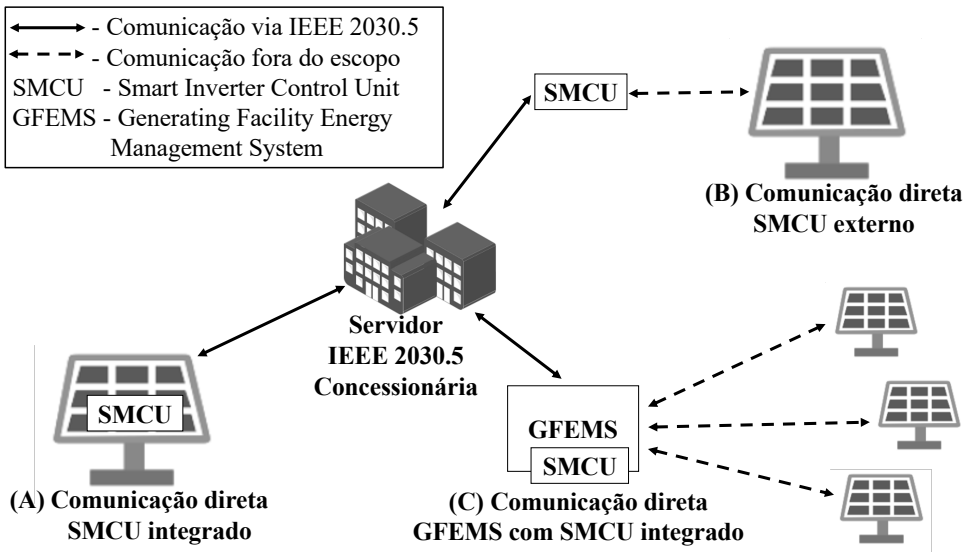


Figura 3.4: Comunicação entre concessionária e DERs usando SMCU e GFEMS. (adaptado de [15]).

a comunicação entre o SMCU e a DER pode fazer uso de outros padrões ou protocolos os quais estão fora do escopo do CSIP conforme mostrado na Figura 3.4-(B).

Já no segundo caso, a concessionária se comunica com uma ou mais DERs por meio de um GFEMS, que é basicamente um dispositivo controlador que possui um SMCU integrado e controla as DERs que estão conectadas a ele. A comunicação entre a concessionária e o GFEMS usa o IEEE 2030.5-2018. No entanto, a comunicação entre o GFEMS e as DERs utiliza padrões ou protocolos proprietários fora do escopo do CSIP, conforme mostrado na Figura 3.4-(C). Embora na última arquitetura possa haver mais de uma DER sendo controlada e gerenciada pelo GFEMS, para a concessionária, todas as DERs aparecem como uma única DER.

Deve-se notar que a noção de DER para o CSIP é um conceito lógico onde geralmente um ou mais inversores físicos são agrupados e operam como um sistema único tendo um ponto em comum de agregação dos mesmos seguido de um ponto único de acoplamento com a concessionária [15].

Na comunicação via agregador, a concessionária comunica-se com um sistema de controle e gerenciamento agregador em vez de diretamente com as DERs [15]. A comunicação entre o agregador e os diversos dispositivos conectados a ele pode fazer uso do IEEE 2030.5-2018 ou de qualquer outro protocolo/padrão de comunicação, incluindo protocolos proprietários, conforme mostrado na Figura 3.5. Já a comunicação entre a concessionária e o agregador usa IEEE 2030.5-2018. O agregador atua como um tradutor e retransmissor de informações. Ele converte as informações e comandos recebidos da concessionária

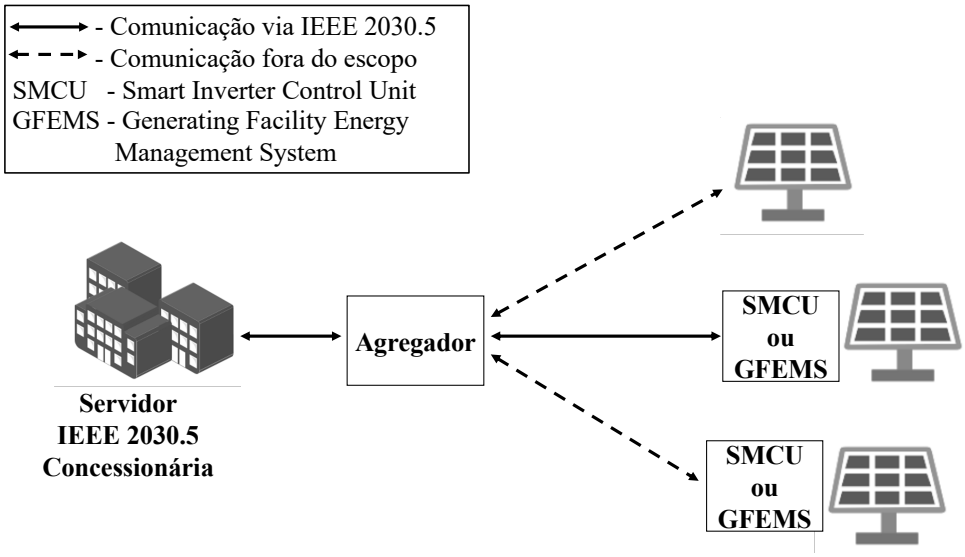


Figura 3.5: Comunicação entre concessionária e DERs usando agregador. (adaptado de [15]).

no protocolo/padrão usado pela DER de destino e, em seguida, retransmite as informações/comandos. Quando a DER envia informações para a concessionária, o agregador, ao receber a informação, faz a tradução inversa e, logo após, a retransmite para a concessionária. Ao contrário da arquitetura com GFEMS, cada DER conectada ao agregador é vista e reconhecida de forma individual pela concessionária.

Com as diferenças e especificidades de cada topologia, a escolha da mais adequada a se utilizar para conexão de uma DER depende das necessidades e dos requisitos de cada concessionária.

Capítulo 4

Arquitetura do IEEE 2030.5-2018

O IEEE 2030.5-2018 define uma camada de aplicação no topo da pilha de protocolos TCP/IP para permitir que as concessionárias gerenciem o ambiente de energia dos usuários finais, incluindo resposta à demanda e controle de carga, preço em determinado horário do dia, gerenciamento de geração de energia distribuída e suporte a veículos elétricos [33]. Além disso, o padrão fornece suporte total ao IEEE 1547 [32], que é um padrão para interconectar recursos distribuídos com sistemas de energia elétrica. O padrão também define os mecanismos a serem utilizados para a troca de mensagens da aplicação e os recursos de segurança utilizados para proteger tais mensagens. A Figura 4.1 apresenta uma pilha TCP/IP com a referida camada de aplicação no topo.

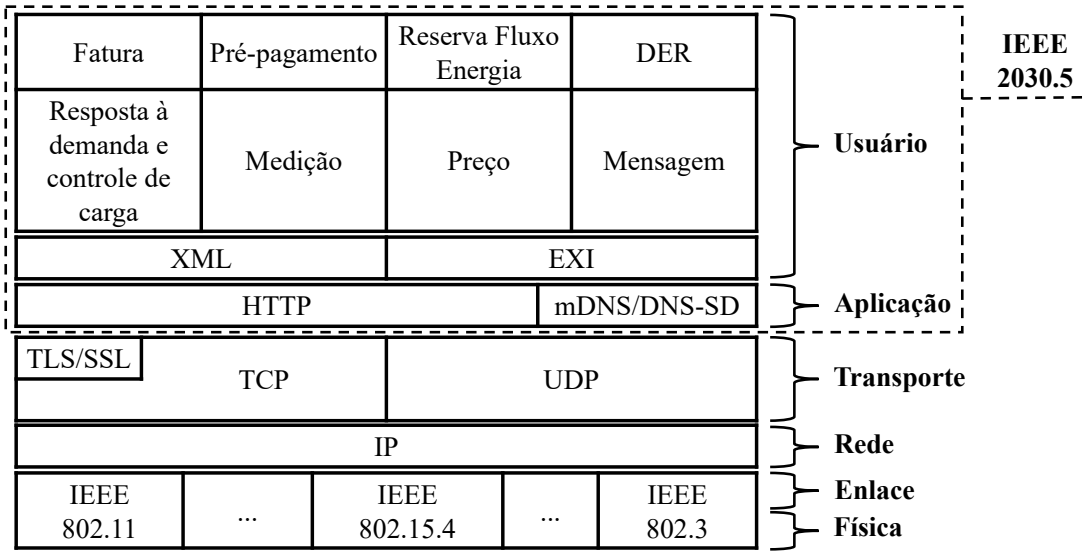


Figura 4.1: Arquitetura da pilha IEEE 2030.5-2018 [43]

Alguns dos elementos que compõem o IEEE 2030.5-2018 são originados de outros padrões, como IEC 61968 e IEC 61850. O IEC 61968 é uma série de padrões que se desti-

nam a facilitar a integração entre sistemas de distribuição de eletricidade [42]. O padrão IEC 61850 define protocolos, topologias e requisitos de comunicação para automação e proteção em subestações de transmissão e distribuição elétrica [42]. O IEC 61850 também é um protocolo que pode ser utilizado no gerenciamento de uma DER. No entanto, enquanto o IEEE 2030.5 é mais relacionado com o lado do usuário final, promovendo o controle e gerenciamento dos dispositivos inteligentes — incluindo a DER —, o IEC 61850 é mais relacionado com as necessidades e o ambiente da concessionária, principalmente o ambiente de subestações de energia elétrica. Outra diferença está no fato de que o IEC 61850 não inclui nativamente autenticação nem segurança na comunicação, enquanto que o IEEE 2030.5 inclui ambas.

O padrão define as propriedades e atributos dos dispositivos que podem ser manipulados. Tais propriedades e atributos são conhecidos como "recursos". Leitura de medidores e eventos de resposta à demanda são exemplos de recursos. Os recursos podem ser agrupados em conjuntos lógicos denominados "conjuntos de funções". Cada conjunto de funções representa um conjunto mínimo de comportamentos e funcionalidades de um dispositivo [68]. Os dispositivos inteligentes podem implementar um ou mais conjunto de funções. Por exemplo, preço da tarifa e medição são conjuntos de funções que podem ser utilizados em um medidor elétrico inteligente.

Os recursos são disponibilizados por meio de uma interface *Representational State Transfer* (RESTful) [22] — que é um modelo cliente-servidor — sobre o *Hypertext Transfer Protocol* (HTTP) e são manipulados através dos métodos *GET*, *HEAD*, *PUT*, *POST* e *DELETE*. Os clientes acessam os recursos disponibilizados pelos servidores via *Uniform Resource Identifier* (URI) e realizam operações como leitura, escrita, criação e deleção por meio dos métodos.

Por fazer uso do HTTP, toda a comunicação no IEEE 2030.5-2018 é realizada implicitamente por meio de *Transmission Control Protocol* (TCP), que fornece troca de dados confiável entre os envolvidos. Além disso, o HTTP também permite o transporte de metadados tais como "*content coding*" e "*media types*" que fornecem à camada de aplicação informações úteis sobre a interpretação dos dados [34].

Em se tratando dos cabeçalhos do HTTP, devido às restrições presentes nos dispositivos — principalmente restrições de *hardware* — nem todos os cabeçalhos devem ser utilizados. O padrão classifica os cabeçalhos HTTP, quanto à utilização, em: obrigatórios, opcionais e desencorajados. A Tabela 4.1 apresenta os cabeçalhos classificados como obrigatórios pelo IEEE 2030.5-2018.

Tabela 4.1: Classificação dos cabeçalhos HTTP

Cabeçalho HTTP	Tipo de mensagem	Classificação
Accept	Requisição	Obrigatório
Allow	Resposta	Obrigatório
Content-Type	Requisição/Resposta	Obrigatório
Date	Requisição/Resposta	Obrigatório
Host	Requisição	Obrigatório
Location	Resposta	Obrigatório

Para dar suporte à interface RESTful, o padrão IEEE 2030.5-2018 apresenta dois materiais suplementares: o *IEEE 2030.5 XML Schema Definition* (disponível na forma do arquivo `sep.xsd`) e o *IEEE 2030.5 WADL* (disponível na forma do arquivo `sep_wadl.xml`). O `sep.xsd` especifica um formato *eXtensible Mark-up Language* (XML) para que os dispositivos IEEE 2030.5 possam representar seus recursos, atributos, elementos e suas descrições textuais. Já o `sep_wadl.xml` contém as estruturas de URI recomendadas para o acesso a tais propriedades e os métodos HTTP associados.

4.1 Web Application Description Language

O *Web Application Description Language* (WADL) é um vocabulário XML usado para descrever serviços RESTful. Um arquivo WADL descreve uma requisição HTTP legítima, ou seja, quais URIs estão diponíveis para acesso, quais dados são esperados por essas URIs e quais dados elas podem retornar. Um cliente genérico qualquer pode ler um arquivo WADL e ser imediatamente capaz de utilizar os serviços RESTful [34].

A interface RESTful do IEEE 2030.5-2018 é definida pelo `sep_wadl.xml`. O padrão adota tanto o XML quanto o *Efficient XML Interchange* (EXI) — que é uma representação compacta do XML — como possíveis codificações para o *payload* das operações RESTful. No entanto, é requerido que o lado do servidor implemente ambas as codificações, enquanto o lado do cliente pode implementar qualquer uma. Uma das vantagens do formato XML é que ele pode ser lido por humanos e máquinas com relativa facilidade.

Dependendo do formato de codificação utilizado, as respostas usam de dois tipos possíveis de "*Content-Type*":

- `application/sep+xml`
- `application/sep-exi`

Quando usado o `application/sep-exi`, um nível ("*level*") de extensibilidade deve ser especificado no cabeçalho "`Accept`" para negociação da versão do *schema*. Esse *level* define a base do *schema* e sua capacidade para extensão arbitrária. O *level* pode ser `-S1` ou `+S1`. Em ambos os casos, `S1` indica a versão do *schema* base: IEEE 2030.5-2018. Entretanto, `-S1` indica que *tags* que não são definidas no *schema* base não são aceitas, enquanto `+S1` indica que *tags* arbitrárias são aceitas. Todas as representações dos recursos são validadas de acordo com o *schema* padronizado de *namespace* XML (`urn:ieee:std:2030.5:ns`).

Por exemplo, o cabeçalho "`Accept: application/sep-exi; level=-S1`" indica que o cliente deseja receber o conteúdo codificado usando EXI onde o *schema* base do cliente é o IEEE 2030.5-2018 e ele não aceita *tags* arbitrárias não definidas no *schema*.

As operações HTTP para cada recurso são classificadas no `sep_wadl.xml` de acordo com se são ou não recomendadas para a implementação. As possíveis classificações são:

- Obrigatórias (`Mandatory - ‘‘M’’`): devem ser implementadas de acordo com o especificado.
- Opcionais (`Optional - ‘‘O’’`): podem ser implementadas e, caso sejam, devem ser conforme especificado.
- Desencorajadas (`Discouraged - ‘‘D’’`): não devem ser implementadas, mas, caso sejam, devem ser conforme especificado.
- Erro (`Error - ‘‘E’’`): devem ser implementadas retornando um código de status específico.

A Figura 4.2 apresenta o trecho do `sep_wadl.xml` referente ao recurso *EndDevice*. Tal recurso contém informações próprias e individuais de cada dispositivo na rede. De forma resumida:

- Amostra de URI: /edev/{id1}
- Representação da requisição: EndDevice
- Representação da resposta: EndDevice
- Métodos:
  - GET/HEAD: Obrigatório (M);

- PUT: Opcional (O);
- POST: Error (E);
- DELETE: Opcional (O);

Nota-se tanto a estrutura da URI para acesso ao recurso quanto a referida recomendação dos métodos HTTP, sendo que o método POST deve retornar um dos códigos especificados (*e.g.,* 400 Bad Request ou 405 Method Not Allowed).

```
1  <resource id="EndDevice" wx:samplePath="/edev/{id1}">
2      <doc title="EndDevice">End device instance.</doc>
3      <method id="GETEndDevice" name="GET" wx:mode="M">
4          <response>
5              <representation mediaType="application/sep+xml" element="sep:EndDevice"/>
6              <representation mediaType="application/sep-exi" element="sep:EndDevice"/>
7          </response>
8      </method>
9      <method id="HEADEndDevice" name="HEAD" wx:mode="M"/>
10     <method id="PUTEndDevice" name="PUT" wx:mode="O">
11         <request>
12             <representation mediaType="application/sep+xml" element="sep:EndDevice"/>
13             <representation mediaType="application/sep-exi" element="sep:EndDevice"/>
14         </request>
15     </method>
16     <method id="POSTEndDevice" name="POST" wx:mode="E">
17         <request>
18             <representation mediaType="application/sep+xml" element="sep:EndDevice"/>
19             <representation mediaType="application/sep-exi" element="sep:EndDevice"/>
20         </request>
21         <response status="400"/>
22         <response status="405"/>
23     </method>
24     <method id="DELETEEndDevice" name="DELETE" wx:mode="O">
25         <response>
26             <representation mediaType="application/xml" element="sep:EndDevice"/>
27         </response>
28     </method>
29     <wx:sampleParam name="id1" style="template" type="idType"/>
30 </resource>
```

Figura 4.2: Recurso *EndDevice* - `sep_wadl.xml`

4.2 XML Schema Definition

Um *XML Schema Definition* (XSD) é uma linguagem para expressar restrições sobre documentos XML. O objetivo de um XSD é definir os blocos de construções legais de um documento XML [34].

Tecnicamente, um *schema* é uma coleção abstrata de metadados consistindo em um conjunto de componentes, principalmente declarações de elementos e atributos e definição de tipos simples e complexos. Além disso, ele permite que o conteúdo do documento XML seja tratado como objeto dentro do ambiente de programação [34].

No IEEE 2030.5-2018, o `sep.xsd` é o documento base de validação utilizado para verificar se um XML está em conformidade ou não. Nele, estão contidas as definições dos recursos, atributos, elementos e as descrições textuais. A Figura 4.3 apresenta o trecho do arquivo `sep.xsd` referente ao recurso "*EndDevice*".

```
1  <xs:complexType name="EndDevice">
2      <xs:annotation>
3          <xs:documentation>Asset container that performs one or more end device
           functions. Contains information about individual devices in the
           network.</xs:documentation>
4      </xs:annotation>
5      <xs:complexContent>
6          <xs:extension base="AbstractDevice">
7              <xs:sequence>
8                  <xs:element name="changedTime"
9                              minOccurs="1"
10                             maxOccurs="1"
11                             type="TimeType">
12                     <xs:annotation>
13                         <xs:documentation>The time at which this resource was last
                           modified or created.</xs:documentation>
14                     </xs:annotation>
15                 </xs:element>
16                 <xs:element name="enabled"
17                             minOccurs="0"
18                             maxOccurs="1"
19                             type="xs:boolean">
20                     <xs:annotation>
21                         <xs:documentation>This attribute indicates whether or not an
                           EndDevice is enabled, or registered, on the server. If a
                           server sets this attribute to false, the device is no longer
                           registered. It should be noted that servers can delete
                           EndDevice instances, but using this attribute for some time
                           is more convenient for clients.</xs:documentation>
22                     </xs:annotation>
23                 </xs:element>
24                 <xs:element name="FlowReservationRequestListLink"
25                             type="FlowReservationRequestListLink"
26                             minOccurs="0"
27                             maxOccurs="1"/>
28                 <xs:element name="FlowReservationResponseListLink"
29                             type="FlowReservationResponseListLink"
30                             minOccurs="0"
31                             maxOccurs="1"/>
32                 <xs:element name="FunctionSetAssignmentsListLink"
33                             type="FunctionSetAssignmentsListLink"
34                             minOccurs="0"
35                             maxOccurs="1"/>
36                 <xs:element name="RegistrationLink"
37                             type="RegistrationLink"
38                             minOccurs="0"
39                             maxOccurs="1"/>
40                 <xs:element name="SubscriptionListLink"
41                             type="SubscriptionListLink"
42                             minOccurs="0"
43                             maxOccurs="1"/>
44             </xs:sequence>
45         </xs:extension>
46     </xs:complexContent>
47 </xs:complexType>
```

Figura 4.3: Recurso *EndDevice* - `sep.xsd`

Nela, é possível notar a definição do tipo complexo (`<xs:complexType>`) e dos elementos (`<xs:element>`) que o compõem. A Tabela 4.2 apresenta, de forma consolidada, os elementos e os seus respectivos tipos.

Tabela 4.2: Tipos dos elementos do recurso *EndDevice* - `sep.xsd`

Elemento	Tipo
changedTime	TimeType
enabled	xs:boolean
FlowReservationRequestListLink	FlowReservationRequestListLink
FlowReservationResponseListLink	FlowReservationResponseListLink
FunctionSetAssignmentsListLink	FunctionSetAssignmentsListLink
RegistrationLink	RegistrationLink
SubscriptionListLink	SubscriptionListLink

4.3 Uniform Resource Identifier

Um *Uniform Resource Identifier* (URI) é uma compacta sequência de caracteres que permite identificar unicamente um recurso [5]. A URI identifica um recurso pela sua localização, pelo nome ou por ambos.

A identificação feita pela localização é denominada *Uniform Resource Locator* (URL). Ela especifica onde o recurso pode ser encontrado e o mecanismo para acesso ao mesmo. Já a identificação feita pelo nome é denominada *Uniform Resource Name* (URN). Ela identifica unicamente um recurso pelo nome em um determinado *namespace* independente da localização. Sendo assim, URL e URN são tipos de URI. Um exemplo prático para URL são os endereços *web*, onde é possível identificar o protocolo de acesso — geralmente HTTP — e a localização do recurso a ser acessado. Já para URN, um exemplo seria o *International Standard Book Number* (ISBN) que é utilizado para identificar livros.

O IEEE 2030.5-2018 define diretrizes gerais para a alocação de URIs aos recursos a fim de equilibrar a eficiência — *i.e.*, através de endereços curtos — e inteligibilidade. Os elementos que constituem a parte do caminho URI devem:

- ter no máximo 4 caracteres;
- ser escritos em letras minúsculas;
- ser constituídos basicamente de consoantes, a menos que o uso de vogal acrescente maior clareza; e
- a URI como um todo não deve ter mais de 255 bytes (na prática elas têm menos de 80 bytes).

Os recursos do padrão possuem *links* para seus recursos subordinados a fim de promover flexibilidade e extensibilidade futura. Portanto, as URIs para acesso aos recursos

nos dispositivos não são fixas. Sendo assim, é possível recuperar a URI apropriada de um recurso por meio da descoberta de recursos ou por meio de *links* em outros recursos já acessados. Como não há garantia de que todos os dispositivos tenham acesso a um *Domain Name System* (DNS), as URIs devem usar apenas nomes de domínio *Extended Multicast* DNS (xmDNS) [45].

4.4 Descoberta de Recursos

O IEEE 2030-2018 faz uso do DNS *Service Discovery* (DNS-SD) [13] para descobrir a localização dos recursos e dos conjuntos de funções na rede. O DNS-SD usa o conceito de *Multicast* DNS (mDNS) [14] o qual realiza consultas DNS na rede local na ausência de um servidor DNS dedicado.

A definição do DNS-SD especifica que um par de registros DNS SRV e TXT são utilizados para descrever instâncias de um serviço em um dado domínio. Serviço é definido como uma instância de uma aplicação unicamente identificada por *host*, porta e protocolo. Ambos os registros possuem a mesma forma "`<Instance>.<Service>.<Domain>`" que é denominada *Service Instance Name*. O registro DNS SRV contém o *hostname* e a porta do serviço. Já o registro DNS TXT possui informações adicionais tal como caminho relativo. No DNS-SD, a descoberta de instâncias de um serviço é feita através de

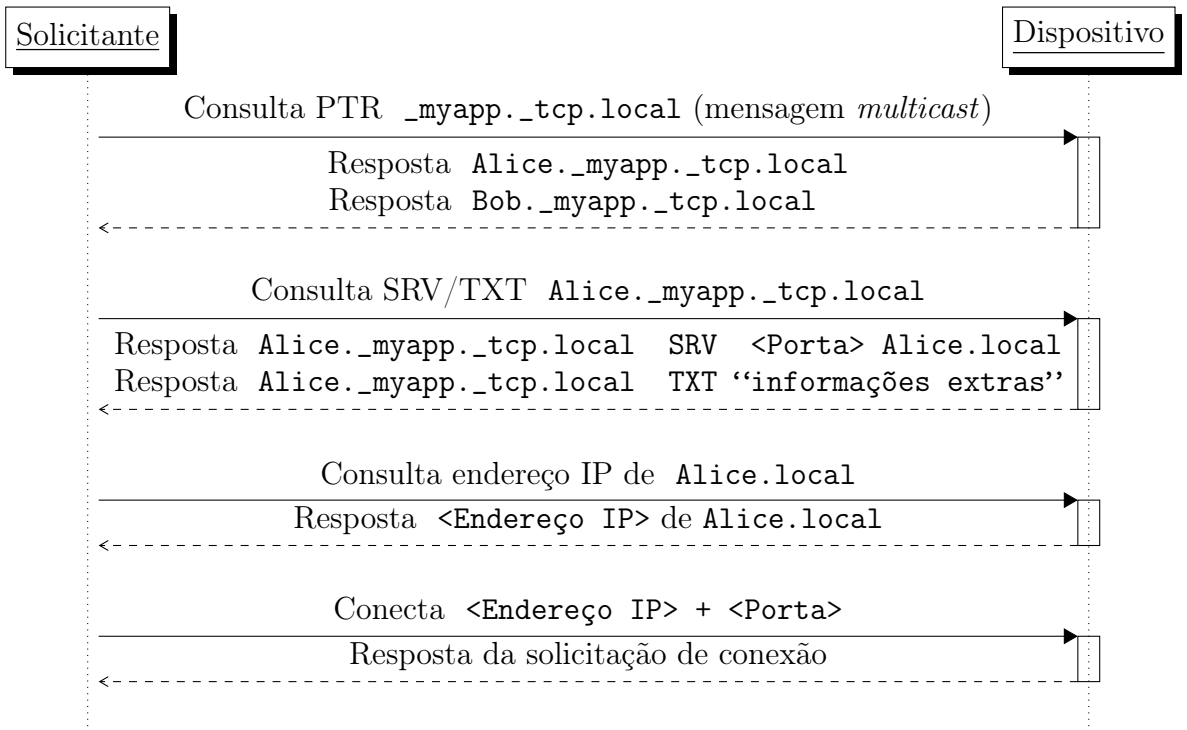


Figura 4.4: Fluxo DNS-SD para localizar instância de um serviço

consultas a registros DNS PTR. A consulta segue a forma "`<Service>.<Domain>`". O retorno a essa consulta pode ser um conjunto de zero ou mais pares de registros DNS SRV/TXT. A Figura 4.4 apresenta um exemplo. Inicialmente é feita uma consulta *multicast* DNS PTR para encontrar as instâncias na rede do serviço denominado "*myapp*". A *string* utilizada na consulta é "`_myapp._tcp.local`", onde "`myapp`" é o serviço, "`tcp`" é o protocolo de transporte e "`local`" é o domínio. A resposta obtida são duas instâncias: `Alice._myapp._tcp.local` e `Bob._myapp._tcp.local`. Consultando a instância `Alice._myapp._tcp.local`, obtêm-se as informações dos registros DNS SRV/TXT. Com a informação do *hostname* — que no caso é `Alice.local` — é possível realizar uma consulta a fim de obter o endereço IP. De posse do endereço IP e da porta é possível iniciar uma conexão ao serviço ‘‘`myapp`’’ disponibilizado por ‘‘`Alice`’’.

4.4.1 Instance

A distinção entre as instâncias de um mesmo serviço disponibilizado na rede é feita por meio da `<Instance>`. No exemplo anterior, o mesmo serviço "*myapp*" possuía dois recursos, um denominado "*Alice._ myapp._ tcp.local*" e o outro "*Bob._ myapp._ tcp.local*". Nesse caso, um possuía a `<Instace>` denominada "*Alice*" e o outro "*Bob*". Os dispositivos IEEE 2030.5-2018 que possuem recursos e conjuntos de funções a serem disponibilizados na rede atribuem à `<Instance>` um nome único de até 63 bytes codificados em UTF-8, para cada par de registros DNS SRV/TXT. Caso ocorram conflitos de nome, eles podem ser resolvidos adicionando um número decimal inteiro entre parêntesis no nome atribuído à `<Instance>`.

4.4.2 Service

A porção relativa ao `<Service>` é composta pelo nome do serviço precedido por um símbolo de *underscore* (_). No caso do IEEE 2030.5-2018, o nome do serviço é `smartenergy`. Esse nome foi devidamente registrado na *Internet Assigned Numbers Authority* (IANA) [33]. Por exemplo, uma designação válida de *Service Instance Name* pode ser:

- `device-000001111114._smartenergy._tcp.site.`

onde "`device-000001111114`" é a porção relacionada à `<Instance>`, "`smartenergy`" é o nome do serviço, "`tcp`" é o protocolo de transporte e "`site`" é a porção relacionada ao `<Domain>`.

4.4.3 Domain

A atuação do mDNS é restrita à rede local sob o domínio "`.local`". Já o xmDNS expande a atuação do mDNS para além da rede local com o uso de solicitações e respostas *multicast site-local* sob o domínio (`<Domain>`) "`.site`". Sendo assim, devido à ampliação do alcance das solicitações e respostas com o uso do xmDNS, ele é normativo para o IEEE 2030.5-2018.

4.4.4 Registro DNS TXT

O registro DNS TXT permite o envio de informações adicionais aos solicitantes presentes tanto dentro quanto fora do domínio. O registro pode ser usado para verificar propriedades de domínio, informar caminho relativo, adicionar assinatura digital em e-mail e evitar saída de *spam*. No caso do IEEE 2030.5-2018, o registro DNS TXT possui a finalidade específica de envio dos seguintes parâmetros:

- `txtvers` – deve ser o primeiro parâmetro no registro DNS TXT. Ele deve conter o valor 1 e solicitações que apresentem o parâmetro com valor vazio ou diferente de 1 devem ser descartadas.
- `dcap` – fornece o caminho relativo do recurso que é usado para informar todos os recursos que determinado dispositivo disponibiliza na rede. Deve sempre estar presente com um valor não vazio. Caso apresente um valor vazio, o registro DNS TXT deve ser descartado.
- `path` – fornece o caminho relativo usado para localizar um recurso ou conjunto de funções específico em resposta a uma consulta de subtipo.
- `https` – usado para indicar se o recurso ou conjunto de funções requer conexão segura para a troca de informação. Caso um valor esteja presente, o solicitante deve usar conexão segura na porta especificada pelo valor.
- `level` – usado para indicar nível de extensibilidade do *schema*. Caso o valor presente seja "`-S1`", indica que não são aceitas *tags* não definidas no *schema*. Caso o valor seja "`+S1`", *tags* arbitrárias são aceitas.

A Tabela 4.3 consolida e apresenta exemplos dos parâmetros do registro DNS TXT que são utilizados pelo padrão.

Tabela 4.3: Parâmetros do registro DNS TXT

Parâmetro	Exemplo
txtvers	txtvers = 1
dcap	dcap = /dcap
path	path = /file
https	https = 443
level	level = -S1

4.4.5 Consulta de subtipo

Uma rede pode ser constituída por diversos dispositivos que disponibilizam os mais variados recursos e conjuntos de funções. Fazendo uso do caminho relativo presente no parâmetro "`dcap`" do DNS TXT é possível acessar o recurso que tem por finalidade elencar todos os demais recursos e conjuntos de funções disponibilizados na rede pelo referido dispositivo.

No caso dos conjuntos de funções, em vez de consultar dispositivo por dispositivo a procura de um conjunto de funções em específico, pode-se fazer uso de uma consulta de subtipo. A consulta de subtipo é uma forma de filtrar os pares de registros DNS SRV/TXT a serem retornados de acordo com o desejado. Por exemplo, todos os dispositivos IEEE 2030.5-2018 fazem uso do nome "`smartenergy`" na porção de nome do `<Service>`. Caso um solicitante deseje encontrar o conjunto de funções que forneça suporte a transferência de arquivos entre dispositivos IEEE 2030.5-2018 (denominado "`file`"), em vez de consultar dispositivo por dispositivo a fim de encontrar o que forneça tal conjunto de funções, o solicitante pode fazer uma consulta de subtipo na forma "`file._sub._smartenergy._tcp.site.`" e apenas os dispositivos com tal conjunto de funções irão responder a consulta informando no parâmetro "`path`" do registro DNS TXT o caminho relativo para acesso ao conjunto de funções requerido.

Para que isso seja possível, cada conjunto de funções disponibilizado na rede por um dispositivo deve ter um registro DNS PTR que deverá corresponder à consulta realizada por um solicitante, ou seja, as formas das consultas de subtipo e o registro DNS PTR devem ser iguais. Cada conjunto de funções também deve ter um par de registros DNS SRV/TXT, lembrando que ambos possuem a forma "`<Instance>.<Service>.<Domain>`". No caso em específico dos conjuntos de funções, o nome da `<Instance>` deve conter a *string* de subtipo que identifica o conjunto de funções seguido de um hífen (-) e outra *string* resistente a colisão. O padrão define algumas *strings* de subtipo para identificar os conjuntos de funções. A Tabela 4.4 apresenta as *strings* definidas no padrão. No exem-

Tabela 4.4: *Strings* de subtipo para conjunto de funções

Subtipo	Conjunto de funções IEEE 2030.5-2018
bill	Billing
derp	Distributed Energy Resources
dr	Demand Response and Load Control
edev	End Device
file	File Download
msg	Messaging
mup	Metering Mirror
ppy	Prepayment
rsps	Response
sdev	Self Device
tm	Time
tp	Pricing
upt	Metering

plo anterior, sendo "`file`" a *string* de subtipo que identifica o conjunto de funções para *download* de arquivos, um possível nome para `<Instance>` seria "`file-000001111114`" e o par de registros DNS SRV/TXT referente a tal conjunto de funções possuiria a forma "`file-000001111114._smartenergy._tcp.site.`". A Figura 4.5 demonstra o fluxo DNS-SD utilizando consulta de subtipo para a descoberta do conjunto de funções "`file`".

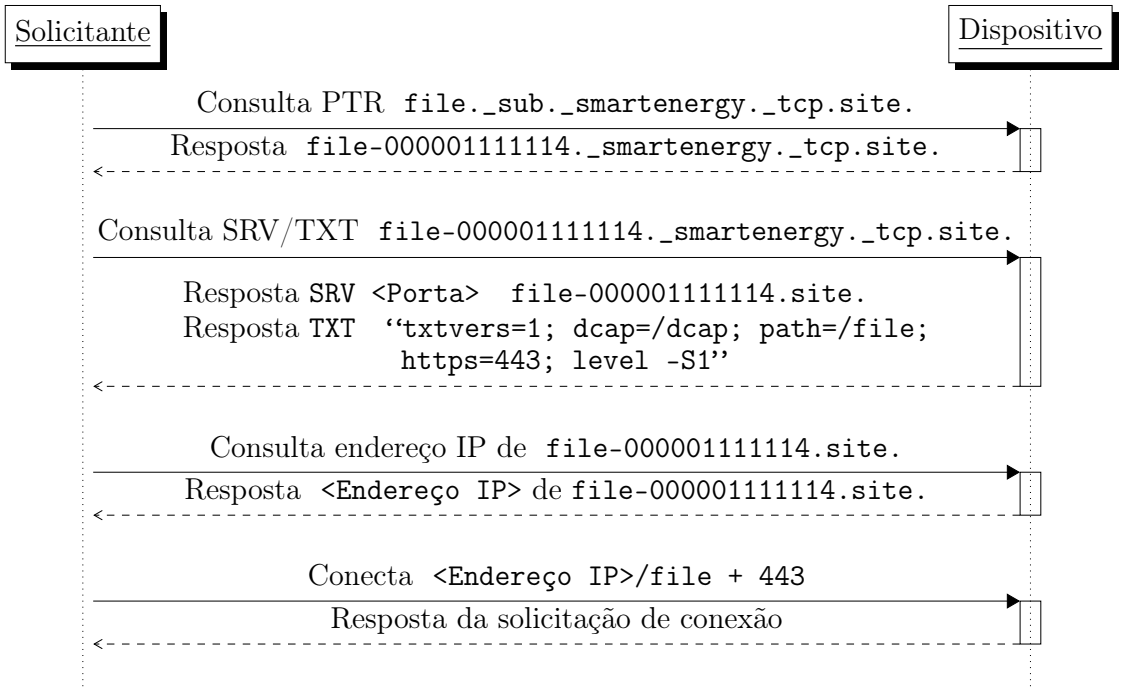


Figura 4.5: Fluxo DNS-SD para localizar conjunto de funções para *download* de arquivos

Todos os registros DNS SRV possuem valores idênticos em um mesmo dispositivo. Eles possuem o *hostname* e a porta padrão de conexão do HTTP que é definida no *schema*. Já os registros DNS TXT diferem entre si pelos parâmetros presentes e o conteúdo de cada um. No caso do exemplo apresentado na Figura 4.5, o parâmetro "`https`" informa a necessidade de realizar conexão segura juntamente com a porta para acessar o conjunto de funções. Já o parâmetro "`path`" informa o caminho relativo onde o conjunto de funções se encontra.

Em resumo, há duas possibilidades para encontrar um conjunto de funções e consequentemente um recurso (lembrando que conjunto de funções são agrupamentos lógicos de recursos): o solicitante pode realizar uma consulta usando ou não uma consulta de subtipo. O que difere é que na consulta que não faz uso de subtipo, o solicitante deve elencar todos os conjuntos de funções disponibilizados pelos dispositivos para verificar qual ou quais deles possuem o conjunto de funções desejado, enquanto que na consulta de subtipo a resposta recebida já identifica o dispositivo, porta e caminho relativo para acesso ao conjunto de funções desejado.

4.5 Conjuntos de funções

Os conjuntos de funções são agrupamentos lógicos de recursos com propósitos em comuns. Um recurso é um objeto acessado por meio da interface RESTful que é disponibilizada por um dispositivo. O IEEE 2030.5-2018 divide os conjuntos de funções em basicamente três categorias:

- ***Support Resources*** – conjuntos de funções que fornecem informações ou serviços operacionais para gerenciar e oferecer suporte à operação, tais como: descoberta de serviços na rede, resposta às solicitações, consulta de dados e informações, serviços de assinatura e notificações de eventos sobre recursos.
- ***Common Resources*** – conjuntos de funções que fornecem funcionalidade de propósito geral, tais como: hora atual, informações sobre a interface de rede, status de energia, serviços de registro de eventos, configuração e serviços de *download* de arquivos.
- ***Smart Energy Resources*** – conjuntos de funções que são específicos do domínio da *Smart Energy*, tais como: consulta do preço da energia em um determinado momento, agendamento para grandes demandas de fluxo de energia (*e.g.*, carregamento

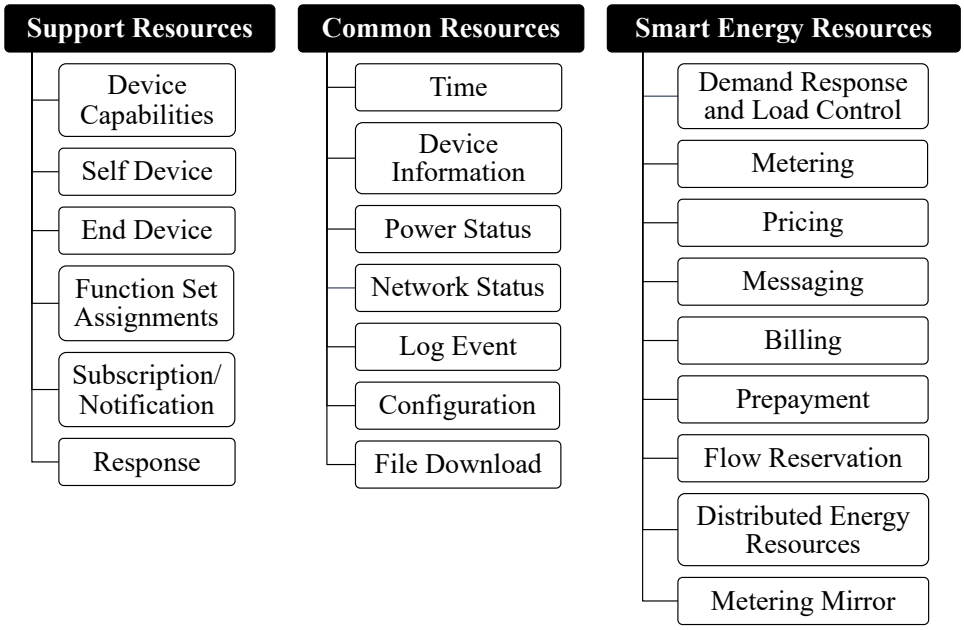


Figura 4.6: Categorias e conjuntos de funções do IEEE 2030.5-2018

de um carro elétrico), envio de informações de medição, controle de gerenciamento de recursos de energia distribuída, serviços de mensagens, provisão de custos, estimativa e histórico de consumo e provisão de serviços com base em crédito ou dívida pendente.

A Figura 4.6 exibe as três categorias e seus respectivos exemplos de conjuntos de funções que os integram. O Apêndice A apresenta de forma mais detalhada os conjuntos de funções previsto no IEEE 2030.5-2018 juntamente com os recursos presentes em cada um. Um conjunto de função pode conter tanto um recurso individual como uma lista de recursos. As listas de recursos são conjuntos de recursos agrupados de forma ordenada. Eles são ordenados de acordo com os atributos que eles possuem em comum, conforme critérios previstos no próprio IEEE 2030.5-2018. Esta ordem é importante porque é possível consultar os recursos que compõem uma lista filtrando-os de acordo com suas posições ou valores das chaves. A sintaxe geral da consulta de recursos é `{URI}?s={x}&a={y}&l={z}`, onde:

- *{URI}* – representa o endereço da lista de recursos;
- *s* ("start") – indica a posição ordinal *{x}* do primeiro recurso a compor a lista de resultados desejado. Composto de números decimais e as posições são contadas a partir do índice 0;

```
<MyTypeList href="http://host1/the/list" all="7" results="7">
  <MyType href="http://host1/instance/of/type/red">
    <timeStamp>100</timeStamp>
  </MyType>
  <MyType href="http://host2/instance/of/type/green">
    <timeStamp>200</timeStamp>
  </MyType>
  <MyType href="http://host3/instance/of/type/blue">
    <timeStamp>300</timeStamp>
  </MyType>
  <MyType href="http://host4/instance/of/type/yellow">
    <timeStamp>400</timeStamp>
  </MyType>
  <MyType href="http://host5/instance/of/type/black">
    <timeStamp>500</timeStamp>
  </MyType>
  <MyType href="http://host6/instance/of/type/white">
    <timeStamp>600</timeStamp>
  </MyType>
  <MyType href="http://host7/instance/of/type/orange">
    <timeStamp>700</timeStamp>
  </MyType>
</MyTypeList>
```

Figura 4.7: Recurso `MyTypeList`

- *a* ("after") – usado em listas que possuem *timestamps* como chave, indica que somente atributos com datas posteriores a *{y}* são desejadas. Composto de um número decimal de 64 bits;

- *l* ("limit") – indica o número máximo *{z}* de itens que podem ser incluídos na lista de resultados, sendo 1 (um) o padrão. Composto de número decimal.

Uma consulta a uma lista de recursos também retorna os atributos *all* e *results*, sendo que o primeiro indica o número total de itens que a lista possui originalmente e o segundo informa o número de itens retornados na resposta da consulta.

Considere a lista apresentada pelo recurso `MyTypeList` conforme demonstrado na Figura 4.7. Ela está ordenada de forma ascendente com relação ao `<timeStamp>`. Alguns exemplos de consultas podem ser realizadas a fim de demonstrar o uso dos parâmetros e seus respectivos retornos. Por exemplo:

- Uma operação GET em `http://host1/the/list?s=0&l=1` irá retornar:

```
<MyTypeList href="http://host1/the/list" all="7" results="1">
  <MyType href="http://host1/instance/of/type/red">
    <timeStamp>100</timeStamp>
```

```
    </MyType>
</MyTypeList>
```

- Uma operação GET em `http://host1/the/list?s=0&l=5` irá retornar:

```
<MyTypeList href="http://host1/the/list" all="7" results="5">
    <MyType href="http://host1/instance/of/type/red">
        <timeStamp>100</timeStamp>
    </MyType>
    <MyType href="http://host2/instance/of/type/green">
        <timeStamp>200</timeStamp>
    </MyType>
    <MyType href="http://host3/instance/of/type/blue">
        <timeStamp>300</timeStamp>
    </MyType>
    <MyType href="http://host4/instance/of/type/yellow">
        <timeStamp>400</timeStamp>
    </MyType>
    <MyType href="http://host5/instance/of/type/black">
        <timeStamp>500</timeStamp>
    </MyType>
</MyTypeList>
```

- Uma operação GET em `http://host1/the/list?s=5&l=1` irá retornar:

```
<MyTypeList href="http://host1/the/list" all="7" results="1">
    <MyType href="http://host6/instance/of/type/white">
        <timeStamp>600</timeStamp>
    </MyType>
</MyTypeList>
```

- Uma operação GET em `http://host1/the/list?a=400&l=4` irá retornar:

```
<MyTypeList href="http://host1/the/list" all="7" results="3">
    <MyType href="http://host5/instance/of/type/black">
        <timeStamp>500</timeStamp>
    </MyType>
    <MyType href="http://host6/instance/of/type/white">
        <timeStamp>600</timeStamp>
    </MyType>
    <MyType href="http://host7/instance/of/type/orange">
        <timeStamp>700</timeStamp>
    </MyType>
</MyTypeList>
```

- Uma operação GET em `http://host1/the/list?a=400&s=2&l=2` irá retornar:

```
<MyTypeList href="http://host1/the/list" all="7" results="1">
    <MyType href="http://host7/instance/of/type/orange">
        <timeStamp>700</timeStamp>
    </MyType>
</MyTypeList>
```

Vale lembrar também que os recursos em uma lista também possuem seus próprios endereços, o que permite o acesso direto. Como por exemplo:

- Uma operação GET em `http://host6/instance/of/type/white` irá retornar:

```
<MyType href="http://host6/instance/of/type/white">
    <timeStamp>600</timeStamp>
</MyType>
```

4.5.1 Categoria Support Resources

A categoria *Support Resources* agrupa recursos e conjuntos de funções que têm por finalidade prover informações operacionais para os dispositivos de uma rede IEEE 2030.5-2018, bem como serviços para gerenciar e oferecer suporte a operação dos mesmos.

Os conjuntos de funções que fazem parte dessa categoria são: *Device Capabilities*, *Self Device*, *End Device*, *Function Set Assignments*, *Subscription/Notification* e *Response*. A Tabela 4.5 apresenta os conjuntos de funções com alguns exemplos de recursos que estão presentes nos mesmos.

O conjunto de funções *Function Set Assignments* destaca-se por permitir que um grupo de *EndDevice* utilize a mesma coleção de recursos. Isso facilita o gerenciamento uma vez que não é necessário atribuir individualmente cada recurso a cada *EndDevice*.

Por exemplo, uma concessionária deseja que determinado grupo de medidores inteligentes de uma região em específico utilize uma instância específica do conjunto de funções *Pricing* e *Time*. Para isso, a concessionária atribui a um recurso *FunctionSetAssignments* as referências aos determinados conjuntos de funções de *Pricing* e *Time* desejado. Em seguida, o recurso *FunctionSetAssignments* é atribuído a cada recurso *EndDevice* referente a cada medidor inteligente presente no grupo. Assim, quando cada medidor inteligente acessar seu respectivo recurso *EndDevice*, eles serão direcionados ao recurso *FunctionSetAssignments* para receberem as URIs relativas aos recursos *Pricing* e *Time* que deverão fazer uso.

Tabela 4.5: Exemplos de conjunto de funções e recursos da categoria Support Resources

Conjunto de Função Recurso	**Descrição**
Device Capabilities	enumera os recursos e os conjuntos de funções suportados por um dispositivo.
DeviceCapability	recurso utilizado na enumeração de recursos e conjuntos de funções que estão presentes em um dispositivo.
Self Device	fornece informações gerais sobre o dispositivo (*e.g.*, fabricante e número da versão).
SelfDevice	recurso utilizado para prover informações gerais.
End Device	armazenado em um dispositivo servidor/controlador da HAN; é semelhante a uma ficha de registro a qual contém informações sobre determinado ativo; promove a troca de informações, tanto gerais quanto específicas.
EndDevice	recurso que provê a interface para troca de informações.
EndDeviceList	recurso que retorna a lista dos *EndDevice* hospedados em determinado dispositivo.
Function Set Assignments	define uma coleção de referências para instâncias de conjuntos de funções a serem usadas por um grupo específico de dispositivos.
FunctionSetAssignments	recurso que fornece uma coleção de recursos para determinado dispositivo.
Subscription/Notification	fornece mecanismos para que um dispositivo seja informado sobre alguma alteração de status em um determinado recurso.
Subscription	recurso que contém informações relacionadas a uma assinatura para receber informação automaticamente.
Notification	recurso que recebe notificação sobre alterações ocorridas em um recurso ao qual o dispositivo possui assinatura.
Response	fornece uma interface para captura da resposta enviada pelos dispositivos após receberem a ocorrência de determinado evento o qual necessita de resposta.
Response	recurso que fornece um repositório de dados de resposta genérico.
PriceResponse	recurso que fornece resposta a um evento relacionado a preço (*e.g.*, alteração de preço).
TextResponse	recurso que fornece resposta a um evento relacionado a mensagem de texto.

4.5.2 Categoria Common Resources

A categoria *Common Resources* agrupa recursos e conjuntos de funções que fornecem funcionalidades de propósito geral não sendo específico de um domínio.

Os conjuntos de funções que fazem parte dessa categoria são: *Time*, *Device Information*, *Power Status*, *Network Status*, *Log Event*, *Configuration* e *File Download*. A Tabela 4.6 apresenta os conjuntos de funções com alguns exemplos de recursos que estão presentes nos mesmos.

Com relação ao recurso *Time*, é importante destacar o fato de que todos os dispositivos devem implementar tal recurso — seja como cliente, servidor ou ambos — a fim de manter a sincronização. O padrão prevê que as atualizações de tempo devem ser feitas por meio de requisições ao recurso *Time* e, a fim de manter um tráfego mínimo na rede, essas requisições:

- não devem ser mais frequentes do que 1 a cada 15 minutos quando já se tem estabelecido um dispositivo fornecedor do recurso *Time*; e
- não devem ser maior do que 1 a cada 15 segundos, empregando *backoff* exponencial, quando se está procurando um dispositivo fornecedor do recurso *Time*.

Encontrando mais de um recurso *Time* disponível na rede, o dispositivo deve escolher o que possui melhor métrica de qualidade. De acordo com o documento `sep.xsd`, as métricas de qualidade são classificadas por níveis, sendo os níveis mais baixos os de melhor qualidade. Os níveis são:

- Nível 3 - tempo obtido por uma fonte externa autoritativa (*e.g.*, fonte NTP).
- Nível 4 - tempo obtido por uma fonte nível 3.
- Nível 5 - tempo configurado manualmente ou obtido de uma fonte nível 4.
- Nível 6 - tempo obtido por uma fonte nível 5.
- Nível 7 - tempo intencionalmente descoordenado (tempo fora do padrão definido no `sep.xsd`).

Todos os demais valores de níveis são reservados para uso futuro.

Tabela 4.6: Exemplos de conjunto de funções e recursos da categoria *Common Resources*

Conjunto de Função Recurso	**Descrição**
Time	tem por finalidade fornecer a base de tempo para que todos os dispositivos estejam sincronizados.
Time	recurso que fornece a base de tempo.
Device Information	fornece identificação e informações de determinado dispositivo que são disponibilizadas pelo fabricante.
DeviceInformation	recurso utilizado para fornecer identificação e informação de determinado dispositivo.
Power Status	fornece informações sobre a fonte de alimentação atual do dispositivo bem como de outras utilizadas como reserva de energia.
PowerStatus	recurso que fornece informação sobre a fonte de alimentação do dispositivo.
Network Status	fornece informações a respeito da camada de rede (IP) do dispositivo.
IPInterface	recurso que informa o status de uma interface específica de determinado dispositivo.
Log Event	tem por finalidade registrar os eventos gerados pelos dispositivos.
LogEvent	recurso que registra data e hora de um evento significativo detectado por um dispositivo.
Configuration	fornece um acesso centralizado de leitura e gravação da configuração operacional de um dispositivo.
Configuration	recurso que contém os parâmetros de configuração de um dispositivo.
File Download	fornece os mecanismos para realização de suporte remoto seguro e *download* de arquivos nos dispositivos.
File	recurso que contém vários metadados sobre um determinado arquivo.
FileList	recurso que lista as instâncias *File* presente em um dispositivo.

4.5.3 Categoria Smart Energy Resources

Composto por conjuntos de funções que são específicos do domínio da *Smart Energy*. Grande parte desses conjuntos de funções exercem controle sobre os dispositivos e são usados tipicamente para prover o gerenciamento dos mesmos.

Os conjuntos de funções que integram essa categoria são: *Demand Response and Load Control* (DRLC), *Metering*, *Pricing*, *Messaging*, *Billing*, *Prepayment*, *Flow Reservation*, *Distributed Energy Resources* e *Metering Mirror*. A Tabela 4.7 apresenta os conjuntos de funções com alguns exemplos de recursos que estão presentes nos mesmos.

Dentre os conjuntos de funções presentes na categoria *Smart Energy*, o conjunto *Metering Mirror* destaca-se por não ser necessariamente um conjunto voltado ao ambiente elétrico. Ele permite que outros dispositivos de outras áreas e com restrições de comunicação (*e.g.*, energia limitada) possam fazer uso desse recurso a fim de enviar e receber comandos e informações.

Por exemplo, um medidor inteligente de gás pode ter limitação de energia e com isso ele "dorme" e "desperta" em períodos de tempos para enviar informações e receber comandos. Uma concessionária de gás que necessita fazer leituras das medições desse medidor inteligente teria sérias dificuldades, pois necessitaria sincronizar a leitura com o exato momento em que o medidor inteligente está ativo para a comunicação. Em vez disso, o medidor inteligente pode fazer uso do recurso *MirrorUsagePoint*, em um dispositivo que forneça tal recurso, e criar uma instância para que possa enviar suas medições quando estiver ativo. Assim, a concessionária de gás pode fazer a leitura das medições do medidor inteligente acessando o dispositivo no qual o medidor criou a instância.

Tabela 4.7: Exemplos de conjunto de funções e recursos da categoria Smart Energy

Conjunto de Função Recurso	**Descrição**
DRLC	fornece uma interface para o controle de carga e resposta a demanda.
EndDeviceControl	recurso que contém os parâmetros de controle para instruir um *EndDevice* a executar uma ação específica.
Metering	fornece uma interface para a troca de informações sobre leituras e medições das *commodities*.
UsagePoint	recurso que representa um ponto lógico onde consumo e/ou produção são medidos fisicamente.
MeterReading	recurso que contém conjunto de valores obtidos dos medidores.
Pricing	fornece modelos de estruturas tarifárias com suporte a uma variedade de tipos (*e.g.,* preço por hora e preço por consumo).
TariffProfile	recurso que fornece um código de cobrança tarifário por meio de uma tabela de cobrança.
Messaging	fornece uma interface para o serviço de troca de mensagens de texto.
TextMessage	recurso que contém mensagem de texto individual usada em uma notificação.
Billing	tem por finalidade fornecer o consumo, custo e estimativa futura de consumo das *commodities*.
ProjectionReading	recurso que contém valores previstos em uma leitura futura para um dado intervalo de tempo específico.
Prepayment	fornece mecanismos para o fornecimento de determinada *commodity* baseado em crédito ou débito do usuário.
Prepayment	recurso que contém informações sobre crédito e/ou débito do usuário.
Flow Reservation	fornece uma interface para a reserva de eventos de troca de fluxo de energia (*e.g.*, carga ou descarga).
FlowReservationRequest	recurso que contém informações sobre a necessidade de reserva de um período de tempo para realização de uma operação de troca de fluxo.
DER	fornece uma interface para gerenciamento das *Distributed Energy Resources*.
DER	recurso que contém informação sobre uma determinada DER.
Metering Mirror	fornece mecanismos para facilitar que dispositivos com restrições (*e.g.*, limitação de energia) possam enviar seus dados de maneira mais eficiente.
MirrorUsagePoint	recurso que fornece suporte idêntico ao *UsagePoint* para dispositivos com limitações.

Capítulo 5

Segurança

A segurança é um requisito fundamental em se tratando de acesso a dispositivos e tráfego de dados sensíveis. No entanto, deve haver um equilíbrio, pois sistemas que implementam segurança em excesso podem não ter um tempo de resposta adequado. Da mesma forma, sistemas altamente responsivos podem não ter uma segurança adequada.

Dado que os dispositivos em uma DER/HAN enviam informações e podem ser controlados e gerenciados remotamente pelas concessionárias, é necessário fornecer uma comunicação e um acesso seguro para evitar que terceiros mal intencionados tenham acesso e executem ações prejudiciais, tais como o envio de comandos não autorizados ou roubo de informações. Um terceiro mal intencionado pode enviar um comando malicioso que pode dar início a uma reação em cadeia prejudicando todo o sistema elétrico. Em 2008, um ataque cibernético proveniente da Rússia derrubou todo o sistema de energia da Geórgia na guerra Rússia-Geórgia [26]. Em 2015, um ataque cibernético no sistema elétrico da Ucrânia causou um *blackout* que atingiu aproximadamente 225 mil consumidores por diversas horas [26].

Portanto, segurança no acesso e na comunicação entre dispositivos torna-se uma grande preocupação. Sendo assim, os dispositivos IEEE 2030.5-2018 que compõem uma DER/HAN devem garantir que apenas partes autorizadas tenham acesso aos recursos que eles hospedam; reciprocamente, as partes autorizadas devem ser capazes de confirmar a autenticidade do nó remoto com o qual estão se comunicando. Além disso, os dispositivos também devem fazer uso de protocolos adequados de segurança a fim de garantir uma comunicação segura.

No IEEE 2030.5-2018, as transações seguras entre os dispositivos são feitas por meio de requisições HTTP encapsuladas em *Transport Layer Security* (TLS) [54] constituindo

assim o HTTPS. O mecanismo de *handshake* do TLS fornece autenticação mútua entre os dispositivos baseada nos certificados digitais presentes em ambos. Além disso, os registros TLS fornecem criptografia e autenticação de mensagens.

5.1 Credenciais dos dispositivos

No IEEE 2030.5-2018, cada dispositivo é provido de três credenciais que o identificam:

- *Short Form Device Identifier* (SFDI);
- *Long Form Device Identifier* (LFDI); e
- PIN.

O SFDI e o LFDI são derivados da impressão digital que por sua vez é proveniente do certificado digital do dispositivo. A impressão digital é o resultado da aplicação de uma operação *hash* SHA-256 sobre o certificado digital em codificação *Distinguished Encoding Rules* (DER). O SFDI é composto dos 36 bits mais significativos da impressão digital, expressos em decimal, concatenados com um dígito de *checksum*, totalizando 12 dígitos decimais. Ele é usado para identificar o dispositivo dentro de uma HAN/domínio. O LFDI é composto dos 160 bits mais significativos da impressão digital, expressos em hexadecimal, sendo usado para identificar o dispositivo em âmbito externo da HAN/domínio. O PIN é formado por uma sequência aleatória de 5 dígitos decimais concatenados com mais um dígito de *checksum*, totalizando 6 dígitos decimais. Ele pode ser compartilhado fora de banda para ser usado em conjunto com o SFDI e o LFDI.

O SFDI e o PIN, em certos casos, podem ser usados em conjunto para formar um código simples de registro de dispositivo. Portanto, a simples concatenação do SFDI com o PIN (SFDI || PIN) dá origem ao chamado "Código de Registro" que é um número decimal de 18 dígitos.

5.2 Autenticação

Para assegurar que os recursos serão acessados por dispositivos legítimos e com os devidos privilégios de acesso, procedimentos de autenticação e autorização dos dispositivos solicitantes são realizados de acordo com a necessidade de cada recurso conforme definido nas políticas de segurança do sistema. Dependendo do recurso e das políticas de segurança,

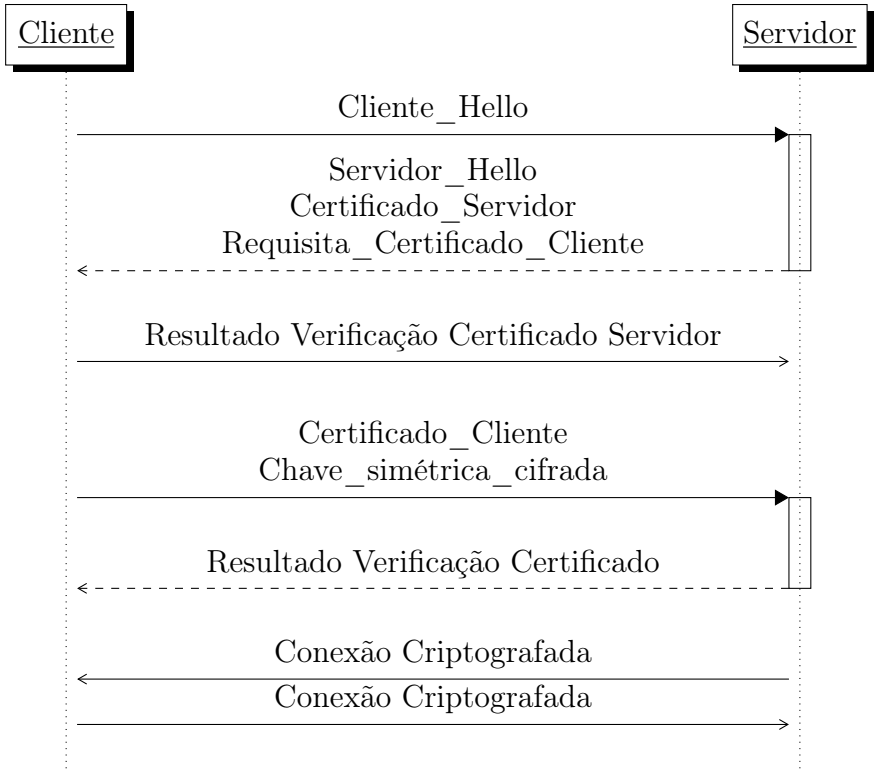


Figura 5.1: Fluxo autenticação mútua TLS

os usos de autenticação, confidencialidade de dados e verificação de integridade podem ser necessários. Para esses recursos, o acesso deve ocorrer por meio de requisições HTTPS. Para recursos sem tais requisitos e para os quais as políticas de segurança permitem, o acesso pode ser feito por meio de solicitações HTTP não criptografadas.

A autenticação dos dispositivos é fornecida pelo TLS. Ele fornece confidencialidade, autenticação de mensagens e autenticação mútua de dispositivos com base nos certificados digitais.

A Figura 5.1 apresenta a demonstração simplificada do fluxo TLS seguido para ocorrer a autenticação mútua entre dois dispositivos quaisquer denominados “Cliente” e “Servidor”. Ao receber uma solicitação *hello* HTTPS, o dispositivo servidor inicia o *handshake* TLS. Durante este procedimento, o dispositivo servidor envia seu certificado digital ao dispositivo cliente para fins de validação e solicita o certificado digital do mesmo. O dispositivo cliente verifica o certificado digital do servidor conforme especificado na RFC 5246 [55]. Após a verificação bem sucedida, o dispositivo cliente envia seu certificado digital juntamente com uma mensagem contendo a chave simétrica cifrada na chave pública do dispositivo servidor. Ao receber o certificado digital do cliente, o servidor também realiza a verificação para os mesmos fins de validação. Sendo bem sucedida a verificação, o dispositivo servidor decifra a mensagem utilizando a chave privada e recupera a chave

simétrica. A partir de então, as mensagens trocadas entre os dispositivos passam a ser cifradas utilizando a chave simétrica.

A mesma dinâmica ocorre entre os dispositivos IEEE 2030.5-2018. O uso de certificado digital nos dispositivos que hospedam os recursos é obrigatório. Eles agem como servidores e o uso de certificados digitais auto-assinados não é permitido. Os dispositivos solicitantes agem como clientes fazendo uso dos recursos, podendo ou não fazer uso de certificados digitais e, se o fizerem, tais certificados digitais podem ser auto-assinados.

O uso ou não de certificados digitais nos dispositivos solicitantes depende da política de segurança implementada para cada sistema. Sendo assim, caso o dispositivo solicitante não faça uso de certificado digital, ele não será autenticado utilizando o TLS, mas pode vir a ser autenticado por meio de uma autenticação secundária designada pela política de segurança do sistema.

5.3 Autorização

A autorização para acesso aos recursos é obtida por meio de uma *Access Control List* (ACL). Ela fornece mecanismos que especificam quais dispositivos têm acesso a determinados recursos e quais operações podem ser executadas nesses recursos. Cada recurso pode ter uma ACL específica. Se o recurso não tiver um acesso regulado por ACL, o acesso é permitido para qualquer dispositivo.

Normalmente, a ACL define um privilégio padrão e pode conceder privilégios adicionais para dispositivos específicos. Tanto o privilégio padrão quanto os privilégios adicionais devem seguir a política de segurança designada para o sistema. Para que seja permitido o acesso de um determinado dispositivo a um determinado recurso, os atributos presentes na ACL devem corresponder aos atributos apresentados pelo dispositivo solicitante. No caso do IEEE 2030.5-2018, esses atributos são:

- `IPAddr` – endereço IP do solicitante;
- `Port` – número da porta do solicitante;
- `Method` – método HTTP utilizado pelo solicitante;
- `AuthType` – tipo de autenticação feita pelo solicitante (sem autenticação, autenticação secundária, certificado digital auto-assinado ou certificado digital emitido por autoridade certificadora); e

Tabela 5.1: Atributos presentes na ACL.

Atributo	Tipo	Descrição
IPAddr	IPAddr	Endereço IP do dispositivo solicitante
Port	Inteiro	Número da porta do dispositivo solicitante
Method	Bitmap	Método HTTP: 0x1: GET 0x2: PUT 0x4: POST 0x8: DELETE 0x10: HEAD
AuthType	Inteiro	0x1: Sem autenticação 0x2: Autenticação secundária 0x4: Certificado auto-assinado 0x8: Certificado emitido por autoridade
DeviceType	Inteiro	Baseado no `OBJECT IDENTIFER` do certificado digital

- DeviceType – tipo de dispositivo solicitante (baseado no `OBJECT IDENTIFER` presente no certificado digital do dispositivo solicitante).

Se houver uma correspondência entre os atributos apresentados e os presentes na ACL, o acesso ao recurso é permitido. Caso contrário, o acesso está condicionado ao privilégio padrão especificado pela ACL. A Tabela 5.1 consolida e apresenta os atributos presentes na ACL do IEEE 2030.5-2018, bem como os tipos de cada um e suas respectivas descrições.

5.4 Registro de dispositivo

Para acesso a determinados recursos específicos, pode ser necessário o registro prévio do dispositivo solicitante no dispositivo que hospeda tais recursos. O processo de registro informa ao dispositivo que hospeda os recursos o SFDI e, opcionalmente, o PIN do dispositivo solicitante. Essas informações identificam exclusivamente o dispositivo solicitante em um determinado contexto.

O procedimento de envio das informações de cadastro ao dispositivo que hospeda o recurso desejado é feito através de um canal fora de banda. Por exemplo, usando uma interface *web* ou até mesmo um acesso direto ao dispositivo que hospeda os recursos é

possível cadastrar previamente as informações SFDI (também, opcionalmente, o PIN) de quem irá acessar os recursos. Esse cadastro cria uma instância *EndDevice* no dispositivo que hospeda os recursos. Assim, quando o dispositivo solicitante realiza o primeiro acesso, ele consulta o recurso *EndDeviceList* para verificar se já existe uma instância dele previamente criada. Essa verificação é feita pela comparação do SFDI do dispositivo solicitante com o SFDI presente nas instâncias *EndDevice* listadas na *EndDeviceList*. Caso tenha sido cadastrado o PIN, o dispositivo solicitante continua a verificação acessando o recurso *Registration* associado à sua instância *EndDevice* para verificação do PIN. Havendo correspondência do PIN cadastrado no dispositivo que hospeda os recursos e o PIN presente no dispositivo solicitante, então há a confirmação do registro do dispositivo solicitante. Nos casos onde não há PIN cadastrado, a verificação se limita ao SFDI.

Há também a possibilidade do dispositivo solicitante não ser previamente cadastrado e solicitar seu cadastro a partir do primeiro acesso ao dispositivo que hospeda os recursos. Por exemplo, ao verificar a *EndDeviceList*, o dispositivo solicitante verifica que não exite nenhuma instância *EndDevice* correspondente ao seu SFDI. Sendo assim, ele solicita a criação de uma instância *EndDevice* correspondente ao seu SFDI. A solicitação fica pendente até que haja uma confirmação por parte do administrador do dispositivo que hospeda os recursos. Havendo a confirmação, a instância é criada e o dispositivo solicitante fica registrado.

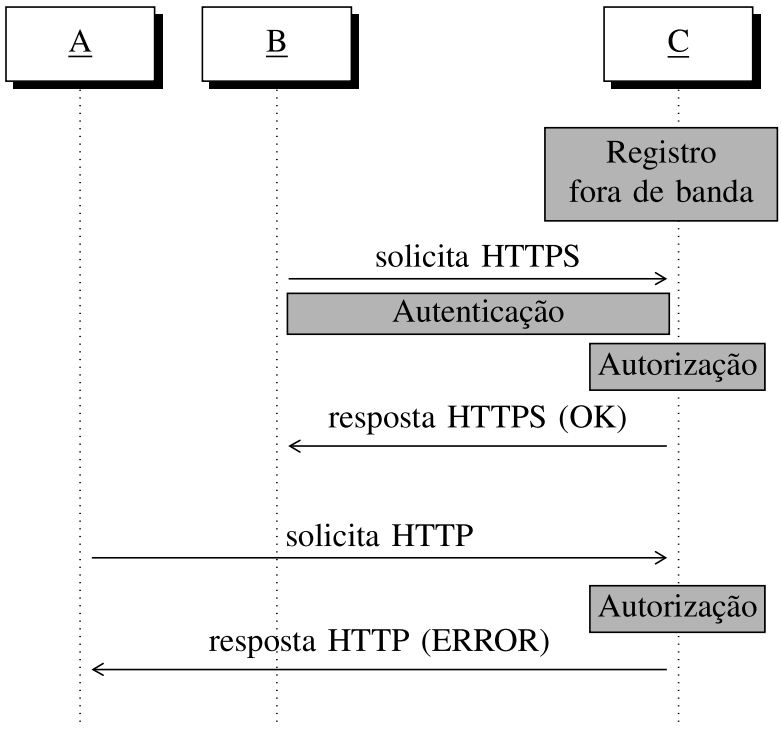


Figura 5.2: Exemplo de registro, autenticação e autorização para acesso à recurso.

Em resumo, para garantir o acesso de dispositivos legítimos e com os devidos privilégios de acesso aos recursos hospedados em determinado dispositivo, métodos de autenticação e autorização são utilizados. Para recursos específicos, além dos métodos de autenticação e autorização, é necessário o registro prévio do dispositivo solicitante.

A Figura 5.2 apresenta um exemplo de dois dispositivos `A` e `B` que necessitam acessar um recurso hospedado em `C` o qual requer comunicação segura e registro prévio. O dispositivo `B` faz uma solicitação HTTPS para `C` a fim de acessar o recurso. Nesse momento, o processo de autenticação tem início juntamente com o processo de verificação do registro prévio. Após o processo de autenticação e verificação do registro prévio finalizar, `C` inicia o processo de autorização, o qual consiste em acessar a ACL do recurso para verificar o direito de acesso com base nas informações apresentadas por `B`. Se as informações corresponderem, como é o caso, o acesso ao recurso é permitido. Por outro lado, `A` faz uma solicitação HTTP para acessar o mesmo recurso. Por ser uma solicitação HTTP, não há procedimento de autenticação. No entanto, há a verificação do registro. Terminado o procedimento de autorização, as informações apresentadas por `A` não correspondem às restrições presentes na ACL do recurso e, portanto, o acesso ao recurso é negado.

5.5 Infraestrutura de Chave Pública

Uma infraestrutura de chave pública — *Public Key Infrastructure* (PKI) — é constituída de elementos e políticas que dão suporte a criação, verificação, distribuição, armazenamento e revogação de certificados digitais, também conhecidos como certificados de chave pública. Além disso, a PKI também fornece suporte para a aplicação da técnica de criptografia assimétrica.

A PKI auxilia no uso eficiente e correto da criptografia assimétrica, pois forma um vínculo seguro entre a chave pública e a entidade proprietária da chave privada correspondente. Para criar esse vínculo confiável entre uma chave pública e uma entidade, são utilizados certificados digitais. O certificado digital é um documento eletrônico emitido por uma Autoridade Certificadora — *Certificate Authority* (CA) — composto por informações e atributos próprios da entidade para a qual ele está sendo emitido. Atualmente, o modelo de certificado digital mais adotado é o X.509 Versão 3 [2].

A CA é o componente principal de uma PKI pois ela é a âncora de confiança do sistema. Sendo assim, todos os demais integrantes do sistema confiam na CA e nos certificados digitais por ela emitidos e assinados. O certificado digital emitido por uma CA

é assinado digitalmente utilizando a chave privada da CA. Isso visa garantir a autenticidade, integridade e o não-repúdio do conteúdo presente em um certificado digital. A verificação da assinatura digital é feita utilizando a chave pública da própria CA. Além de emitir certificados digitais, uma CA pode também renovar e revogar os certificados digitais por ela emitidos.

Os certificados digitais possuem prazo de validade e após a expiração do prazo os mesmos passam a ser inválidos e não confiáveis. Para sanar tal situação, o certificado pode ser renovado. Um certificado digital também pode ser comprometido e não ser mais confiável mesmo estando dentro do prazo de validade. A revogação é o processo pelo qual um certificado digital passa a ser classificado como não mais confiável no sistema. A CA realiza o procedimento de revogação e adiciona o certificado digital na *Certificate Revoked List* (CRL). Sendo assim, para verificar se um certificado digital está válido ou não, é necessário consultar a CRL. O status de um certificado digital também pode ser consultado por meio do protocolo *Online Certificate Status Protocol* (OCSP) [47].

Uma PKI pode ter várias CAs e estas necessitam estabelecer relações de confiança entre si [2]. As relações de confiança entre as CAs podem ser constituídas de diversas maneiras, sendo estas classificadas em modelos de confiança [2]. Dentre os modelos de confiança, o mais comum é o modelo hierárquico no qual as CAs são organizadas de forma hierárquica. Nesse modelo, as CAs superiores emitem certificados para CAs inferiores que por sua vez os utilizam para assinar os certificados digitais que emitem [2]. A CA localizada no topo da estrutura hierárquica é denominada CA raiz. Ela é a unidade âncora detentora de certificado digital auto-assinado no qual todos os demais devem confiar. A Figura 5.3 apresenta um modelo de confiança hierárquico. Nela, a CA raiz emite

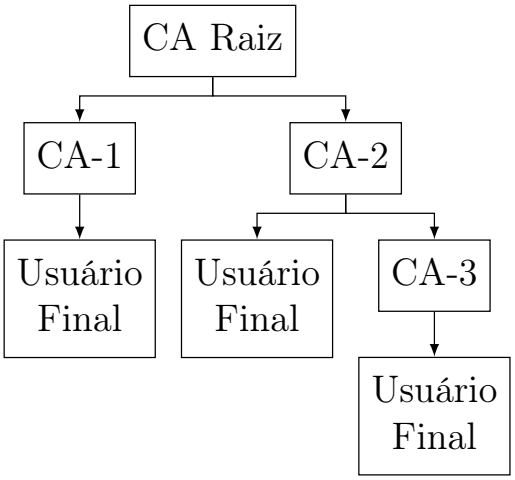


Figura 5.3: Modelo de Confiança Hierárquico

certificado digital para duas outras CAs dando origem a CA-1 e CA-2. A CA-1 emite certificados digitais para os usuários finais. Já a CA-2 emite certificado tanto para usuários finais quanto para dar origem a CA-3, que por sua vez emite certificados para usuários finais.

5.5.1 Manufacturing PKI

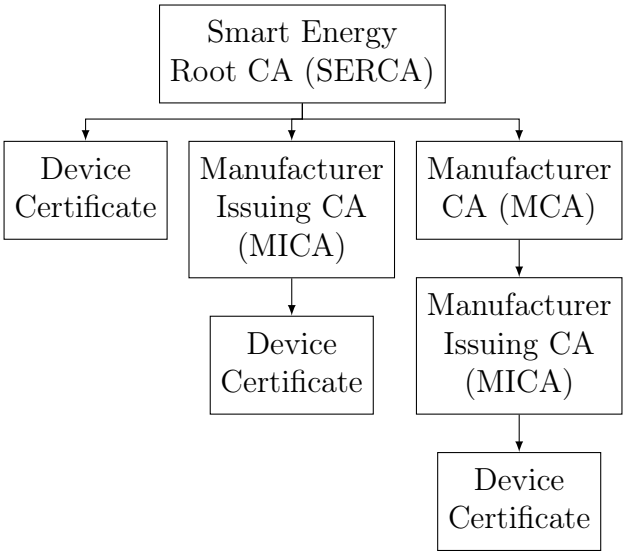
No caso do IEEE 2030.5-2018, o padrão define uma PKI chamada *Manufacturing PKI* para gerenciar os certificados digitais próprios dos dispositivos IEEE 2030.5-2018. Baseado na necessidade de certificados digitais provenientes da *Manufacturing PKI*, espera-se que o mercado implemente uma ou mais *Manufacturing PKI* de acordo com os requisitos presentes no padrão IEEE 2030.5-2018.

A *Manufacturing PKI* é uma estrutura hierárquica com 2, 3 ou 4 níveis, sendo que ao percorrer uma hierarquia, no topo dela deve haver uma única *Smart Energy Root CA* (SERCA), conforme ilustrado no exemplo da Figura 5.4. Os elementos que compõem a *Manufacturing PKI* são:

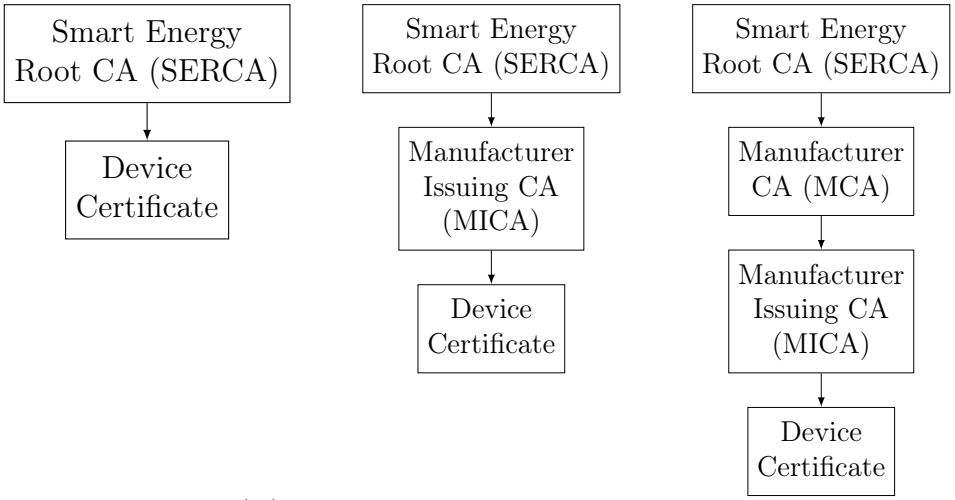
- *Smart Energy Root CA* (SERCA) – autoridade de certificação de nível raiz (*Root*). Pode emitir certificados para dispositivos em nome de um ou mais fabricantes.
- *Manufacturer CA* (MCA) – autoridade de certificação intermediária com o propósito de emitir certificado para MICAs — próximo nível na hierarquia.
- *Manufacturer Issuing CA* (MICA) – autoridade de certificação que emite certificados para dispositivos finais.
- *Device Certificate* – certificado digital que vincula a identidade do dispositivo ao próprio dispositivo.

5.5.2 Certificados Digitais

Os certificados digitais IEEE 2030.5-2018 são certificados X.509 Versão 3 conforme definido na RFC 5280 [6]. Em uma implementação IEEE 2030.5-2018, os certificados emitidos por uma *Manufacturing PKI*, por uma autoridade genérica — *e.g.*, fornecidos por uma autoridade fora da *Manufacturing PKI* — e até mesmo certificados auto-assinados podem coexistir. Há seis classes de certificados digitais que, a depender da configuração do sistema, podem estar ativos:



(A) Única *Manufacturing PKI*



(B) Múltiplas *Manufacturing PKI*

Figura 5.4: Exemplo de estrutura hierárquica de *Manufacturing PKI*.

- Certificado de dispositivo – certificado nativo IEEE 2030.5 emitido por uma *Manufacturing PKI* para fins operacionais;
- Certificado de teste de dispositivo – certificado nativo IEEE 2030.5 emitido por uma *Manufacturing PKI* para fins de testes;
- Certificado adicional para dispositivo IEEE 2030.5 – certificado TLS opcional emitidos por uma CA não *Manufacturing PKI* para dispositivo IEEE 2030.5 (*e.g.*, HEMS);
- Certificado genérico de cliente não nativo – certificado TLS emitido por uma CA não *Manufacturing PKI* para dispositivo cliente (consumidor de recursos);
- Certificado genérico de servidor não nativo – certificado TLS emitido por uma CA não *Manufacturing PKI* para dispositivo servidor (provedor de recursos);
- Certificado auto-assinado não nativo – certificado TLS emitido e auto-assinado pelo dispositivo solicitante.

A Tabela 5.2 demonstra a interação entre esses diferentes tipos de certificados. O dispositivo solicitante que possui certificado IEEE 2030.5 pode interagir apenas com dispositivo que hospeda recurso cujo certificado não seja auto-assinado. Se o solicitante ou o dispositivo que hospeda recurso usa um certificado genérico, a validação do certificado opcionalmente pode usar o OCSP. Para dispositivo solicitante e dispositivo que hospeda recurso que usem certificados IEEE 2030.5, a verificação é limitada apenas a validar as assinaturas da cadeia de certificados.

Tabela 5.2: Autenticação solicitante-hospedeiro por tipo de certificado

		Dispositivo que hospeda recursos		
		Certificado IEEE 2030.5	**Certificado Genérico**	**Certificado Auto-assinado**
Solicitante	**Certificado IEEE 2030.5**	IEEE 2030.5 indef [1]	OCSP opcional	Não permitido
	Certificado Genérico	OCSP opcional	Não especificado	Não especificado
	Certificado Auto-assinado	Validação de assinatura	Não especificado	Não especificado

[1] indefinidamente válidos e a verificação é limitada apenas a uma verificação das assinaturas na cadeia de certificados.

Diferentemente dos certificados digitais provenientes de outras PKIs, os certificados digitais da *Manufacturing PKI* têm um ciclo de vida indefinido e, uma vez emitidos, não podem ser revogados. Como resultado, nenhuma MCA, MICA ou SERCA deve emitir CRL e nenhum MCA ou MICA deve operar ou ter terceiros operando em seu nome servidores OCSP. Entretanto, o certificado de uma CA pode vir a ser inutilizado por diversos motivos. Por exemplo, uma empresa privada operando uma CA pode deixar de fornecer o serviço ao mercado. Sendo assim, o certificado digital e a chave privada a ele associados não devem mais serem utilizados para gerar novos certificados. No entanto, os certificados digitais por ele emitidos ainda devem ser considerados válidos.

5.6 Análise de Segurança

Tomando como base os elementos de segurança e as suas características presentes no padrão IEEE 2030.5-2018, é possível notar semelhanças e diferenças quanto à segurança utilizada em outros sistemas e plataformas. Por exemplo, a utilização de certificados digitais para promover a autenticidade e criptografia para promover o sigilo são elementos de segurança bem conhecidos e utilizados no tráfego de dados sensíveis na Internet. Entretanto, diferentemente do que expressamente determina o IEEE 2030.5-2018, na Internet são utilizados tanto o CRL quanto o protocolo OCSP para verificação da validade e legitimidade dos certificados digitais.

Diversos servidores na Internet, principalmente de bancos, utilizam certificados digitais a fim de promover uma comunicação segura por meio do HTTPS. O usuário, ao acessar determinado servidor utilizando o HTTPS, recebe o certificado digital daquele servidor. Logo em seguida, é realizada uma consulta para verificar se o certificado é válido. Essa consulta é feita utilizando CRL ou OCSP. Também ocorre outras verificações tais como a verificação da assinatura digital tanto do próprio certificado quanto dos demais certificados que compõem a cadeia até o certificado da autoridade certificadora raiz. Sendo válido o certificado, o usuário inicia a comunicação criptografada (HTTPS) com o servidor para a troca de mensagens e dados sensíveis. Já a comunicação segura do IEEE 2030.5, diferentemente da comunicação que ocorre na Internet usando HTTPS, não faz uso de CRL nem OCSP para a verificação dos certificados digitais. A consulta se limita apenas uma verificação da cadeia de assinaturas presente nos certificados. Sendo assim, em não ocorrendo a verificação em tempo real dos certificados, eles são sempre considerados válidos caso não ocorra nenhum erro na verificação da cadeia de assinaturas, independente da situação.

Um dos motivos para a não utilização de CRL e OCSP no IEEE 2030.5-2018 advém do fato de que eles geralmente são hospedados pela CA na Internet e não há nenhuma disposição expressa de que todos os dispositivos IEEE 2030.5 terão acesso à Internet [49]. Sendo assim, os dispositivos não seriam capazes de fazer uso desses mecanismos para verificarem os certificados digitais. Outra razão é que não há uma definição específica de quem é responsável por relatar o comprometimento de um certificado digital (violação da confidencialidade da chave privada a ele associada), uma vez que muitos dispositivos devem operar sem intervenção humana e o proprietário do dispositivo pode nem saber que o certificado digital do dispositivo está comprometido [49].

O fato de a *Manufacturing PKI* emitir certificados não-revogáveis com um ciclo de vida indefinido pode causar sérios problemas de segurança. Se um dispositivo tiver um certificado comprometido, o mesmo não pode ser revogado e nunca irá expirar. Sendo assim, o certificado digital comprometido pode vir a ser usado por um dispositivo malicioso para perpetrar algum tipo de ataque.

Uma medida que pode ser implementada é controlar ainda mais o acesso aos recursos por meio de *black lists* e *white lists* de certificados digitais. No entanto, como a responsabilidade de relatar certificados comprometidos não está claramente definida e muitos dos dispositivos são concebidos para funcionar sem a intervenção humana, o uso de *black/white lists* pode levar à inconsistência dessas listas nos vários dispositivos que compõem o sistema. Também é preciso levar em conta que as *black lists* tendem a aumentar de tamanho indefinidamente em contraste com a capacidade limitada de *hardware* dos dispositivos. Uma vez que os certificados digitais não possuem data de expiração, ao ser incluído em uma lista, ele nunca poderá ser excluído. Consequentemente, o tamanho da lista tende a aumentar indefinidamente, o que é inviável para dispositivos que tipicamente possuem *hardware* limitado.

A questão relativa ao ciclo de vida indefinido e a não revogação dos certificados digitais toma maiores proporções quando consideramos os certificados das MCAs e das MICAs. Uma vez que elas são capazes de emitirem novos certificados digitais — tanto para dispositivos clientes, no caso das MICAS, quanto para outras autoridades emissoras de certificados, no caso das MCAs — caso um terceiro seja capaz de comprometer os certificados de tais entidades, ele será capaz de emitir certificados arbitrários para dispositivos maliciosos que serão considerados legítimos e passarão pelo crivo de validação, mesmo sendo provenientes de uma fonte ilegítima.

Sendo assim, uma possível lacuna de segurança pode ser identificada na *Manufactu-*

ring PKI. Tal lacuna se refere ao problema do comprometimento dos certificados digitais e a sua não expiração e não revogação. Por ser uma estrutura hierárquica, dependendo do nível da hierarquia que o certificado digital comprometido ocupa, os danos podem tomar maiores proporções impactando toda a cadeia hierárquica. Portanto, o comprometimento de certificado digital é uma grande preocupação para a qual o padrão não prevê especificamente uma solução.

Em suma, em um sistema IEEE 2030.5, exite a possibilidade dos seguintes comprometimentos de certificado digital:

- Comprometimento de certificado digital de MCA e MICA.
- Comprometimento de certificado digital de dispositivo que hospeda recursos.
- Comprometimento de certificado digital de dispositivo que solicita recursos.

A seguir, são apresentados possíveis ataques e suas consequências no sistema IEEE 2030.5 tomando como base cada um desses tipos de comprometimento.

5.6.1 Comprometimento de certificado digital de MCA e MICA

As MCAs e MICAs, como o próprio padrão IEEE 2030.5-2018 prevê, podem ser operadas por empresas privadas ou pelos próprios fabricantes de dispositivos IEEE 2030.5. Sendo assim, elas poderão ter políticas de segurança próprias, a depender da empresa fabricante, para promover a guarda segura da chave privada que irá assinar os certificados digitais por elas emitidos.

As MCAs emitem certificados somente para MICAs. Já as MICAs emitem certificados somente para dispositivos IEEE 2030.5. Um fornecedor de dispositivos IEEE 2030.5 pode terceirizar a guarda e a hospedagem da chave privada da MCA para uma empresa CA comercial. A chave privada de uma MICA também pode ser terceirizada para uma empresa CA comercial, mas, por emitir certificados digitais para produtos finais IEEE 2030.5, presume-se que ela irá ficar em posse do fabricante a fim de agilizar o processo de produção dos certificados digitais em tempo de fabricação dos dispositivos.

Partindo-se do pressuposto de que tanto a chave privada da MCA quanto da MICA de um determinado fabricante encontram-se em posse do mesmo e sob suas políticas de segurança, um atacante pode explorar falhas a fim de comprometer os certificados digitais de ambas. Falha no sistema, na rede do fabricante, no processo ou até mesmo falha humana

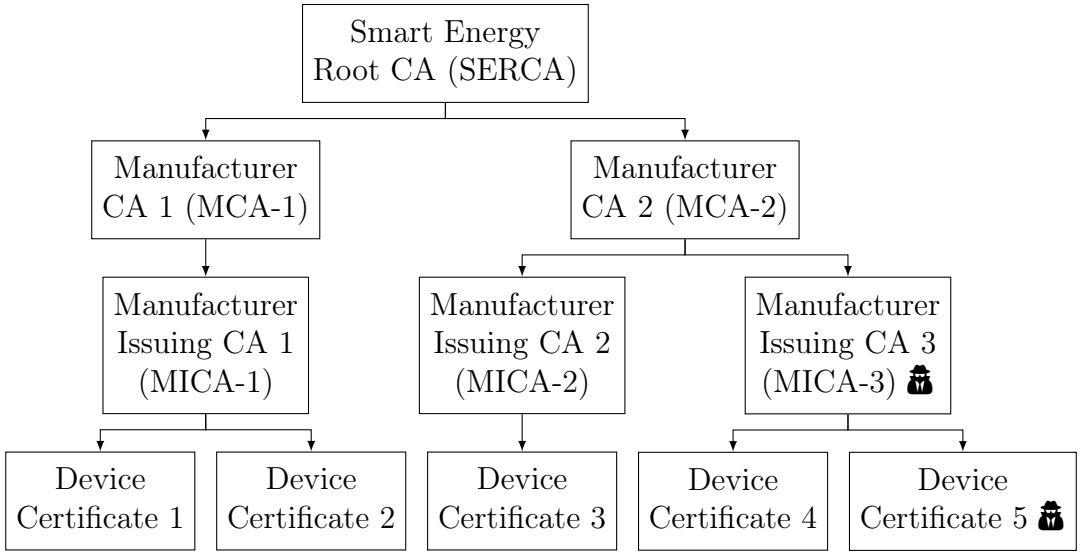


Figura 5.5: Comprometimento da estrutura hierárquica da *Manufacturing PKI*.

– no caso de engenharia social – podem ser exploradas por um atacante a fim de obter êxito em seu ataque. Ou seja, operar uma CA não é uma atividade fim dos fabricantes e, por isso, eles talvez tenham menos condições técnicas para geri-las com segurança. O mais preocupante é que, mesmo que seja descoberta em tempo uma falha que comprometa um certificado digital, não há nenhuma forma prevista pelo padrão para solucionar o possível comprometimento do certificado digital e suas possíveis consequências. Embora o uso de *black lists* possa ser uma das soluções, há sérias desvantagens conforme anteriormente elencado.

Um atacante de posse da chave privada associada a um certificado digital de uma MICA pertencente à hierarquia da *Manufacturing PKI*, como por exemplo a MICA-3 da Figura 5.5, pode emitir certificados digitais para diversos dispositivos maliciosos (*e.g.*, *Device Certificate 5* da Figura 5.5). Tais dispositivos podem ser comercializados e integrados a uma rede IEEE 2030.5 sem maiores problemas uma vez que são dotados de certificado digital proveniente de uma MICA pertencente à hierarquia de uma *Manufacturing PKI*. Sendo assim, ao integrarem a rede, tais dispositivos começam a executar a atividade maliciosa. Tal atividade pode variar desde envio de informações falsas até a execução de comandos que venham a gerar falhas no sistema.

A Figura 5.6 apresenta um exemplo da ação de um dispositivo malicioso com certificado digital comprometido (*Device Certificate 5*) proveniente de uma MICA comprometida (MICA-3). O dispositivo malicioso, ao ser adquirido e conectado à rede IEEE 2030.5, pode começar a disponibilizar recursos na rede a fim de se comunicar com os demais dis-

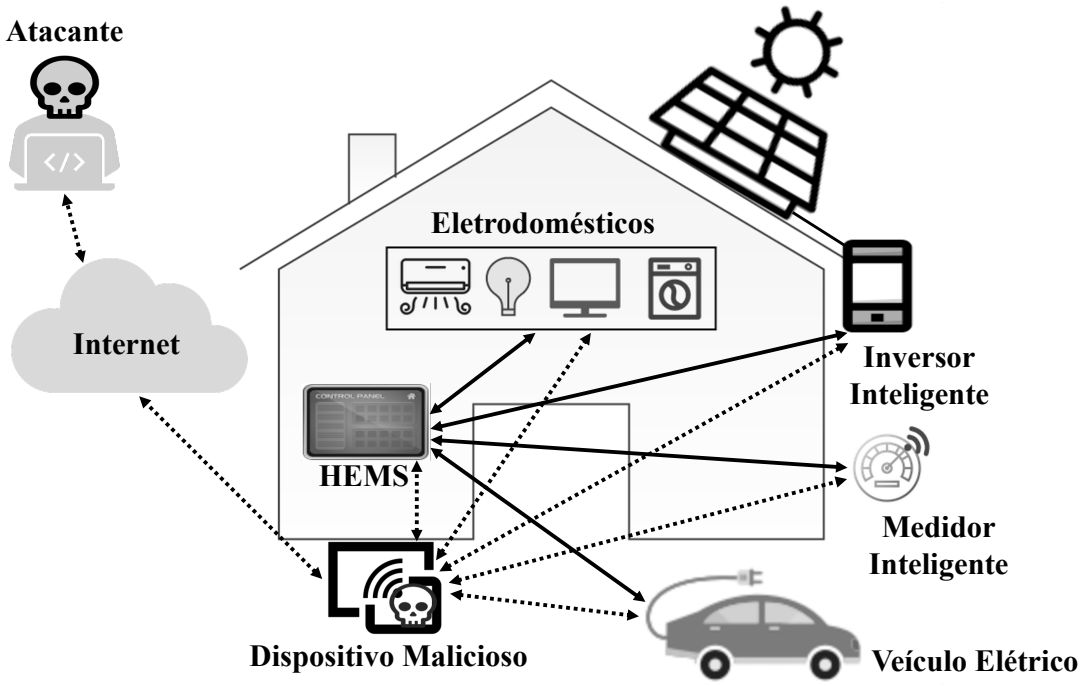


Figura 5.6: Dispositivo malicioso integrado a rede IEEE 2030.5

positivos e/ou pode fazer uso de recursos que são disponibilizados por outros dispositivos. Por apresentar um certificado digital IEEE 2030.5 assinado por uma MICA "legítima", os demais dispositivos irão confiar em tal certificado e consequentemente passarão a se comunicar com o dispositivo malicioso. Com isso, por exemplo, o dispositivo malicioso pode forjar valores de cobranças tarifárias excessivamente baixas dentro do recurso *TariffProfile* de forma a incentivar o aumento no consumo de energia elétrica ocasionando, assim, uma sobrecarga na rede elétrica. Ele também pode alterar parâmetros dentro do recurso *EndDeviceControl* de forma a instruir o acionamento de um inversor inteligente no momento que ele deveria estar desligado gerando, assim, uma falha no sistema.

A gravidade da situação aumenta com o comprometimento de um certificado digital de uma MCA, pois essa ocupa um nível mais alto na hierarquia. Caso ocorra o comprometimento do certificado digital de uma MCA, o atacante pode emitir certificados digitais para outras MICAs que serão consideradas legítimas pois são assinadas por uma MCA "legítima", no entanto, comprometida, e esta, por sua vez, emite certificados digitais para dispositivos.

Mesmo sendo descoberto o comprometimento do certificado digital, seja ele de uma MCA ou MICA, não há nenhuma solução prevista no padrão para que tanto eles quanto os certificados digitais que eles passem a emitir sejam declarados comprometidos e não mais confiáveis.

5.6.2 Comprometimento de certificado digital de dispositivo que hospeda recursos

O padrão IEEE 2030.5-2018 delega aos fabricantes a responsabilidade de proteção dos certificados digitais instalados nos dispositivos IEEE 2030.5. Ou seja, não há uma normatização de melhores práticas para a proteção e segurança dos certificados digitais. Sendo assim, cada fabricante pode instituir tecnologia própria ou de terceiros para assegurar tal proteção.

Um dispositivo que hospeda recursos torna-se o ponto central de comunicação de uma rede IEEE 2030.5, pois todos os demais dispositivos se comunicam com ele a fim de consumir seus recursos. Conforme visto anteriormente, ao dispositivo que consome os recursos (dispositivo solicitante) pode ou não ser atribuído certificado digital e, caso seja, pode ser até mesmo auto-assinado. Já dispositivo que hospeda recursos deve obrigatoriamente fazer uso de certificado digital, não sendo permitido o uso de certificado auto-assinado. Sendo assim, há uma maior preocupação em garantir a autenticidade do dispositivo que hospeda os recursos para que os dispositivos solicitantes saibam que estão se comunicando com uma fonte legítima.

No entanto, por falhas de projetos serem possíveis e por tecnologias tornarem-se defasadas frente à evolução tecnológica, o certificado digital de um dispositivo que hospeda recursos é passível de comprometimento. Mesmo descobertas em tempo as possíveis falhas que exponham vulnerabilidades relativas à segurança dos certificados digitais, a aplicação de correções a tais falhas não garante que elas não já tenham sido anteriormente exploradas e que os certificados digitais tenham sido comprometidos. Caso um atacante venha a ter a posse da chave privada associada ao certificado digital de um dispositivo que hospeda recursos, ele pode vir a causar sérios danos à rede.

Um atacante presente na rede IEEE 2030.5 e de posse da chave privada associada ao certificado digital de um dispositivo que hospeda recursos pode utilizar tal certificado digital para forjar ser um dispositivo legítimo e, com isso, falsificar informações e comandos. Por exemplo, ele pode forjar ser um medidor elétrico inteligente que, ao ser consultado por meio do recurso *MeterReading*, fornece valores das medições adulterados de forma a promover uma falsa sensação de baixo consumo o que pode induzir um possível aumento de carga levando a uma sobrecarga na rede. Ele também pode alterar parâmetros para enviar comandos a serem executados pelos dispositivos inteligentes levando a ocasionar falhas no sistema. Por exemplo, ele pode aceitar várias reservas de fluxo de energia por meio do recurso *FlowReservationRequest* alterando parâmetros de forma que todos ativem

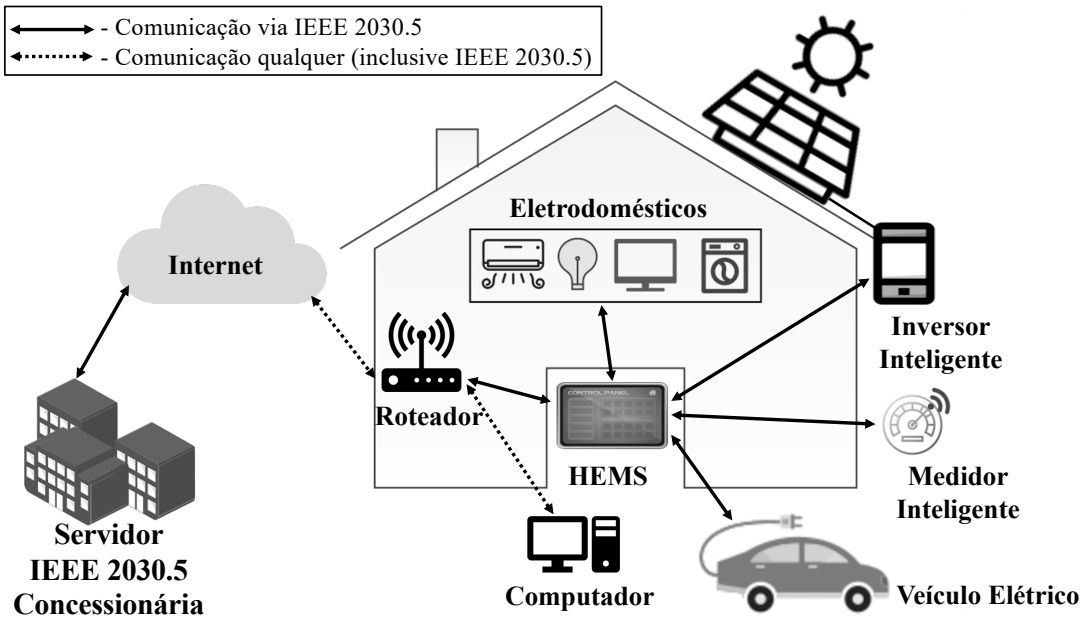


Figura 5.7: Rede residencial com IEEE 2030.5

o fluxo de energia ao mesmo tempo o que levaria a rede a ter um pico de sobrecarga.

Uma vez que o certificado digital utilizado pelo atacante é legítimo, todas informações e comandos por ele emitidos ao se passar por um dispositivo que fornece recursos também serão considerados legítimos e, a depender de como é configurado e interligado o sistema, as falha podem se propagar. No caso da falha relacionada com o reserva de fluxo, o atacante pode não repassar para a concessionária tal necessidade. Assim, com o início do consumo do fluxo, a sobrecarga por ele imposta pode alcançar a rede da concessionária desencadeando uma reação em cadeia.

A Figura 5.7 apresenta um exemplo de rede residencial. Nela, é possível notar dispositivos inteligentes (*e.g.*, medidores inteligentes, inversores inteligentes, veículos elétricos), um dispositivo que hospeda recursos (HEMS) e demais dispositivos que podem estar presentes em um ambiente residencial (*e.g.*, computadores, *notebooks*, *smartphones*, *tablets*). Todos são interligados por um roteador que pode fornecer tanto conexão cabeada quanto a sem fio. Os dispositivos inteligentes comunicam-se com o HEMS que, por sua vez, comunica-se com os servidores da concessionária enviando informações dos dispositivos inteligentes e também recebendo comandos e repassando aos mesmos. O proprietário da residência também pode acessar os dados e controlar os dispositivos inteligentes via interface direta do HEMS (*e.g.*, painel *touch screen*) ou através de um computador, *smartphone*, *tablet* ou qualquer outro meio que possibilite a comunicação com a interface do HEMS.

Um atacante partindo inicialmente de um ponto presente na rede residencial que tenha acesso aos demais dispositivos — como, por exemplo, de posse e controlando um computador (mesmo que remotamente) sem que o proprietário saiba — pode perpetrar ataques de forma a explorar falhas no HEMS a fim de comprometer o certificado digital. Por exemplo, caso a rede residencial tenha um computador comprometido, a partir dele o atacante pode iniciar uma exploração de vulnerabilidade do HEMS a fim de tomar posse da chave privada associada ao certificado digital nele presente. Obtendo sucesso na exploração de vulnerabilidade e de posse da chave privada associada ao certificado digital do HEMS, o atacante configura o computador que controla de forma a ser um provedor de recursos na rede fazendo uso das mesmas credenciais presentes no HEMS. Ao hospedar e anunciar recursos na rede fazendo uso do computador comprometido e do certificado digital do HEMS, o atacante pode realizar uma ataque *Denial of Service* (DoS) no HEMS de forma que os demais dispositivos solicitantes (consumidores de recursos) na rede não consigam mais se comunicar com ele. Ao encontrar os recursos anunciados pelo computador do qual o atacante tem posse, os dispositivos solicitantes iniciam o processo de autenticação. Como o computador possui o mesmo certificado digital do HEMS e os dispositivos solicitantes confiam nesse certificado digital, os dispositivos solicitantes realizam a autenticação e validação do certificado digital sem maiores problemas. A partir desse momento, o computador começa a se comunicar com os dispositivos solicitantes recebendo informações e enviando comandos para os mesmos conforme apresentado na Figura 5.8.

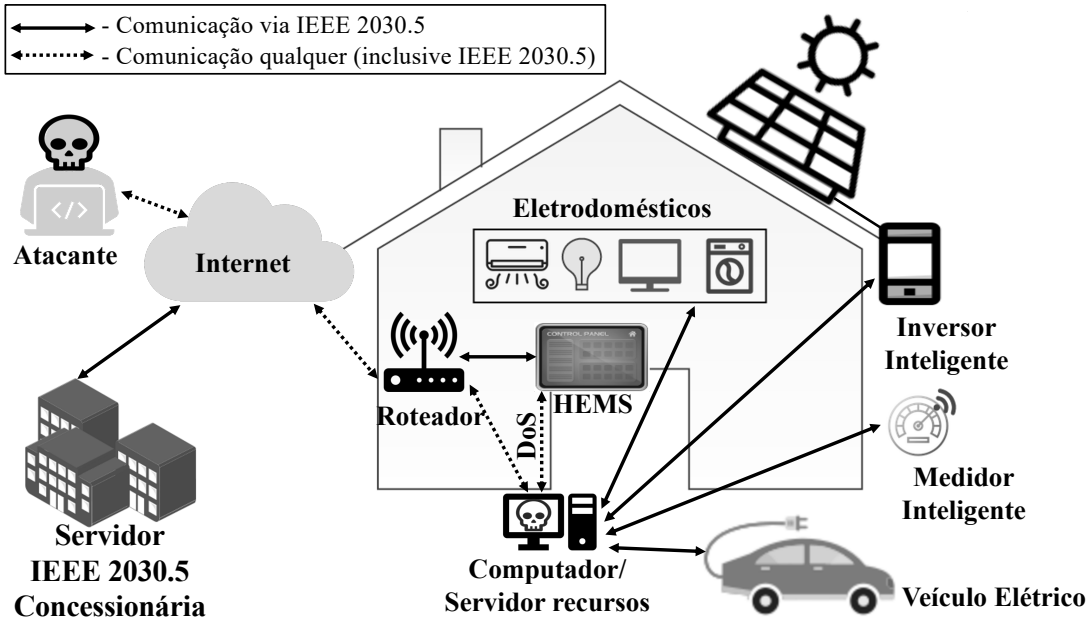


Figura 5.8: Rede residencial com dispositivo que hospeda recursos comprometido

A depender da política de segurança e da forma como é feito o processo de autenticação e comunicação do HEMS com os servidores da concessionária, o atacante pode configurar o computador para assumir de vez todas as funções do HEMS. Com isso, ele pode iniciar o processo de autenticação e comunicação com a concessionária para enviar informações falsas para a mesma. Por exemplo, caso uma concessionária venha a ser interligada à rede, ela pode utilizar o recurso *MeterReading* para consultar valores de medição do consumo de determinada *commodity* para fins de composição da fatura. No entanto, devido à adulteração desses valores feita pelo atacante, a concessionária poderá ter prejuízos financeiros.

Mesmo que seja identificado o comprometimento do certificado digital e seja aplicado algum método que impeça seu uso, esse impedimento será aplicado apenas de forma local. Por exemplo, caso uma concessionária aplique métodos para impedir o uso de determinado certificado digital, ao se extrapolar as fronteiras de controle dessa concessionária, o certificado digital torna-se novamente válido e pronto para o uso. A falta de uma verificação a ser feita em tempo de uso dos certificados digitais e de forma global impede que a informação de não legitimidade e não validade dos mesmos seja propagada para além das fronteiras de controle de determinado círculo.

5.6.3 Comprometimento de certificado digital de dispositivo que solicita recursos

O dispositivo que solicita recursos pode ter qualquer tipo de certificado digital. A depender da política de segurança implementada no sistema ao qual eles integram, pode haver tanto a coexistência dos diversos tipos de certificados digitais quanto apenas que determinado tipo seja permitido. Por exemplo, em um sistema que possua um dispositivo que hospede recursos e faça uso de certificado digital IEEE 2030.5, pode ser definido que apenas dispositivos solicitantes com certificado digital IEEE 2030.5 têm autorização para acessar os recursos. Sendo assim, os dispositivos solicitantes que façam uso dos demais tipos de certificados não irão ter acesso aos recursos.

Uma vez que ambos dispositivos (solicitante e hospedeiro de recursos) satisfaçam a política estabelecida para comunicação, o dispositivo solicitante verifica se há um registro (*EndDevice)* prévio com suas informações no dispositivo que hospeda os recursos. Caso não exista, é solicitada a criação de uma instância *EndDevice* no dispositivo que hospeda recursos. Com a aprovação da criação da instância, o dispositivo solicitante pode começar a fazer uso dos recursos ao qual tem autorização de acesso.

Um tipo de ataque que pode ocorrer a partir do comprometimento do certificado digital de um dispositivo que solicita recursos é relacionado ao envio de informações falsas. Um atacante pode se passar por um dispositivo que solicita recursos e enviar informações falsas ao dispositivo que hospeda os recursos. Esse, por sua vez, pode repassar tais informações falsas a depender de como está interligado todo o sistema. Com isso, decisões e comandos equivocados podem ser executados vindo a ocasionar falhas. Por exemplo, um atacante pode se passar por um medidor inteligente de gás e por meio do recurso *MirrorUsagePoint* enviar falsas medições para o dispositivo provedor de recursos ao qual ele está registrado. Com isso, a concessionária, ao consultar o dispositivo provedor de recursos, realizará a leitura dessas falsas medições o que ocasionará prejuízo financeiro para a mesma.

Por exemplo, a partir da rede residencial apresentada na Figura 5.7 e partindo do pressuposto de que o acesso ao HEMS por dispositivos solicitantes só pode ser feito com o uso de certificados digitais IEEE 2030.5, qualquer dispositivo solicitante que acesse a rede e não tenha certificado digital IEEE 2030.5 não poderá fazer uso dos recursos hospedados no HEMS. Caso um atacante esteja presente nessa rede — por exemplo, de posse e controlando o computador do proprietário da residência — o mesmo pode iniciar uma exploração de vulnerabilidade nos dispositivos solicitantes, como, por exemplo, em um inversor inteligente, visando comprometer e tomar posse da chave privada associada ao certificado digital IEEE 2030.5 presente nele. Levando-se em conta que falhas de projeto, sistema e até mesmo falhas humanas — engenharia social — podem ocorrer, há sempre a possibilidade de sucesso do atacante.

O atacante, de posse do certificado digital e explorando a natureza restrita de *hardware* dos dispositivos solicitantes, pode realizar um ataque DoS no inversor inteligente de forma a não permitir que ele se comunique com o HEMS. Logo em seguida, o atacante pode iniciar uma comunicação com o HEMS apresentando o certificado digital do inversor inteligente se passando por um falso inversor. O HEMS por entender que tal certificado digital é válido e também, por já ter uma instância *EndDevice* referente a tal certificado digital, o autentica e passa a conceder acesso aos recursos aos quais o certificado digital tem autorização. Ele também passa a receber e a registrar as informações falsas provenientes desse falso inversor. Com isso, decisões e comandos equivocados podem ser executados de forma a causar falha no sistema. Caso o HEMS esteja interligado aos servidores da concessionária, ele repassa tais informações falsas aos mesmos de forma a propagar possíveis falhas. A Figura 5.9 apresenta esse possível ataque no ambiente residencial.

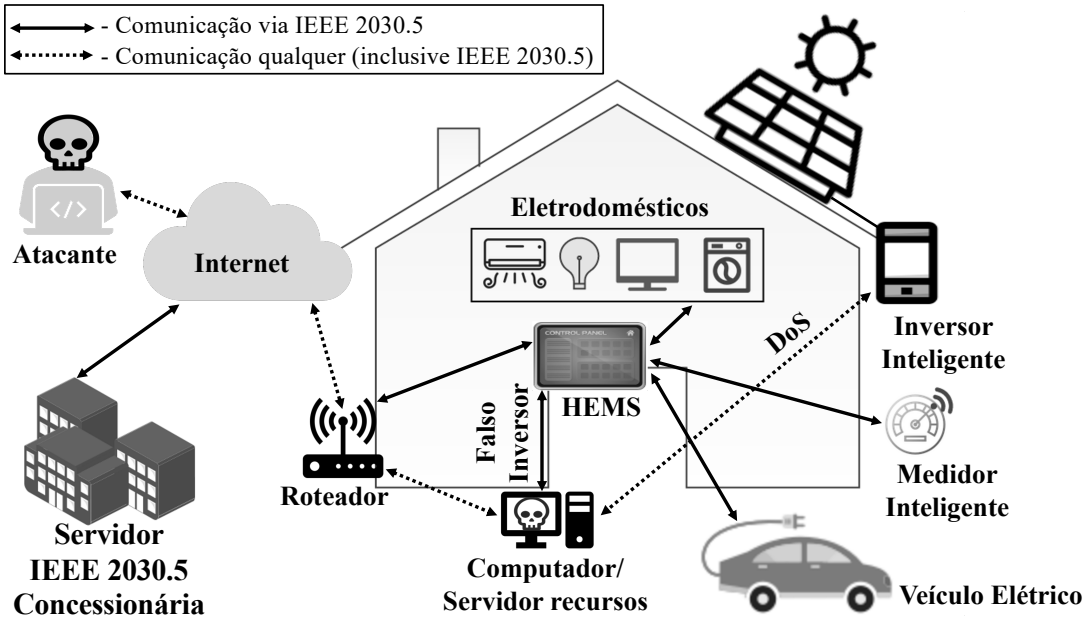


Figura 5.9: Rede residencial com dispositivo solicitante comprometido

Sendo assim, vistos esses três tipos de comprometimento de certificado digital IEEE 2030.5 e suas possíveis consequências para o sistema, nota-se claramente que não há meios e soluções previstos no padrão para sanar tal problema de forma definitiva.

Mesmo que seja descoberto o comprometimento, não há nenhuma forma prevista pelo padrão que consiga uma abrangência geral para a inutilização de tal certificado digital comprometido. Soluções tais como *black lists* podem ser implementadas, no entanto seu escopo se restringe ao ambiente local, ou seja, a solução não abrange áreas fora dos limites de tal ambiente. Além disso, a gravidade do problema aumenta ainda mais quando se trata de comprometimento de certificados digitais de CAs que compõem a hierarquia da *Manufacturing PKI*.

Diferentemente dos trabalhos de segurança apresentados no Capítulo 2, esse trabalho, além de promover uma análise de segurança abordando de maneira de direta e específica uma lacuna de segurança, também apresentou possíveis ataques que foram demonstrados de maneira prática juntamente com suas consequências. Com isso, a lacuna de segurança ganha uma melhor visualização, uma vez que ela é associada aos prejuízos e problemas do mundo real.

5.7 Prova de Conceito

Essa seção tem por finalidade apresentar, de forma prática, a lacuna de segurança presente na estrutura da *Manufacturing PKI* do IEEE 2030.5-2018. Para isso, inicialmente é apresentada a topologia de teste, juntamente com os cenários, e o pré-requisito necessário para a condução da prova de conceito.

5.7.1 Topologia de teste

A topologia de teste é baseada na comunicação entre um dispositivo cliente (que consome recursos) e um dispositivo servidor (que disponibiliza recursos) presentes em uma residência. Ambos são dotados de certificados digitais para realização de comunicação segura utilizando TLS. Além dos dispositivos cliente e servidor, em uma residência também há diversos outros dispositivos locais conectados à rede (*e.g.*, computador).

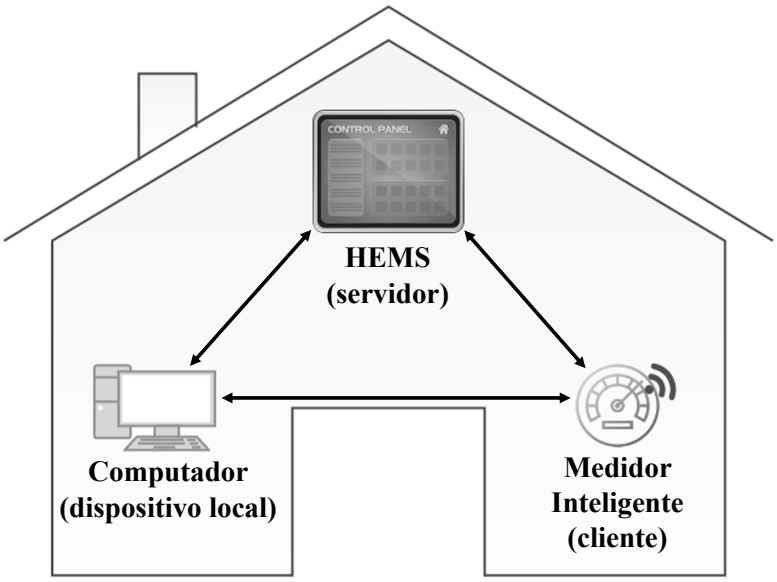


Figura 5.10: Cenário de teste

A Figura 5.10 apresenta a topologia de teste a ser utilizada na prova de conceito. Ela é composta de um HEMS representando o dispositivo servidor, um medidor inteligente representando o dispositivo cliente e um computador representando um dispositivo local na rede residencial. Todos os dispositivos estão interligados de forma que são capazes de comunicar-se uns com os outros.

A partir dessa topologia serão construídos 3 tipos de cenários:

- Cenário 1 – comunicação sem comprometimento de certificado digital;

- Cenário 2 – comprometimento do certificado digital do dispositivo servidor; e
- Cenário 3 – comprometimento do certificado digital do dispositivo cliente.

Por fim, será apresentado um cenário adicional que irá abordar o comprometimento de certificado digital envolvendo ambiente composto por DER.

5.7.2 Pré-requisito

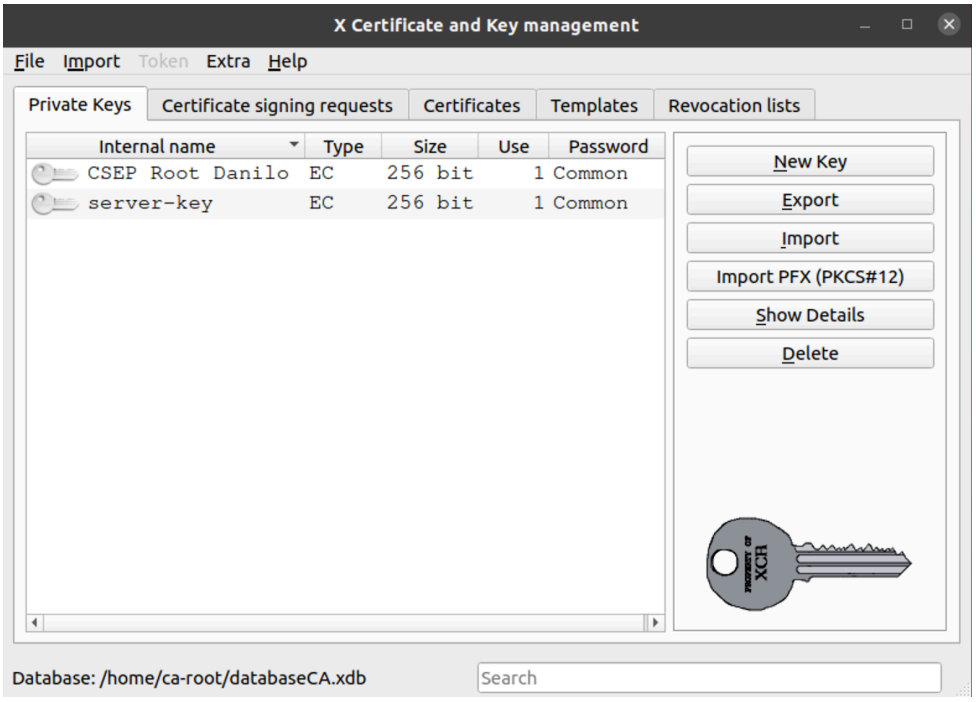
Na comunicação segura do IEEE 2030.5-2018, dispositivos clientes podem ser dotados de certificados digitais auto-assinados. Por outro lado, dispositivos servidores só podem fazer uso de certificados digitais provenientes de autoridade certificadora. Portanto, é exigida a emissão de um certificado digital advindo de uma autoridade certificadora para que o mesmo possa ser usado pelo dispositivo servidor. Para isso, é necessária a criação de uma autoridade certificadora.

A autoridade certificadora requerida foi concebida por meio de uma máquina virtual com sistema Linux onde foi instalado a aplicação "XCA". Tal aplicação é uma interface gráfica que permite a criação e o gerenciamento de certificados digitais. A partir da aplicação XCA, foi criada a chave privada de uma autoridade certificadora raiz denominada "CSEP_Root_Danilo" e também a chave privada "server-key".

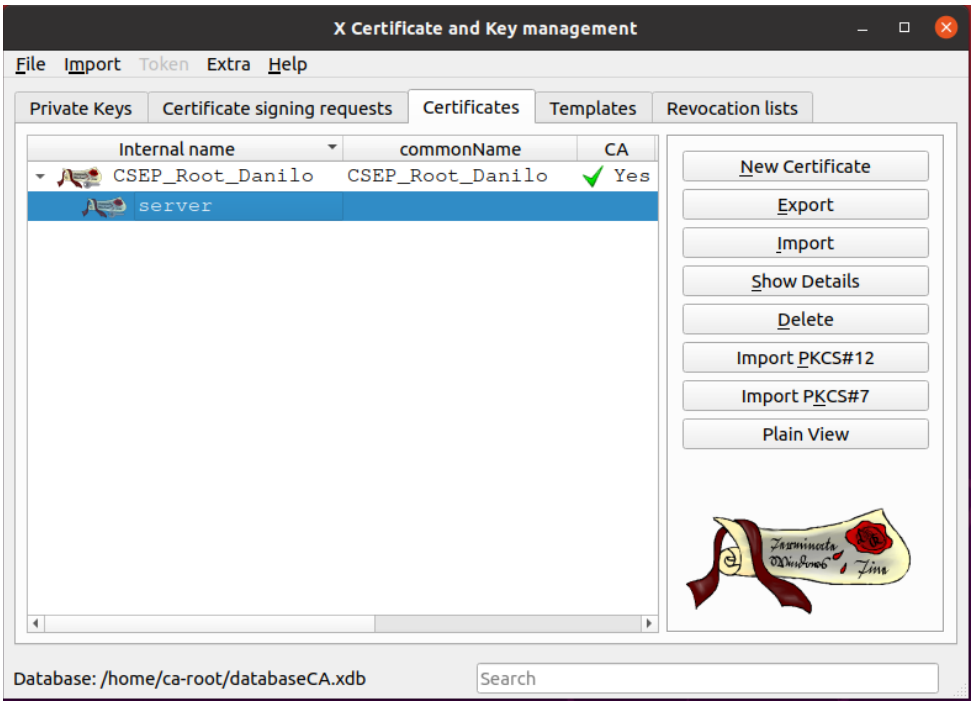
A autoridade certificadora, por ser uma autoridade raiz, possui certificado digital auto-assinado associado a sua chave privada. Já o certificado digital associado à chave privada "server-key" é assinado pela autoridade certificadora raiz "CSEP_Root_Danilo". O certificado "server" será utilizado no dispositivo servidor. A Figura 5.11-(A) apresenta a interface gráfica do XCA juntamente com as chaves privadas relacionadas à autoridade certificadora raiz e ao certificado digital "server". Já a Figura 5.11-(B) apresenta o certificado digital emitido para ambas as chaves privadas.

5.7.3 Dispositivos da topologia de teste

Cada dispositivo que compõe a topologia de teste é representado por uma máquina virtual com sistema operacional Linux. Cada um possui sua especificidade em relação ao seu papel no sistema. Por exemplo, o HEMS deve conter recursos e disponibilizá-los por meio de uma interface RESTful para serem consumidos pelo medidor inteligente. Já o medidor inteligente deve fazer requisições de modo a consumir os recursos disponibilizado pelo HEMS.



(A) Chaves privadas



(B) Certificados digitais

Figura 5.11: Criação de autoridade certificadora raiz e certificado digital do servidor usando XCA

5.7.3.1 Medidor inteligente

O medidor inteligente é representado por uma máquina virtual Linux de nome "`client`". Com relação a função a ser desempenhada, em uma pesquisa realizada, foi encontrada uma implementação referente ao dispositivo cliente IEEE 2030.5 feita pela *Eletric Power Research Institute* (EPRI) [20]. O código da implementação está disponibilizado no site GitHub [64]. No entanto, ao ser compilado, o código apresenta um erro referente à versão do OpenSSL. Um *fork* dessa implementação, também presente no site GitHub (catch-twenty-two/ieee-2030.5-client) [36], contém a alteração necessária para permitir o uso do OpenSSL na versão mais recente. Esse último código, ao ser compilado, não apresentou problemas e foi utilizado para representar o cliente.

Contudo, nem todas as funcionalidades estão implementadas para teste. Funcionalidades básicas tais como registro de dispositivo, função *time*, *meter* e *metering mirror* estão implementadas. Na própria página do repositório no GitHub, é disponibilizado um guia para a compilação e execução do código. Dentre os arquivos presentes no repositório, estão a chave privada (pti_dev.pem) e o certificado digital (pti_dev.x509) a serem utilizados pelo cliente na comunicação segura com o dispositivo servidor. Na pasta "certs", encontra-se o certificado digital "csep_root.pem" pertencente à autoridade certificadora raiz que emitiu o certificado "pti_dev.x509". Essa pasta armazena os certificados digitais das autoridades certificadoras nas quais o dispositivo cliente confia.

Tendo em vista que a comunicação do cliente com o servidor envolve requisições e respostas a uma interface RESTful, a ferramenta CURL — presente no Linux — também pode ser usada para simular a funcionalidade de um cliente. Uma vez que o CURL também permite a utilização de certificados digitais, ele pode ser utilizado para realizar a comunicação segura enviando requisições com diferentes métodos HTTP e recebendo respostas a essas requisições.

5.7.3.2 HEMS

O HEMS é representado por uma máquina virtual Linux — de nome "`server`" — que utiliza o Python 3.8.5 e o *micro-framework* Flask [57]. Escrito em Python, o *micro-framework* Flask permite a criação de APIs RESTful de modo simples e ágil.

O Flask facilita o trabalho por utilizar rotas e métodos HTTP a fim de executar uma função. Por exemplo, para configurar uma função "`ola()`" que retorna uma mensagem, é necessário que o Flask reconheça a função que irá retornar a mensagem e também

```
1    @app.route("/inicio", methods=["GET"])
2    def ola():
3        return "<h1>Ola Flask!</h1>"
```

Figura 5.12: Exemplo de uso do `@app.route()`

deve saber que ela deverá ser executada quando determinada rota e método HTTP for requisitado. Para isso, é utilizada a diretiva "`@app.route()`". Tal diretiva contém a função a ser executada, o método HTTP e a rota a qual, ao ser requisitada, executa a função.

O trecho de código apresentado na Figura 5.12 demonstra a utilização da diretiva. No código, caso a rota "`/inicio`" seja requisitada, juntamente com o método "`GET`", a função "`ola()`" será executada retornando a mensagem. O Flask também permite a configuração de certificado digital para comunicação segura e a seleção da porta na qual o serviço será executado.

Sendo assim, utilizando o Flask, foi implementado o protótipo de um HEMS (servidor de recursos). O protótipo, além de fazer uso da diretiva `@app.route()`, implementa a funcionalidade de autenticação mútua na qual o certificado digital apresentado pelo dispositivo cliente é verificado de forma que somente seja permitida a comunicação caso ele seja válido. Para isso, é criado um contexto SSL no qual são passados a chave privada e o certificado digital usados pelo dispositivo servidor e também o certificado digital da autoridade certificadora que irá validar o certificado apresentado pelo dispositivo cliente.

Nem todos os recursos foram implementados no HEMS. Recursos mais básicos, tais como *EndDevice*, *Register*, *TariffProfile* e *Time*, foram implementados de maneira simplificada. Isso foi feito, pois o objetivo é demonstrar a autenticação e a comunicação segura entre cliente e servidor usando o TLS para consumir os recursos.

5.7.3.3 Computador local

O computador local presente na residencia é representado por uma máquina virtual Linux — de nome "`server-devil`" — onde encontra-se instalado o Flask e a ferramenta HPING3 [58]. Tal ferramenta pode ser utilizada na execução de um tipo de ataque DoS denominado *TCP SYN Flooding Attack* [19].

Basicamente, o ataque *TCP SYN Flooding Attack* consiste no envio de inúmeros pacotes TCP solicitando conexão ao servidor com a flag SYN ativa sem fechar as conexões anteriores. Isso sobrecarrega o servidor fazendo com que ele pare de responder as demais

solicitações.

Ataques que envolvam a utilização e alocação excessiva de *hardware* possuem uma eficiência elevada quando realizado sobre dispositivos que possuem restrições de *hardware* — que é o caso da maioria dos dispositivos IoT. Sendo assim, a alocação excessiva de *hardware* faz com que o dispositivo não consiga responder as solicitações.

Vale ressaltar que já existem meios e contramedidas para se evitar o ataque *TCP SYN Flooding*. A utilização do mesmo nessa prova de conceito possui mero caráter exemplificativo de uma forma de ataque. Na prática, um atacante poderia usar qualquer outro meio de ataque que fosse viável a fim de causar uma negação de serviço.

5.7.4 Cenário 1 – sem comprometimento de certificado digital

Inicialmente a comunicação entre o medidor inteligente e o HEMS ocorre para verificar quais conjuntos de funções são disponibilizados pelo HEMS. Isso é feito consultando o recurso *DeviceCapability* (dcap) do HEMS. Para que isso ocorra, é necessário que haja previamente uma autenticação mútua entre eles usando o TLS.

Vale lembrar que alguns recursos não necessitam de comunicação segura, no entanto, para melhor entendimento, toda comunicação entre dispositivos será considerada segura e faz uso do TLS.

A Figura 5.13 apresenta essa comunicação entre o medidor inteligente (parte superior da Figura) e o HEMS (parte inferior da Figura). Nela, é possível verificar que o HEMS possui a interface RESTful ativa na porta 8443 acessada por meio do HTTPS. O medidor inteligente faz uma consulta ao recurso "dcap" utilizando a ferramenta CURL. Em um primeiro momento, ocorre a autenticação mútua de ambos por meio do TLS. Havendo sucesso na autenticação, o HEMS envia ao medidor inteligente o recurso solicitado. Caso a autenticação mútua não seja bem sucedida, a comunicação é interrompida. Lembrando que, conforme previsto no padrão, a verificação mútua de ambos certificados provenientes de uma *Manufacturing PKI* é feita por meio da verificação da cadeia de assinaturas até a autoridade certificadora raiz.

Em seguida, ao verificar que o HEMS possui o conjunto de funções *EndDeviceList*, o medidor inteligente verifica se ele está registrado no HEMS. Nessa verificação é utilizada a implementação presente no GitHub. A utilização dessa implementação também serviu para comprovar que o HEMS (servidor de recursos) está em conformidade com o previsto no padrão. Para que o medidor inteligente confie no certificado digital do

```
danilo — client@client: ~/IEEE-2030.5-Client — ssh -R 52698:localhost:52698 client@10.211.55.7 — 94×59
client@client:~/IEEE-2030.5-Client$ curl -v --cert pti_dev.crt --key pti_dev.pem --cacert ../C
SEP_Root_Danilo.crt -k -H "Accept: application/sep+xml; level=-S1" https://10.211.55.6:8443/dc
ap
*   Trying 10.211.55.6:8443...
* TCP_NODELAY set
* Connected to 10.211.55.6 (10.211.55.6) port 8443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: ../CSEP_Root_Danilo.crt
  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Request CERT (13):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Certificate (11):
* TLSv1.3 (OUT), TLS handshake, CERT verify (15):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: [NONE]
*  start date: Feb 11 17:01:00 2021 GMT
*  expire date: Dec 31 23:59:59 9999 GMT
*  issuer: O=CSEP; CN=CSEP_Root_Danilo; serialNumber=1
*  SSL certificate verify ok.
> GET /dcap HTTP/1.1
> Host: 10.211.55.6:8443
> User-Agent: curl/7.68.0
> Accept: application/sep+xml; level=-S1
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: application/sep+xml
< Content-Length: 510
< Server: Werkzeug/1.0.1 Python/3.8.5
< Date: Thu, 25 Feb 2021 19:00:35 GMT
<
<DeviceCapability xmlns="http://ieee.org/2030.5" href="/dcap">
        <DemandResponseProgramListLink all="2" href="/dr"/>
        <DERProgramListLink all="2" href="/derp"/>
        <MessagingProgramListLink all="2" href="/msg"/>
        <ResponseSetListLink all="2" href="/rsps"/>
        <TariffProfileListLink all="1" href="/tp"/>
        <TimeLink href="/tm"/>
        <UsagePointListLink all="1" href="/upt"/>
        <EndDeviceListLink all="1" href="/edev"/>
        <MirrorUsagePointListLink all="0" href="/mup"/>
        <SelfDeviceLink href="/sdev"/>
</DeviceCapability>
* Connection #0 to host 10.211.55.6 left intact
client@client:~/IEEE-2030.5-Client$
```

```
danilo — server@server: ~/server — ssh -R 52698:localhost:52698 server@10.211.55.6 — 94×12
server@server:~/server$ python3 resources.py
 * Serving Flask app "resources" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on https://0.0.0.0:8443/ (Press CTRL+C to quit)
10.211.55.7 - - [25/Feb/2021 19:00:35] "GET /dcap HTTP/1.1" 200 -
```

Figura 5.13: Comunicação para verificação do recurso *DeviceCapability*

HEMS, foi adicionado na pasta “certs” o certificado digital da autoridade certificadora “CSEP_Root_Danilo.pem”.

Previamente, foram inseridos no HEMS o SFDI e o PIN referentes ao medidor inteligente. Vale lembrar que o uso do PIN para verificação do registro é opcional. No caso, a implementação faz uso do PIN e o mesmo possui o valor “1111115”. Portanto ao percorrer o fluxo para verificação do registro, caso haja correspondência entre as informações, o medidor inteligente confirma que está registrado no HEMS. Para a execução dessa verificação de registro, o medidor inteligente executa o seguinte comando:

```
./build/client_test enp0s2 pti_dev.x509 certs https://10.211.55.6:8443/dcap register
```

Onde:

- ./build/client_test – representa o comando para execução das funcionalidades.
- enp0s2 – representa a interface de rede.
- pti_dev.x509 – representa o certificado digital do medidor inteligente.
- certs – representa o diretório onde se encontram os certificados as autoridades certificadoras confiáveis.
- https://10.211.55.6:8443/dcap – representa o endereço do HEMS para acesso ao recurso *DeviceCapability*.
- register – representa a funcionalidade a ser executada pelo medidor inteligente conforme previsto no manual da implementação.

A Figura 5.14 apresenta o medidor inteligente (lado esquerdo da Figura) e o HEMS (lado direito da Figura) trocando informações para verificar se o medidor inteligente está registrado no HEMS.

Inicialmente é calculado o LFDI e o SFDI do medidor inteligente. Em seguida, os certificados digitais das autoridades certificadoras confiáveis são carregados e é realizada a autenticação mútua dos dispositivos. Em caso de falha, a comunicação é interrompida. Caso contrário, é iniciada a troca de mensagens entre os dispositivos. Em vermelho estão destacadas algumas informações importantes que são verificadas a fim de confirmar o registro do medidor inteligente.

Em se tratando de um medidor inteligente de energia elétrica, em determinado, momento ele consulta o HEMS para saber o preço da energia. Esse preço pode ser apresen-

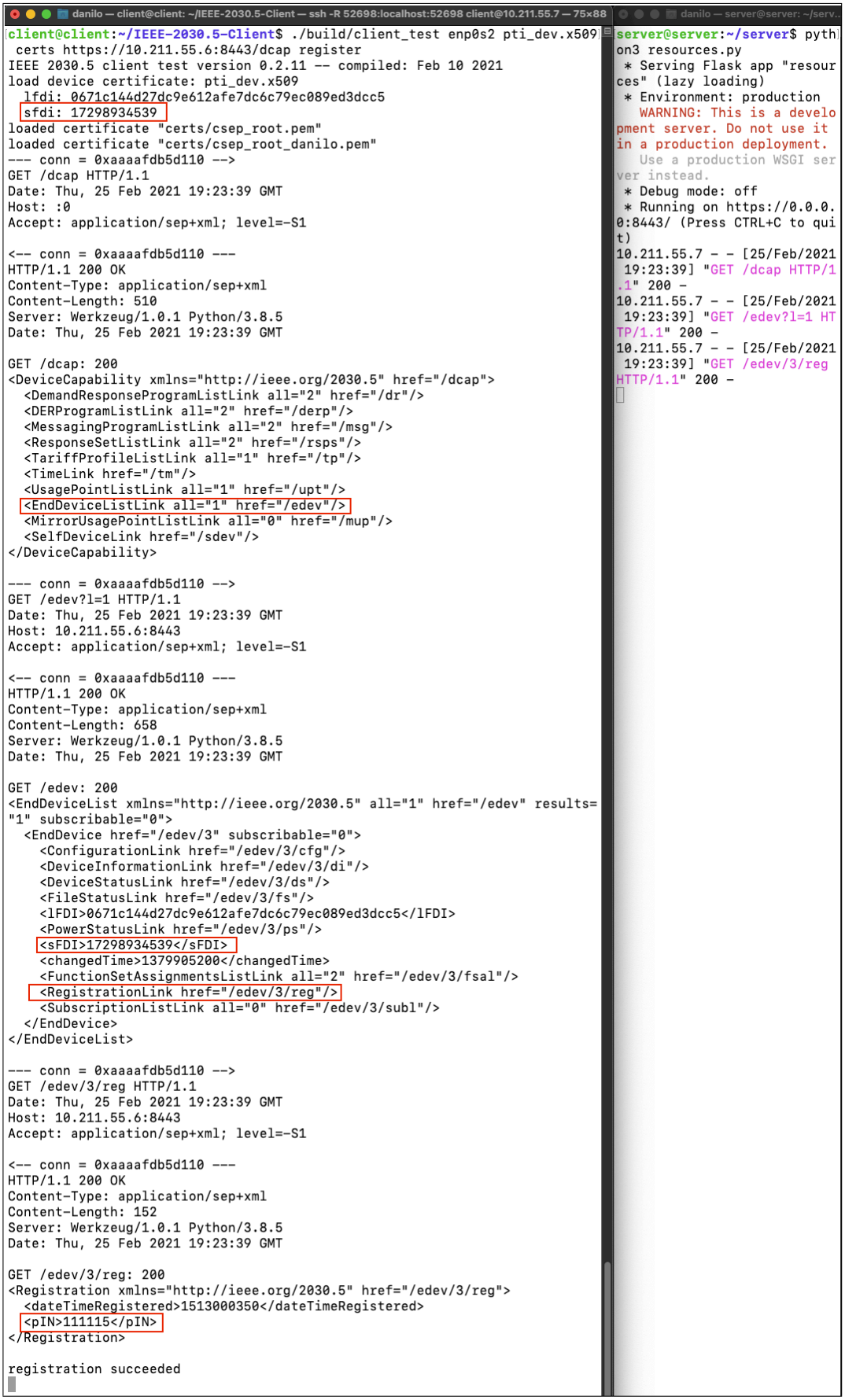


Figura 5.14: Comunicação para verificação de registro do medidor inteligente

tado em um painel do medidor inteligente de forma que o proprietário da residência tenha ciência do valor que está sendo cobrado pela energia naquele determinado momento.

O preço da energia pode variar a depender do horário do dia. Em horário de pico de consumo, o preço é mais elevado a fim de inibir o consumo excessivo e desnecessário os quais podem sobrecarregar o sistema elétrico o levando a falhas. A informação do preço da energia ajuda o proprietário da residencia a controlar o consumo e o valor final da fatura. O fluxo da troca de mensagens para verificação do preço da energia previsto no padrão é apresentado na Figura 5.15.

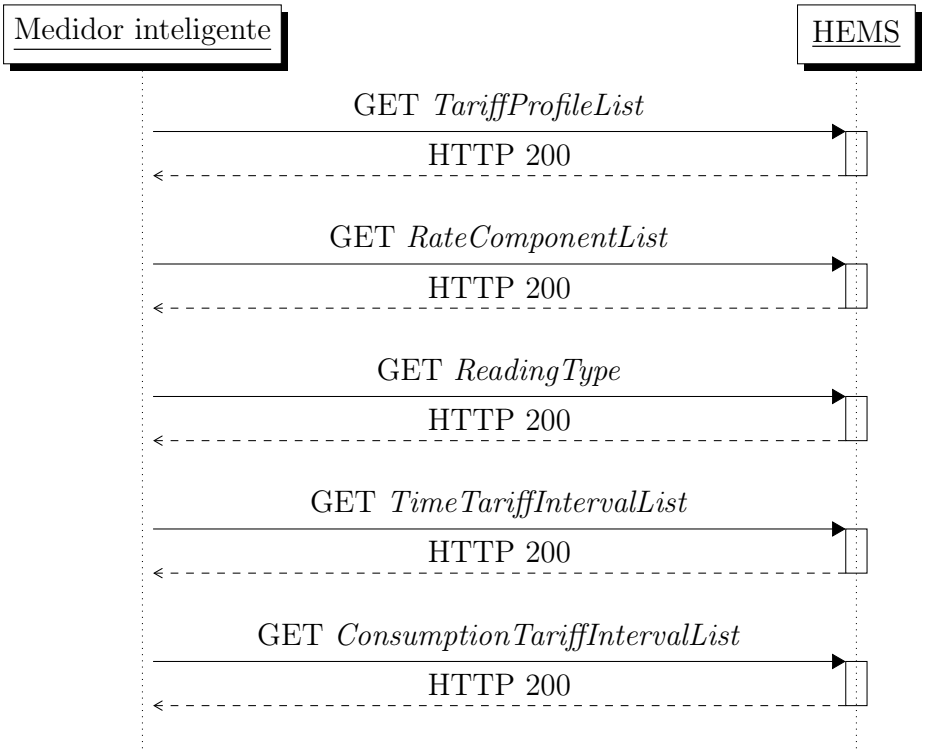


Figura 5.15: Fluxo de mensagens cliente-servidor para obter preço da energia

As Figuras 5.16 e 5.17 apresentam as trocas de mensagens entre o medidor inteligente e o HEMS. Antes de inciar a troca de mensagens, é feita a autenticação mútua. Sendo assim, obtendo sucesso na autenticação, a troca de mensagens é iniciada e os dispositivos consideram que as informações trocadas são verdadeiras, não ocorrendo dúvida sobre a legitimidade das mesmas.

O HEMS também pode armazenar informações de medições de medidores inteligentes de outras *commodities* (*e.g.*, gás). O medidor inteligente, além de consultar informações no HEMS, também envia seus dados de medição para que sejam armazenados no mesmo. Para isso, o HEMS deve disponibilizar o conjunto de funções *Metering Mirror*. Tais medições são posteriormente acessadas pela concessionária a fim de compor a fatura de

```
danilo — client@client: ~/IEEE-2030.5-Client — ssh -R 52698:localhost:52698 client@10.211.55.7 — 105×93
client@client:~/IEEE-2030.5-Client$ curl -v --cert pti_dev.crt --key pti_dev.pem --cacert ../CSEP_Root_Da
nilo.crt -k -H "Accept: application/sep+xml; level=-S1" https://10.211.55.6:8443/{tp,'tp/1/rc?l=1',rt/1,'
tp/1/rc/1/acttti?l=1','tp/1/rc/1/tti/0/cti?l=1'}
*   Trying 10.211.55.6:8443...
* TCP_NODELAY set
* Connected to 10.211.55.6 (10.211.55.6) port 8443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: ../CSEP_Root_Danilo.crt
  CApath: /etc/ssl/certs
* TLSv1.3 (OUT), TLS handshake, Client hello (1):
* TLSv1.3 (IN), TLS handshake, Server hello (2):
* TLSv1.3 (IN), TLS handshake, Encrypted Extensions (8):
* TLSv1.3 (IN), TLS handshake, Request CERT (13):
* TLSv1.3 (IN), TLS handshake, Certificate (11):
* TLSv1.3 (IN), TLS handshake, CERT verify (15):
* TLSv1.3 (IN), TLS handshake, Finished (20):
* TLSv1.3 (OUT), TLS change cipher, Change cipher spec (1):
* TLSv1.3 (OUT), TLS handshake, Certificate (11):
* TLSv1.3 (OUT), TLS handshake, CERT verify (15):
* TLSv1.3 (OUT), TLS handshake, Finished (20):
* SSL connection using TLSv1.3 / TLS_AES_256_GCM_SHA384
* ALPN, server did not agree to a protocol
* Server certificate:
*  subject: [NONE]
*  start date: Feb 11 17:01:00 2021 GMT
*  expire date: Dec 31 23:59:59 9999 GMT
*  issuer: O=CSEP; CN=CSEP_Root_Danilo; serialNumber=1
*  SSL certificate verify ok.
> GET /tp HTTP/1.1
> Host: 10.211.55.6:8443
> User-Agent: curl/7.68.0
> Accept: application/sep+xml; level=-S1
>
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* TLSv1.3 (IN), TLS handshake, Newsession Ticket (4):
* old SSL session ID is stale, removing
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: application/sep+xml
< Content-Length: 469
< Server: Werkzeug/1.0.1 Python/3.8.5
< Date: Thu, 25 Feb 2021 19:33:45 GMT
<
<TariffProfileList all="1" href="/tp" results="1" xmlns="urn:ieee:std:2030.5:ns">
        <TariffProfile href="/tp/1">
                <mRID>799794f4620b17e00000e566</mRID>
                <description>Standard Rate</description>
                <currency>840</currency>
                <pricePowerOfTenMultiplier>-6</pricePowerOfTenMultiplier>
                <primacy>0</primacy>
                <rateCode>FLAT</rateCode>
                <RateComponentListLink all="1" href="/tp/1/rc"/>
                <serviceCategoryKind>0</serviceCategoryKind>
        </TariffProfile>
* Connection #0 to host 10.211.55.6 left intact
</TariffProfileList>* Found bundle for host 10.211.55.6: 0xaaaafa33b9e0 [serially]
* Can not multiplex, even if we wanted to!
* Re-using existing connection! (#0) with host 10.211.55.6
* Connected to 10.211.55.6 (10.211.55.6) port 8443 (#0)
> GET /tp/1/rc?l=1 HTTP/1.1
> Host: 10.211.55.6:8443
> User-Agent: curl/7.68.0
> Accept: application/sep+xml; level=-S1
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: application/sep+xml
< Content-Length: 489
< Server: Werkzeug/1.0.1 Python/3.8.5
< Date: Thu, 25 Feb 2021 19:33:45 GMT
<
<RateComponentList:sep all="1" href="/tp/1/rc" results="1" xmlns="urn:ieee:std:2030.5:ns">
        <RateComponent:sep href="/tp/1/rc/1">
                <mRID:sep>fc000b07143d24fc0000e566</mRID:sep>
                <description:sep>Standard Rate</description:sep>
                <ActiveTimeTariffIntervalListLink:sep all="1" href="/tp/1/rc/1/acttti"/>
                <ReadingTypeLink:sep href="/rt/1"/>
                <roleFlags:sep>1</roleFlags:sep>
                <TimeTariffIntervalListLink:sep all="5" href="/tp/1/rc/3/tti"/>
        </RateComponent:sep>
* Connection #0 to host 10.211.55.6 left intact
</RateComponentList:sep>* Found bundle for host 10.211.55.6: 0xaaaafa33b9e0 [serially]
```

Figura 5.16: Consulta preço energia - parte 1

```
* Re-using existing connection! (#0) with host 10.211.55.6
* Connected to 10.211.55.6 (10.211.55.6) port 8443 (#0)
> GET /rt/1 HTTP/1.1
> Host: 10.211.55.6:8443
> User-Agent: curl/7.68.0
> Accept: application/sep+xml; level=-S1
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: application/sep+xml
< Content-Length: 411
< Server: Werkzeug/1.0.1 Python/3.8.5
< Date: Thu, 25 Feb 2021 19:33:45 GMT
<
<ReadingType href="/rt/1" xmlns="urn:ieee:std:2030.5:ns">
        <accumulationBehavior>9</accumulationBehavior>
        <commodity>1</commodity>
        <dataQualifier>12</dataQualifier>
        <flowDirection>1</flowDirection>
        <kind>37</kind>
        <numberOfConsumptionBlocks>1</numberOfConsumptionBlocks>
        <numberOfTouTiers>1</numberOfTouTiers>
        <phase>0</phase>
        <powerOfTenMultiplier>3</powerOfTenMultiplier>
        <uom>72</uom>
* Connection #0 to host 10.211.55.6 left intact
</ReadingType>* Found bundle for host 10.211.55.6: 0xaaaafa33b9e0 [serially]
* Can not multiplex, even if we wanted to!
* Re-using existing connection! (#0) with host 10.211.55.6
* Connected to 10.211.55.6 (10.211.55.6) port 8443 (#0)
> GET /tp/1/rc/1/acttti?l=1 HTTP/1.1
> Host: 10.211.55.6:8443
> User-Agent: curl/7.68.0
> Accept: application/sep+xml; level=-S1
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: application/sep+xml
< Content-Length: 679
< Server: Werkzeug/1.0.1 Python/3.8.5
< Date: Thu, 25 Feb 2021 19:33:45 GMT
<
<TimeTariffIntervalList all="1" href="/tp/1/rc/1/acttti" results="1" subscribable="1" xmlns="urn:ieee:std
:2030.5:ns">
        <TimeTariffInterval href="/tp/1/rc/1/tti/0">
                <mRID>ef06fa23dc0a0f650000e566</mRID>
                <description>Standard Rate</description>
                <creationTime>1357430400</creationTime>
                <EventStatus>
                        <currentStatus>1</currentStatus>
                        <dateTime>1357516800</dateTime>
                        <potentiallySuperseded>false</potentiallySuperseded>
                </EventStatus>
                <randomizeDuration>300</randomizeDuration>
                <randomizeStart>300</randomizeStart>
                <ConsumptionTariffIntervalListLink all="1" href="/tp/1/rc/1/tti/0/cti"/>
                <touTier>1</touTier>
        </TimeTariffInterval>
* Connection #0 to host 10.211.55.6 left intact
</TimeTariffIntervalList>* Found bundle for host 10.211.55.6: 0xaaaafa33b9e0 [serially]
* Can not multiplex, even if we wanted to!
* Re-using existing connection! (#0) with host 10.211.55.6
* Connected to 10.211.55.6 (10.211.55.6) port 8443 (#0)
> GET /tp/1/rc/1/tti/0/cti?l=1 HTTP/1.1
> Host: 10.211.55.6:8443
> User-Agent: curl/7.68.0
> Accept: application/sep+xml; level=-S1
>
* Mark bundle as not supporting multiuse
< HTTP/1.1 200 OK
< Content-Type: application/sep+xml
< Content-Length: 327
< Server: Werkzeug/1.0.1 Python/3.8.5
< Date: Thu, 25 Feb 2021 19:33:45 GMT
<
<ConsumptionTariffIntervalList all="1" href="/tp/1/rc/1/tti/0/cti" results="1" xmlns="urn:ieee:std:2030.5
:ns">
        <ConsumptionTariffInterval href="/tp/1/rc/1/tti/0/cti/0">
                <consumptionBlock>1</consumptionBlock>
                <price>113000</price>
                <startValue>0</startValue>
        </ConsumptionTariffInterval>
</ConsumptionTariffIntervalList>
* Connection #0 to host 10.211.55.6 left intact
client@client:~/IEEE-2030.5-Client$
```

Figura 5.17: Consulta preço energia - parte 2

cobrança.

Por exemplo, caso o medidor inteligente seja um medidor de gás, ele pode armazenar seus dados de medição no HEMS e posteriormente a concessionária consultaria o HEMS a fim de recuperar os dados de medição para gerar a fatura a ser paga.

O medidor inteligente cria um *MirrorUsagePoint* no HEMS. Isso é feito utilizando o método POST. Após a criação, o medidor inteligente envia seus dados de medição via POST para o recurso *MirrorMeterReading* para que essa informação fique registrada no HEMS. As Figuras 5.18 e 5.19 apresentam esse processo.

A Figura 5.18 inicialmente mostra a informação a ser enviada para a criação do *MirrorUsagePoint* no HEMS. Logo em seguida, a informação é enviada via POST ao HEMS que retorna a URI para acesso. A Figura 5.19 apresenta o valor da medição a ser enviada ao HEMS. Em seguida, o medidor inteligente envia a informação da medição e recebe como resposta a URI de acesso a mesma. Por fim, um GET é feito na URI para verificar o valor registrado no HEMS.

Sendo assim, nesse cenário foi apresentado, em um primeiro momento, o envio de informação proveniente do HEMS ao medidor inteligente e posteriormente o envio de informação do medidor inteligente ao HEMS. No primeiro caso, foi enviado o valor do preço da energia ao medidor inteligente e, no segundo caso, foi enviado o valor da medição ao HEMS.

As informações trocadas entre os dispositivos são consideradas verdadeiras e legítimas a partir do momento que se obtém sucesso na autenticação mútua dos dispositivos. Tal autenticação envolve os certificados digitais. No entanto, havendo o comprometimento de um certificado, não há previsão de um meio pelo qual se invalide tal certificado comprometido e impeça o seu uso em outro dispositivo em qualquer outro lugar.

Nos cenários seguintes serão apresentados modos de adulteração de tais informações partindo-se do comprometimento dos certificados digitais dos dispositivos.

5.7.5 Cenário 2 – comprometimento do certificado digital do dispositivo servidor

Nesse cenário, parte-se do pressuposto que um atacante — de posse do computador local — tenha comprometido o certificado digital do HEMS. Com isso, ele tem a posse da chave privada referente ao mesmo.

Figura 5.18: Envio de valores de medições - parte 1

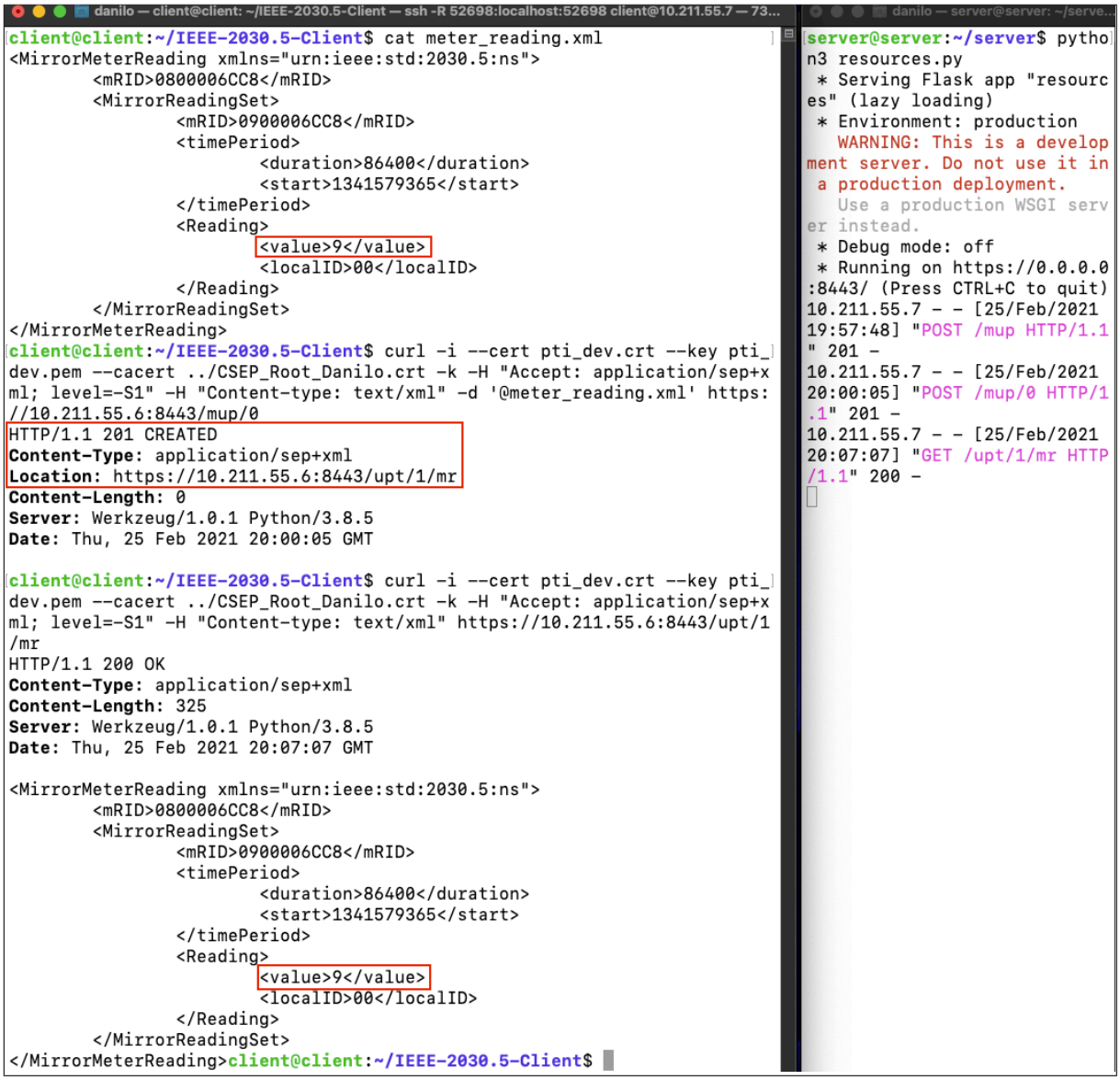


Figura 5.19: Envio de valores de medições - parte 2

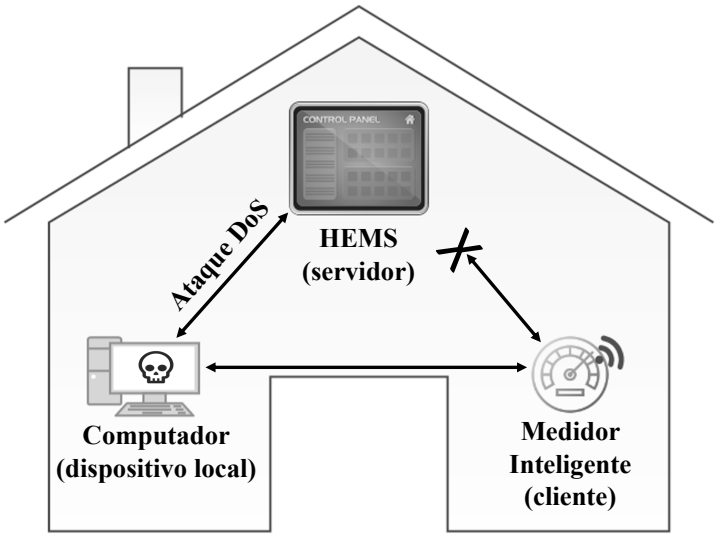


Figura 5.20: Cenário ataque ao HEMS

A partir de então, ele pode se passar pelo HEMS ao apresentar o certificado digital comprometido de forma que o medidor inteligente inicie com ele uma comunicação sem maiores problemas. Assim, ele pode fornecer um valor falso do preço da energia (valor mais baixo), sendo que, naquele momento, o valor encontra-se elevado devido ao pico de demanda.

Para que tal alteração de preço ocorra, o atacante inicia um ataque DoS contra o HEMS usando a ferramenta HPING3. A partir de então, o medidor inteligente não consegue mais se comunicar com o HEMS para verificação do preço da energia.

Esse cenário onde o HEMS sofre um ataque DoS e o medidor inteligente não consegue se comunicar com ele é apresentado na Figura 5.20. Já a demonstração prática desse ataque é apresentada na Figura 5.21. Nela, é possível notar que o computador local (server-devil) realiza o ataque usando o HPING3 contra o HEMS (server) fazendo com que ele pare de responder as solicitações. Com consequência disso, o medidor inteligente (client) não consegue se comunicar com o HEMS para obter o preço da energia.

A partir de então, o computador local, por meio do Flask, disponibiliza recursos na rede. Dentre os recursos está a consulta de preço. Com isso, o medidor inteligente inicia a comunicação com o computador local.

O primeiro passo da comunicação é a autenticação mútua por meio do TLS. Nesse momento, o computador local fornece o certificado digital do HEMS o qual ele teve acesso juntamente com a chave privada. O medidor, por confiar no certificado, o autentica e, a partir de então, eles começam a troca de mensagens na qual o atacante — por meio do

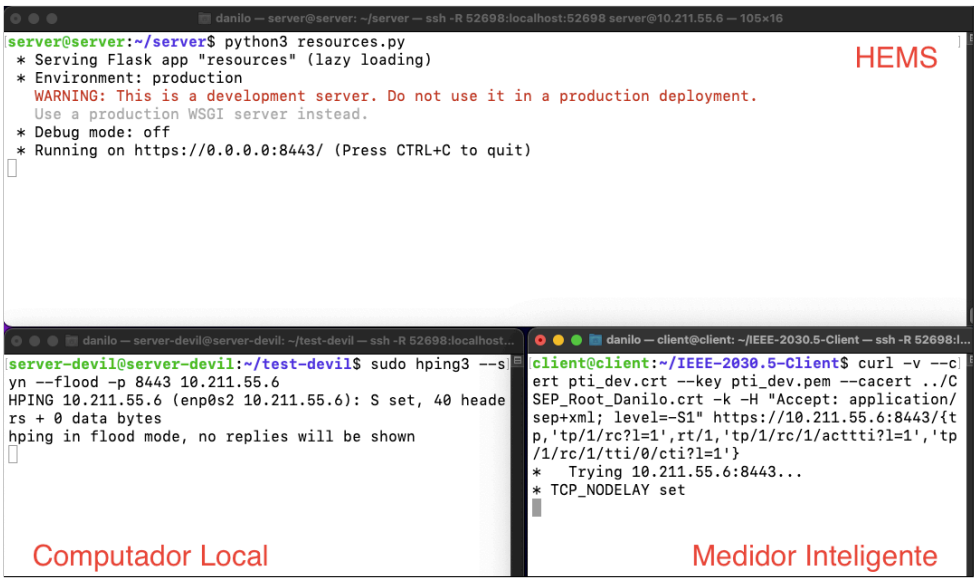


Figura 5.21: Demonstração ataque DoS ao HEMS

computador local — envia um preço falso de energia de forma a enganar o consumidor. Isso pode o induzir a aumentar o consumo de energia justamente em um momento em que a rede elétrica passa por um pico de demanda. Dessa forma, ao aumentar ainda mais o consumo, o sistema elétrico pode ter uma sobrecarga e consequentemente uma falha.

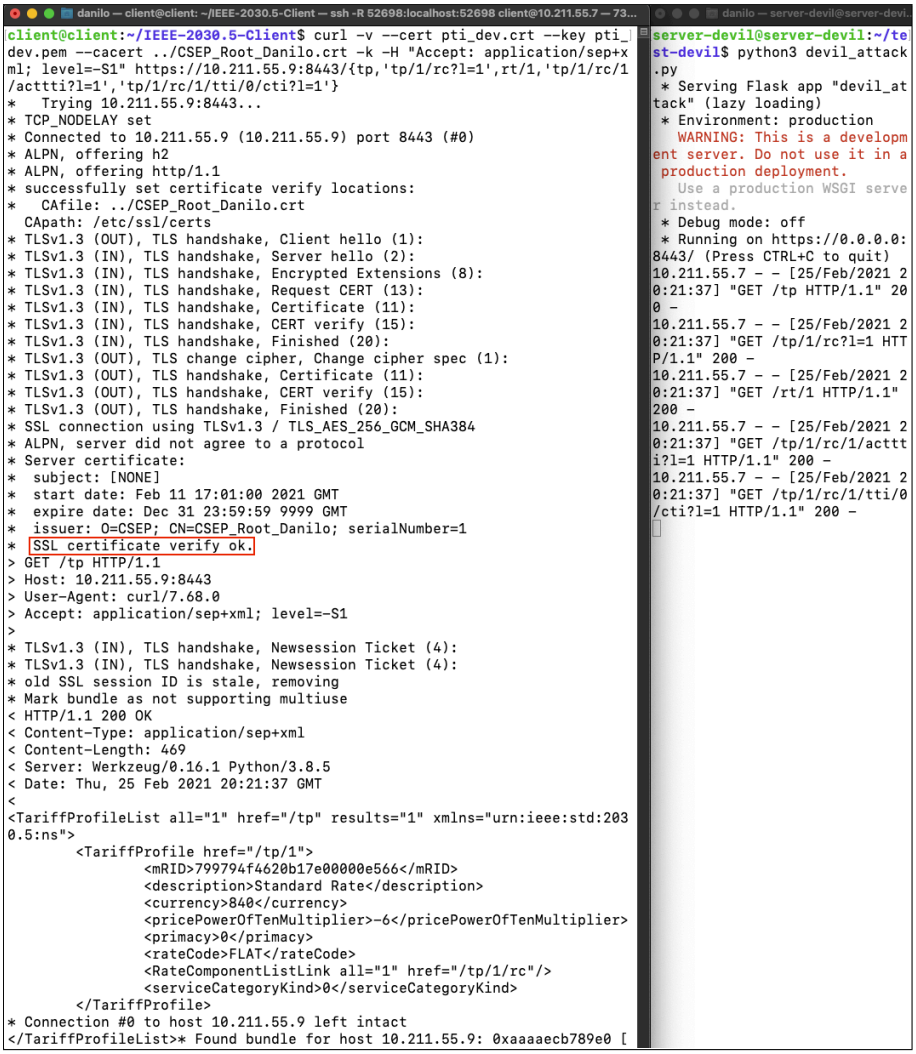
A Figura 5.22-(A) apresenta o trecho inicial da comunicação na qual ocorre a autenticação mútua dos certificados digitais e a Figura 5.22-(B) apresenta o trecho final da comunicação a qual exibe o preço da energia com valor alterado (preço igual a 105000) que é diferente do apresentado na comunicação sem comprometimento de certificado (preço igual a 113000).

Sendo assim, ocorrendo o comprometimento do certificado digital do HEMS, a rede como um todo torna-se não mais confiável e falhas podem ocorrer a qualquer momento devido às informações e comandos que são emitidos pelo atacante.

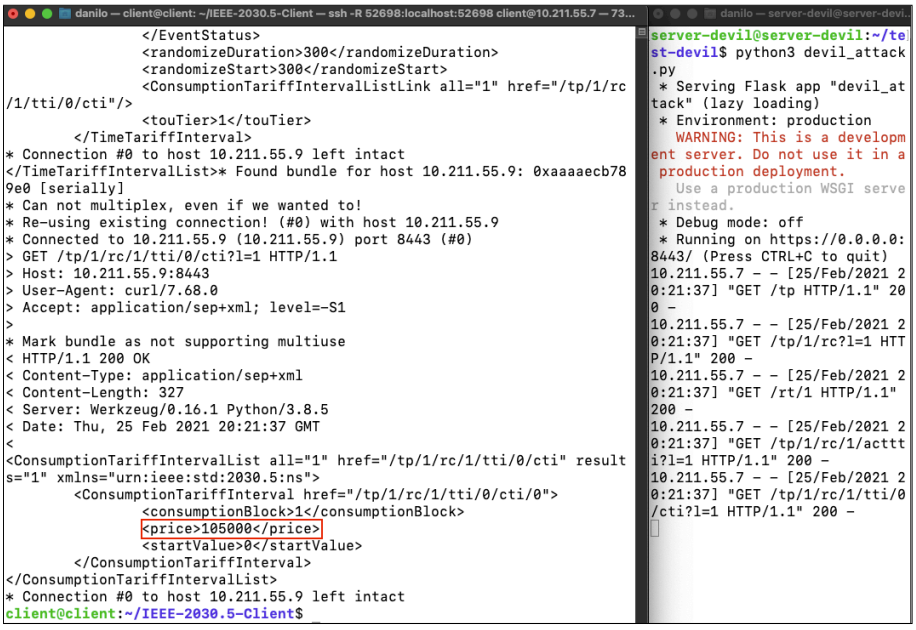
5.7.6 Cenário 3 – comprometimento do certificado digital do dispositivo cliente

Nesse cenário, o medidor inteligente possui o certificado digital comprometido. Parte-se do pressuposto que um atacante — controlando o computador local — tenha a posse da chave privada e do certificado digital do medidor inteligente.

Dessa forma, ele pode se passar pelo medidor e iniciar uma comunicação com o HEMS fornecendo uma informação falsa de medição que passa a ser registrada no HEMS. Fu-



(A) Trecho inicial



(B) Trecho final

Figura 5.22: Envio de preço alterado da energia para medidor inteligente

turamente, a concessionária pode vir a consultar o HEMS a fim de obter os valores de medição para composição da fatura a ser paga. Com isso, os valores a serem pagos não irão condizer com a realidade o que pode causar prejuízos financeiros à concessionária.

Para isso, o atacante, de posse do computador local, inicia um ataque DoS contra o medidor inteligente a fim de impedir que ele se comunique com o HEMS para registro dos valores reais de medição.

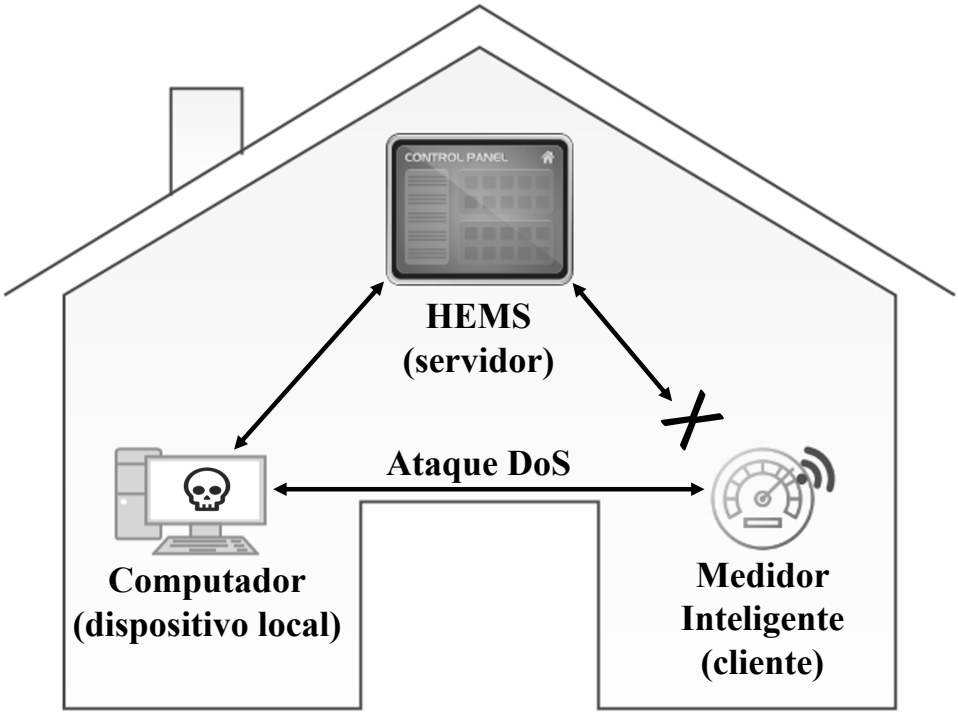


Figura 5.23: Cenário ataque ao medidor inteligente

A Figura 5.23 apresenta esse cenário de ataque. Ele é feito com o uso da ferramenta HPING3, onde o atacante direciona o ataque ao medidor inteligente de forma a impedir que ele consiga receber ou enviar solicitações de comunicação.

A demonstração desse ataque é apresentada na Figura 5.24. Nela, é possível notar que o computador local (server-devil) direciona o ataque contra o medidor inteligente (client) de forma a impedir que ele se comunique com o HEMS (server) para enviar as informações de medições corretas.

A partir de então, o atacante de posse do computador local e utilizando a chave privada e o certificado digital do medidor inteligente, inicia a comunicação com o HEMS para enviar informações falsas de medição.

Primeiramente, eles realizam a autenticação mútua. O atacante apresenta o certificado digital do medidor inteligente o qual é de conhecimento do HEMS e, por isso, ele é

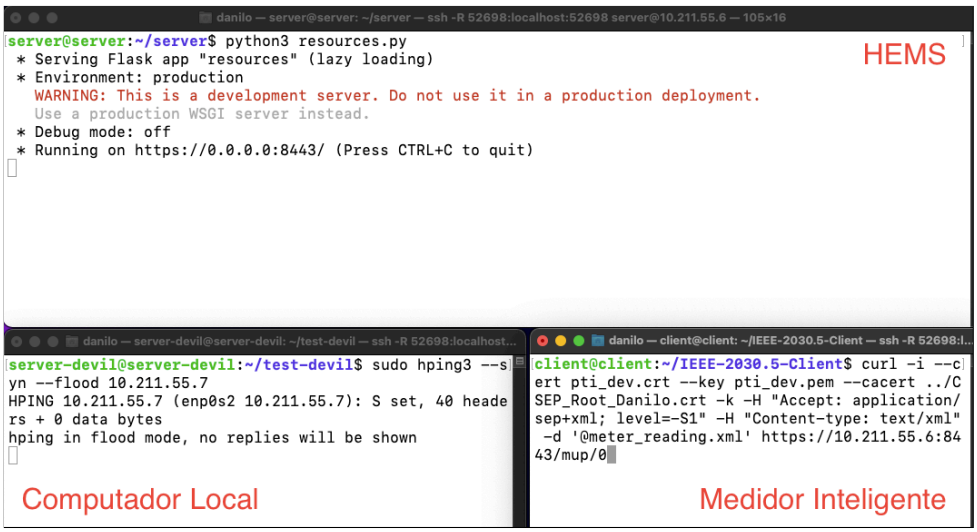


Figura 5.24: Demonstração ataque ao medidor inteligente

autenticado sem maiores problemas. Logo em seguida, o atacante envia uma medição com valor adulterado (valor igual a 5) que não condiz com o valor apresentado na comunicação sem comprometimento de certificado digital (valor igual a 9). O HEMS, por acreditar no certificado digital apresentado, aceita o valor da medição e o registra de forma que a concessionária possa fazer a leitura em um momento futuro.

O envio de um valor adulterado de medição é apresentado na Figura 5.25. Primeiramente, é mostrado o valor falso da medição a ser enviado. Logo em seguida é iniciada a autenticação mútua e o envio da informação adulterada. Posteriormente é feita uma requisição GET para confirmar o novo valor registrado no HEMS.

Com isso, uma vez comprometido o certificado digital do medidor inteligente, falsas medições podem ser enviadas para o HEMS. Posteriormente, essas medições falsas serão utilizadas pela concessionária para gerar a fatura a ser paga, o que pode levar a mesma a ter prejuízos financeiros.

5.7.7 Cenário adicional – comprometimento de certificado digital envolvendo ambiente composto por DER

Esse cenário visa abordar o problema da lacuna de segurança em um ambiente composto por DER. A topologia de teste utilizada nos cenários anteriores é estendida de forma a contemplar a comunicação com os servidores da concessionária. O medidor inteligente é substituído por um inversor inteligente representado pela mesma máquina virtual que representa o medidor inteligente (máquina "`client`").

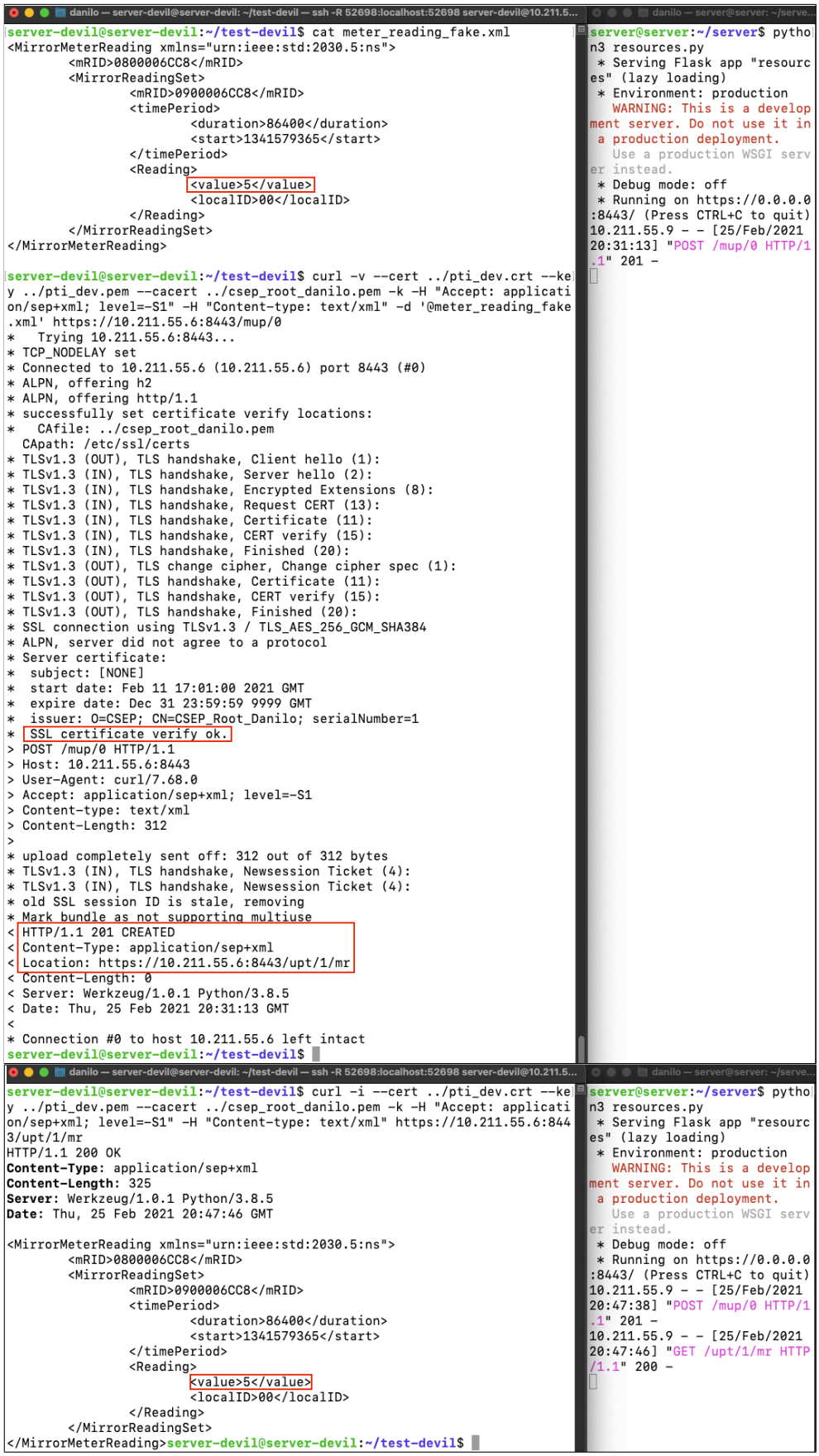


Figura 5.25: Envio de medição falsa para o HEMS

a

Como visto anteriormente, o uso particular do padrão IEEE 2030.5 no ambiente das DERs é consolidado no CSIP. Ele contém os requisitos, forma de implementação e o tipo de informação trocada entre a concessionária e a DER. No caso do ambiente residencial, a DER é logicamente representada pelo inversor inteligente.

O inversor inteligente é quem controla a produção e a liberação da energia produzida nos painéis solares. É ele quem ativa/desativa o envio de energia tanto para a rede da concessionária quanto para a rede residencial. Além disso, ele promove o gerenciamento eficiente de forma, que caso o quantitativo de energia produzida pelos painéis solares não seja suficiente para suprir a demanda da residência, ele permita que o quantitativo que falta seja suprido pela rede da concessionária.

Dentre os conjuntos de funções utilizados pelo inversor inteligente, está o conjunto *Distributed Energy Resources*. Esse conjunto de funções contém recursos que permitem o gerenciamento e o controle das DERs. Especificamente, os recursos *DERControl* e *DefaultDERControl* permitem tal controle e gerenciamento. O *DERControl* contém atributos que permitem monitorar e controlar a DER (*e.g.*, status de conexão e voltagem) em determinado intervalo de tempo definido. Já o *DefaultDERControl* permite o mesmo controle e monitoramento de forma que é ativado quando não há nenhum outro *DERControl* ativo naquele determinado momento. Ou seja, caso tenha um *DERControl* em atividade ele é quem controla e gerencia a DER, caso não tenha, o *DefaultDERControl* é quem controla e gerencia. A depender da necessidade, a concessionária pode fazer uso apenas do *DefaultDERControl* para controlar e gerenciar as DERs de forma a simplificar a operação.

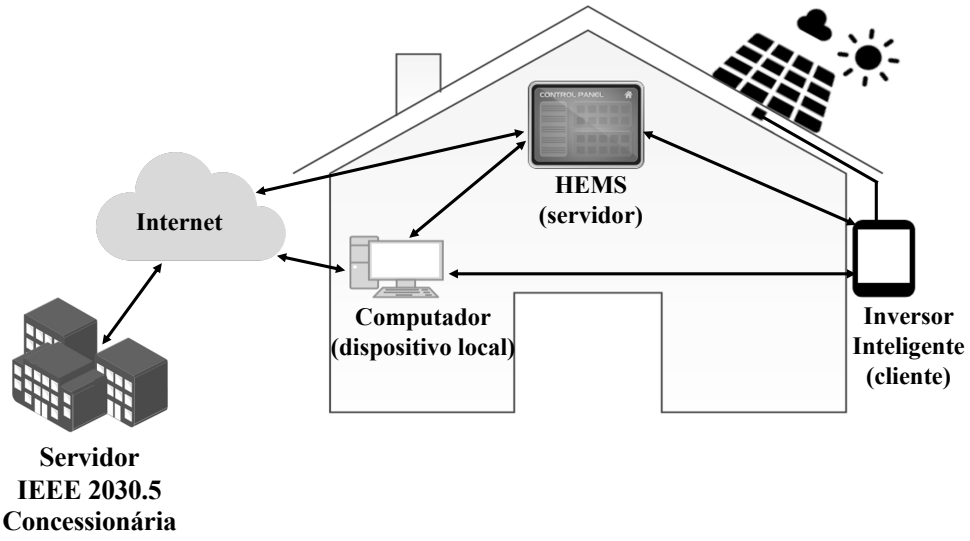


Figura 5.26: Inversor inteligente integrado a concessionária

A Figura 5.26 apresenta a topologia de teste estendida a ser utilizada nesse cenário adicional. Nela, a HAN está interligada aos servidores da concessionária o que permite que ela seja gerenciada e controlada pela concessionária. No caso específico evolvendo o controle e a gerencia do inversor inteligente, a concessionária envia o *DefaultDERControl* para o HEMS e este, por sua vez, repassa para o inversor inteligente. Também é repassada a informação do *pollRate* que define o intervalo de tempo no qual o inversor inteligente deve se comunicar com o HEMS para verificar se há alguma alteração no "*DefaultDERControl*". Conforme previsto no CSIP, a DER é responsável por garantir que todas as operações recebidas do HEMS serão processadas de forma adequada.

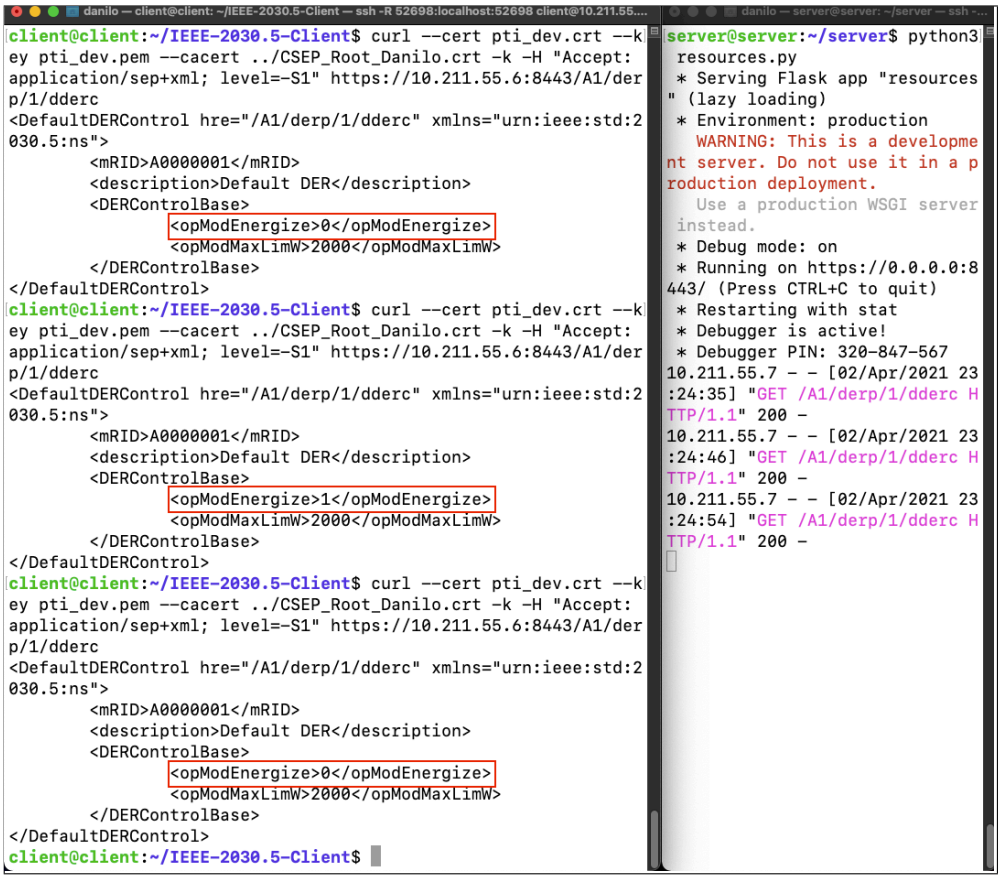


Figura 5.27: Alteração do atributo para conectar inversor inteligente

Em determinado momento, a concessionária, percebendo que há um excedente de energia produzido pelos painéis solares que não está sendo consumido pela residência, ativa o inversor inteligente de forma que ele forneça esse excedente para a rede da concessionária. Para isso, a concessionária ativa o atributo *opModEnergize* no recurso *DefaultDERControl*. O inversor inteligente, ao consultar o HEMS, recebe a alteração no *DefaultDERControl* e a implementa.

A Figura 5.27 apresenta essa dinâmica. E em um primeiro momento o inversor inte-

ligente consulta o HEMS e configura seus parâmetros conforme definido no *DefaultDER-Control*. Ao fim do *pollRate*, o inversor realiza nova consulta ao HEMS para verificar se há alguma alteração a ser executada. Ao receber a informação de alteração no atributo *opModEnergize*, o inversor inteligente ativa o envio do excedente de energia para a rede da concessionária ajudando a suprir a demanda em outros lugares de forma a não permitir que falhas relacionadas a sobrecargas venham a ocorrer. Uma vez suprida a demanda de energia, a concessionária pode alterar novamente o atributo de modo a desativar o inversor inteligente para que ele não mais envie o excedente de energia para a rede da concessionária. Com isso, a concessionária mantém a estabilidade na rede elétrica.

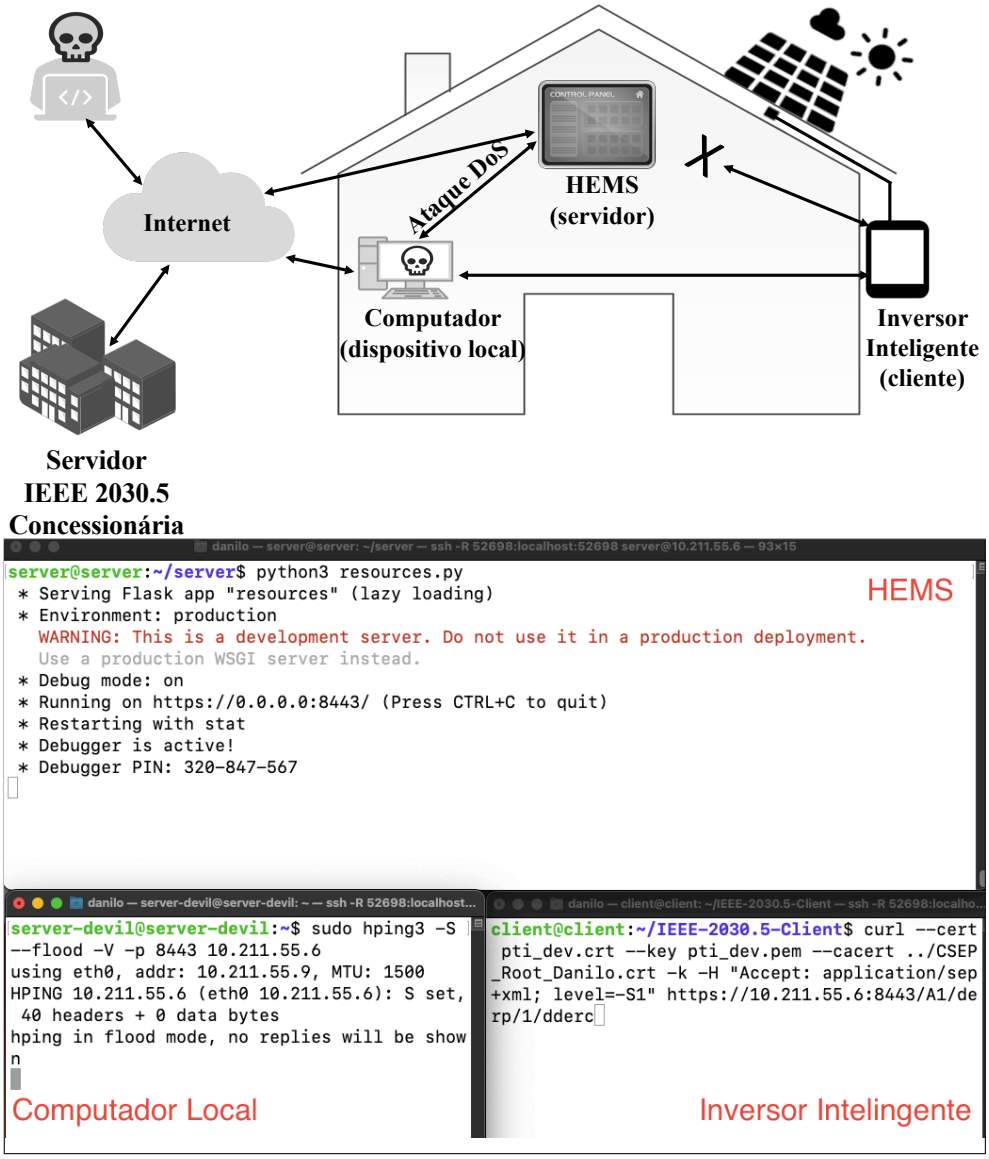


Figura 5.28: Ataque em ambiente DER

Nesse contexto, um atacante de posse da chave privada do certificado digital do HEMS e controlando o computador local pode iniciar um ataque DoS contra o HEMS de forma a

impedir que o inversor inteligente se comunique com ele. Com isso, o inversor não recebe a informação se houve ou não alguma alteração no *DefaultDERControl*. A Figura 5.28 apresenta na parte superior esse cenário de ataque DoS contra o HEMS e na parte inferior apresenta na prática a realização desse ataque.

Com a impossibilidade de comunicação entre HEMS e inversor inteligente, o atacante, fazendo uso do certificado digital do HEMS, começa a se comunicar com o inversor inteligente. A autenticação mútua entre ambos é feita sem maiores problemas. A partir de então, o inversor inteligente começa a interagir com o computador local e a consumir os recursos por ele disponibilizados. Dentre os recursos disponibilizados está o *DefaultDERControl*, no qual o atacante pode controlar o inversor inteligente.

Em determinado momento anterior ao ataque DoS, a concessionária, por não mais necessitar da energia proveniente do inversor inteligente, desativou o atributo *opModEnergize*. Já sob o ataque DoS, o atacante, que agora se comunica com o inversor inteligente, pode alterar o *opModEnergize* de forma a ativar novamente o envio de energia para a rede da concessionária. A Figura 5.29 apresenta essa interação entre atacante e inversor inteligente de modo a promover a alteração do atributo *opModEnergize*. Em um primeiro momento o inversor consulta o *DefaultDERControl* presente no computador local e verifica que o atributo *opModEnergize* está desativado da mesma forma que anteriormente havia sido designado pela concessionária. Logo em seguida, o atacante altera o atributo de forma a ativá-lo. Com isso, o inversor inteligente, ao consultar novamente o recurso ao fim do *poolRate*, recebe a alteração e ativa o envio de energia para a rede da concessionária.

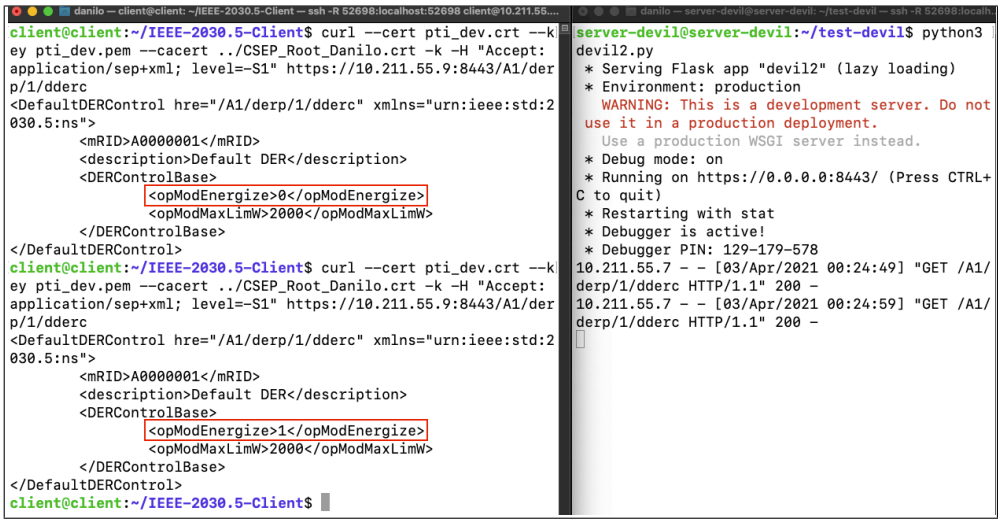


Figura 5.29: Ataque para ativação inversor inteligente

A depender quantidade de energia a ser enviada para a rede elétrica com a ativação do inversor inteligente, a rede elétrica da concessionária pode não estar preparada para

suportar a alta quantidade de energia entrante. Com isso, falhas na rede elétrica podem ocorrer de forma a causar tanto prejuízos a concessionária quanto aos demais consumidores.

De modo semelhante ao comprometimento do certificado do medidor inteligente, caso um inversor inteligente tenha seu certificado comprometido, o atacante, de posse da chave privada, pode inciar um ataque DoS contra o inversor inteligente impedindo que o mesmo se comunique com o HEMS. Com isso, o atacante, fazendo uso do certificado digital do inversor inteligente, inicia uma comunicação com o HEMS apresentando tal certificado que irá ser validado e autenticado. Logo em seguida o atacante, se passando pelo inversor inteligente, começa a enviar informações falsas (*e.g.*, capacidade energética, status operacional e alarmes) que são registradas no HEMS e repassadas a concessionária.

Por exemplo, um inversor inteligente possui capacidade de fornecer no máximo 20 *ampères* de corrente elétrica. Essa informação é repassada ao HEMS por meio do método POST utilizando o atributo *rtgA* presente no recurso *DERCapability*. Tal informação é repassada à concessionária e ela a utiliza para tomada de decisão que visa promover a estabilidade da rede. Em determinado momento, o atacante, de posse da chave privada do certificado digital do inversor inteligente, começa um ataque DoS contra o inversor nos mesmos moldes do ataque apresentado contra o medidor inteligente. Dessa forma, o inversor fica impossibilitado de comunicar com o HEMS. Logo em seguida, o atacante inicia a comunicação com o HEMS apresentando o certificado digital do inversor inteligente. Realizada a autenticação, o atacante envia uma informação falsa alterando o atributo *rtgA* para um valor superior à real capacidade do inversor. Com isso, se utilizando da informação falsa, a concessionária adéqua a capacidade energética da rede de forma equivocada o que pode levar a falhas.

Tal dinâmica é apresentada na Figura 5.30. Nela, em um primeiro momento, o atacante, se passando pelo inversor inteligente, realiza uma consulta para saber qual o real valor da sua capacidade de corrente elétrica (*rtgA*) que está registrado no HEMS — que no caso é 20. Recebendo a informação, o atacante gera um novo valor (44) que será enviado ao HEMS — via método POST — e, posteriormente, à concessionária. A concessionária, recebendo o valor falso, entende que, com tal capacidade, a demanda energética pode ser elevada e ainda será suprida. No entanto, com o aumento da demanda, a rede não consegue suprir e fica sobrecarregada dando início às falhas.

Sendo assim, após as demonstrações práticas, é notável o problema advindo da não revogação e não expiração dos certificados digitais. Uma vez comprometido, o certificado

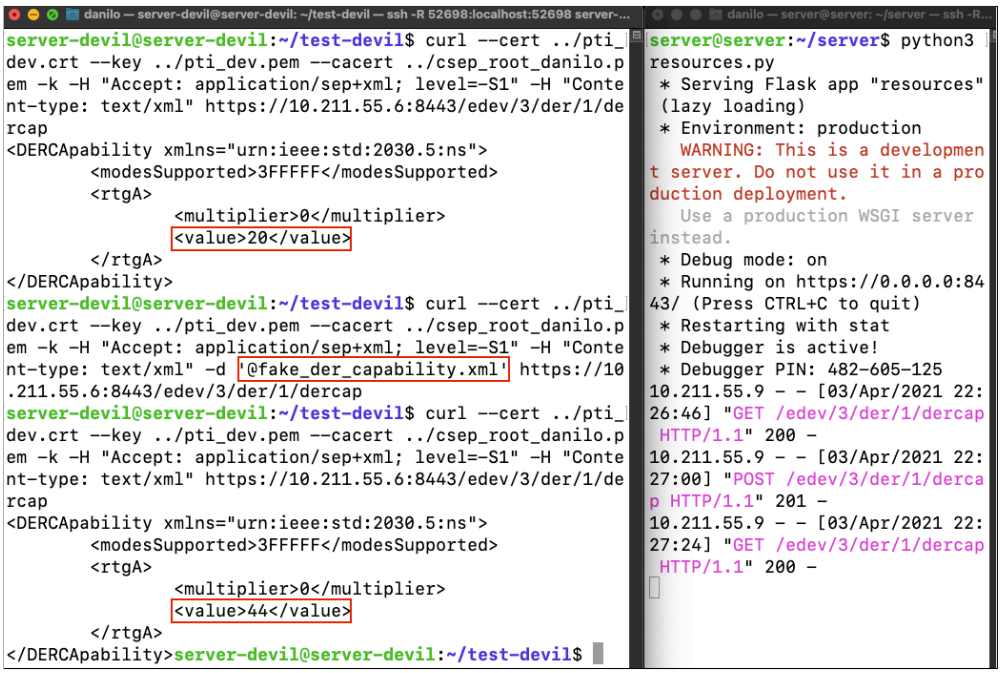


Figura 5.30: Envio de informação de falsa capacidade da DER

digital pode ser usado por atacantes na realização não só dessas atividades demonstradas como também de diversas outras atividades maliciosas. Sendo o mais preocupante a inexistência de uma forma de inviabilizar o uso do certificado comprometido em qualquer outro lugar por qualquer outro dispositivo.

Capítulo 6

Conclusão

Esse trabalho promoveu um estudo detalhado sobre o IEEE 2030.5-2018 — padrão que visa atender às necessidades de interoperabilidade, monitoramento e controle dos dispositivos inteligentes no ambiente de energia do usuário — abordando tanto aspectos arquiteturais e técnicos da comunicação quanto questões relacionadas a segurança. Além disso, também foi apresentado um mapeamento do modelo de dados do padrão para casos de uso reais juntamente com uma prova de conceito que visa demonstrar uma lacuna de segurança existente na *Manufacturing PKI*

O padrão faz uso extensivo de uma interface *RESTful* e foi projetado para funcionar com qualquer tecnologia de comunicação que suporte a pilha de protocolo TCP/IP, o que permite implementações em grande escala e em diferentes ambientes. Ele também não está restrito apenas ao domínio elétrico, embora tenha sido aplicado principalmente nesse contexto até o momento. O grande número de conjuntos de funções definidos no padrão permite maior controle e gerenciamento dos dispositivos inteligentes e proporciona uma melhor experiência para os usuários proprietários de tais dispositivos. Essa interação também se estende para fora do âmbito das residências, uma vez que o padrão também permite que as concessionárias façam a gestão do ambiente doméstico do usuário.

A escolha do seu uso dentro do programa *California's Rule 21* destacou o potencial disruptivo do IEEE 2030.5-2018. Adicionalmente, motivou uma revisão da norma de forma a responder às necessidades do programa e para promover uma melhor integração e suporte com os recursos energéticos distribuídos. O programa *California's Rule 21* emprega o padrão para promover a comunicação entre os recursos de energia distribuídos e as concessionárias, permitindo que as concessionárias os gerenciem e controlem a fim de otimizar a demanda e o fornecimento de energia elétrica.

Mesmo com um uso prático tão relevante, a literatura científica revisando e analisando o IEEE 2030.5 ainda é escassa, principalmente considerando a revisão de 2018. Em particular, os aspectos de segurança do padrão — uma parte essencial de qualquer sistema de informação moderno — permanecem amplamente não estudados, especialmente nos círculos acadêmicos. Isso se manifesta na forma de lacunas, principalmente no que diz respeito ao manuseio dos certificados digitais. Um ponto a se destacar é a falta de suporte para revogação e a não expiração dos certificados digitais IEEE 2030.5 que representam uma ameaça potencialmente séria. Esse problema assume uma proporção ainda maior no contexto dos certificados digitais das MCAs e MICAs, uma vez que tais certificados são responsáveis pela emissão de outros certificados digitais na hierarquia da infraestrutura de chave pública (*Manufacturing PKI*).

A prova de conceito realizada nesse trabalho aborda essa lacuna de segurança e suas possíveis consequências. Nela, são apresentados cenários que demonstram na prática os problemas advindos do comprometimento do certificado digital e da sua não revogação e não expiração. Problemas esses que vão desde prejuízos financeiros até falhas em todo o sistema elétrico. Os dispositivos foram representados por máquinas virtuais com sistema operacional Linux, onde, em cada máquina, foram implementadas as funcionalidades referentes a cada dispositivo. A comunicação segura entre um dispositivo que consome recursos (cliente) e um dispositivo que disponibiliza recursos (servidor) foi colocada a prova frente à possibilidade do comprometimento dos certificados digitais e foi constatada a falha proveniente da referida lacuna de segurança.

Embora soluções descentralizadas de *black/white lists* tenham sido mencionadas para mitigar esse problema, elas podem sofrer problemas de inconsistências relativas ao gerenciamento, uma vez que não há consenso sobre quem deve relatar certificados comprometidos. Além disso, o uso de *black lists* pode enfrentar obstáculos de implementação de ordem prática acentuados pelas restrições de armazenamento geralmente severas dos dispositivos inteligentes típicos.

Atualmente, no programa *California's Rule 21*, as DERs são configuradas previamente com o LFDI dos dispositivos servidores aos quais elas irão se conectar [44]. Ou seja, as DERs são dotadas de uma *whitelist* dos dispositivos servidores. Caso um certificado digital de um desses dispositivos servidores venha a ser comprometido, é necessário remover o LFDI correspondente ao certificado de todas as DERs. No entanto, como visto anteriormente, o uso tanto de *whitelist* quanto de *blacklist* está sujeito a problemas de inconsistências.

Exigir que os fabricantes dos dispositivos forneçam meios seguros de proteção e armazenamento da chave privada é visto como uma solução à prova de falhas, uma vez que a norma não fornece nenhum procedimento de segurança para resolver possíveis problemas dessa natureza de forma geral e global. No entanto, com a evolução da tecnologia e com as possíveis falhas de projetos e até mesmo falhas humanas, violações de segurança podem ocorrer ou ser descobertas a qualquer momento, e a falta de medidas de segurança alternativas padronizadas pode tornar o sistema como um todo vulnerável.

Desde 2011, houve pelo menos 15 problemas em autoridades certificadoras que vieram ao conhecimento público [46]. Um caso de destaque ocorreu em 2018 com a autoridade certificadora da Symantec. Naquele ano, o Google passou a não considerar mais confiável a autoridade certificadora devido a falhas ocorridas na mesma que envolveram desde acesso não autorizado até falta de auditoria [46]. Com isso, sites que faziam uso dos certificados digitais emitidos pela Symantec passaram a não ser mais acessíveis pelo navegador Google Chrome. Em 2020, em um caso mais recente, a autoridade certificadora Digicert teve que revogar 50.000 certificados digitais devido a um erro interno no processo de emissão [18].

Se até mesmo grandes empresas que são dedicadas ao ramo de emissão, controle e gerenciamento de certificados digitais passam por problemas para garantir a segurança no processo como um todo, então não é razoável esperar que fabricantes de *hardware* o façam.

Sendo assim, espera-se que a indústria e a academia possam colaborar em um futuro próximo para preencher essa lacuna de forma a promover um ambiente mais seguro e com medidas alternativas de segurança que possam ser utilizadas caso alguma outra medida venha a sofrer falhas.

Estudos futuros podem ser feitos visando propor uma solução que venha a preencher a lacuna de segurança exposta. O uso de CRL e OCSP, em um primeiro momento, torna-se inviável devido aos motivos apresentados nesse trabalho. No entanto, um estudo mais aprofundado e detalhado poderia identificar meios e alternativas para viabilizar o uso dessas tecnologias, podendo, até mesmo, chegar ao ponto de elencar argumentos para contrapor a tese da não garantia de que todos os dispositivos estejam conectados à Internet. Com isso, uma vez conectados à Internet, o uso do CRL e do OCSP torna-se viável. A utilização de módulos criptográficos também pode fazer parte das possíveis soluções para preencher a lacuna de segurança. Entretanto, é necessário um estudo mais detalhado a fim de que haja uma perfeita implementação do módulo criptográfico de modo que eles mantenham sua completa funcionalidade frente às mais diversas restrições,

principalmente restrições de *hardware* que são características próprias dos dispositivos IoT.

Referências

[1] Adefarati, T.; Bansal, R. Chapter 2 - energizing renewable energy systems and distribution generation. In *Pathways to a Smarter Power System*, A. Taşcıkaraoğlu and O. Erdinç, Eds. Academic Press, 2019, pp. 29 – 65.

[2] Ardigo, J. D. *Modelo de Infra-estrutura de Chaves Públicas como Organização Virtual para Processos de Avaliação Somativa à Distância Modelo de Infra-estrutura de Chaves Públicas como Organização Virtual para Processos de Avaliação Somativa à Distância*. Tese de Doutorado, Universidade Federal de Santa Catarina, 2004.

[3] Atzori, L.; Iera, A.; Morabito, G. The Internet of Things: A survey. *Computer Networks 54*, 15 (2010), 2787 – 2805.

[4] Baker, J.; Cordeiro, P. G.; Doepke, T.; Hossain-McKenzie, S.; Howerter, C. M.; Jacobs, N.; Jose, D.; Lai, C. F.; Zhao, J. General Requirements for Designing and Implementing a Cryptography Module for Distributed Energy Resource (DER) Systems. Tech. rep., Sandia National Laboratories, 2018.

[5] Berners-Lee, T.; Fielding, R. T.; Masinter, L. M. Uniform Resource Identifier (URI): Generic Syntax. RFC 3986, Jan. 2005.

[6] Boeyen, S.; Santesson, S.; Polk, T.; Housley, R.; Farrell, S.; Cooper, D. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. RFC 5280, May 2008.

[7] Bushby, S. T.; Holmberg, D. G. Standardization of Smart Grid Customer Interfaces. *Distribution and Utilization 32* (2015).

[8] California Energy Commission. Rule 21 Smart Inverter Working Group. `https://www.energy.ca.gov/programs-and-topics/topics/energy-assessment/rule-21-smart-inverter-working-group`. (acessado em 10 Junho 2020).

[9] California Public Utilities Commission. Rule 21 Interconnection. `https://www.cpuc.ca.gov/Rule21/`. (acessado em 08 Junho 2020).

[10] California Public Utilities Commission. Smart Inverter Working Group. `https://www.cpuc.ca.gov/General.aspx?id=4154`. (acessado em 10 Junho 2020).

[11] California Solar Initiative. California Leads the Nation in Distributed Generation. `https://www.californiadgstats.ca.gov`. (acessado em 10 Junho 2020).

[12] Carter, C.; Onunkwo, I.; Cordeiro, P.; Johnson, J. Cyber Security Assessment of Distributed Energy Resources. *Photovoltaic Specialist Conference (PVSC)* (2017), 2135–2140.

[13] Cheshire, S.; Krochmal, M. DNS-Based Service Discovery. RFC 6763, 2013.

[14] Cheshire, S.; Krochmal, M. Multicast DNS. RFC 6762, 2013.

[15] Common Smart Inverter Profile Working Group. Common Smart Inverter Profile: IEEE 2030.5 Implementation Guide for Smart Inverters. `https://sunspec.org/wp-content/uploads/2018/03/CSIPImplementationGuidev2.003-02-2018-1.pdf`, 2018.

[16] de Carvalho, R. S.; Saleem, D. Recommended Functionalities for Improving Cybersecurity of Distributed Energy Resources. *Resilience Week (RWS) 1* (2019), 226–231.

[17] dos Santos Alonso, A. M.; Carlos Afonso, L.; Brandao, D. I.; Tedeschi, E.; Marafão, F. P. Considerations on Communication Infrastructures for Cooperative Operation of Smart Inverters. *IEEE 15th Brazilian Power Electronics Conference and 5th IEEE Southern Power Electronics Conference (COBEP/SPEC)* (2019), 1–6.

[18] Ducklin, P. Digicert revokes a raft of web security certificates. `https://nakedsecurity.sophos.com/2020/07/13/digicert-revokes-a-raft-of-web-security-certificates/`. (acessado em 01 Fevereiro 2021).

[19] Eddy, W. TCP SYN Flooding Attacks and Common Mitigations. RFC 4987, Aug. 2007.

[20] Eletric Power Reearch Institute. IEEE-2030.5-Client. `https://www.epri.com/research/products/000000003002014087`. (acessado em 01 Fevereiro 2021).

[21] Fattahi, J.; Samadi, M.; Erol-Kantarci, M.; Schriemer, H. Transactive Demand Response Operation at the Grid Edge using the IEEE 2030.5 Standard. *Engineering 6*, 7 (2020), 801–811.

[22] Fielding, R. T.; Taylor, R. N. Principled design of the modern Web architecture. *Proceedings of the 2000 International Conference on Software Engineering. ICSE 2000 the New Millennium* (2000), 407–416.

[23] Ghalib, M.; Ahmed, A.; Al-Shiab, I.; Bouida, Z.; Ibnkahla, M. Implementation of a Smart Grid Communication System Compliant with IEEE 2030.5. *IEEE International Conference on Communications Workshops (ICC Workshops)* (2018), 1–6.

[24] Goudos, S. K.; Sarigiannidis, P.; Dallas, P. I.; Kyriazakos, S. Communication protocols for the IoT-based smart grid. In *IoT for Smart Grids*. Springer, 2019, pp. 55–83.

[25] Henry, J.; Ramirez, R.; Cleveland, F.; Lee, A.; Seal, B.; Tansy, T.; Fox, B.; Pochiraju, A. Cyber security requirements and recommendations for CSI RD&D solicitation# 4 distributed energy resource communications, 2015.

[26] Hossain, M. M.; Peng, C. Cyber–physical security for on-going smart grid initiatives: a survey. *IET Cyber-Physical Systems: Theory Applications 5*, 3 (2020), 233–244.

[27] IEEE. IEEE Guide for Smart Grid Interoperability of Energy Technology and Information Technology Operation with the Electric Power System (EPS), End-Use Applications, and Loads. *IEEE Std 2030-2011* (2011), 1–126.

[28] IEEE. IEEE Adoption of Smart Energy Profile 2.0 Application Protocol Standard. *IEEE Std 2030.5-2013* (2013), 1–348.

[29] IEEE. IEEE Adoption of Smart Energy Profile 2.0 Application Protocol Standard (Draft). *IEEE P2030.5/D1, June 2013* (2013), 1–348.

[30] IEEE. IEEE Draft Standard for Smart Energy Profile Application Protocol. *IEEE P2030.5/D1, September 2017* (2017), 1–342.

[31] IEEE. IEEE Approved Draft Standard for Smart Energy Profile Application Protocol. *IEEE P2030.5/D2, March 2018* (2018), 1–358.

[32] IEEE. IEEE Standard for Interconnection and Interoperability of Distributed Energy Resources with Associated Electric Power Systems Interfaces. *IEEE Std 1547-2018 (Revision of IEEE Std 1547-2003)* (2018), 1–138.

[33] IEEE. IEEE Standard for Smart Energy Profile Application Protocol. *IEEE Std 2030.5-2018 (Revision of IEEE Std 2030.5-2013)* (2018), 1–361.

[34] Jacobsen, R.; Mikkelsen, S. Infrastructure for Intelligent Automation Services in the Smart Grid. *Wireless Personal Communications 76* (2014), 125–147.

[35] Jain, A. K.; Nagarajan, A.; Chernyakhovskiy, I.; Bowen, T.; Mather, B.; Cochran, J. Evolution of Distributed Energy Resource Grid Interconnection Standards for Integrating Emerging Storage Technologies. *North American Power Symposium (NAPS)* (2019), 1–6.

[36] Johnson, J. ieee-2030.5-client. `https://github.com/catch-twenty-two/ieee-2030.5-client`. (acessado em 01 Fevereiro 2021).

[37] Johnson, J.; Onunkwo, I.; Cordeiro, P.; Wright, B.; Jacobs, N.; Lai, C. Assessing DER network cybersecurity defences in a power-communication co-simulation environment. *IET Cyber-Physical Systems: Theory and Applications 5*, 3 (2020), 274–282.

[38] Johnson, J. T. PV Cybersecurity Final Report. Tech. rep., Sandia National Laboratories, 2019.

[39] Kounev, V.; Tipper, D. Advanced Metering and Demand Response communication performance in Zigbee based HANs. *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)* (2013), 31–36.

[40] Lai, C.; Jacobs, N.; Hossain-McKenzie, S.; Carter, C.; Cordeiro, P.; Onunkwo, I.; Johnson, J. Cyber security primer for DER vendors, aggregators, and grid operators. Tech. rep., Sandia National Laboratories, 2017.

[41] Li, B.; Jia, B.; Cao, W.; Tian, S.; Qi, B.; Sun, Y.; Zhu, W.; Zheng, A. Application Prospect of Edge Computing in Power Demand Response Business. *Dianwang Jishu/Power System Technology 42*, 1 (2018), 79–87.

[42] Ling, L.; Hongyong, Y.; Xia, C. Model Differences between IEC 61970/61968 and IEC 61850. *International Conference on Intelligent System Design and Engineering Applications* (2013), 938–941.

[43] Lu, Y.; Ding, Y.; Duan, Q.; Li, X.; Tian, Y. Upper-Middleware Development of Smart Energy Profile 2.0 for Demand-Side Communications in Smart Grid. *IECON 2018 - 44th Annual Conference of the IEEE Industrial Electronics Society* (2018), 306–310.

[44] Lum, G. (comunicação privada, 2020-06-29).

[45] Lynn, K.; Sturek, D. Extended Multicast DNS. Internet-Draft draft-lynn-dnsext-site-mdns-01, IETF Secretariat, March 2011.

[46] Milber, M. What happens when a certificate authority is compromised? `https://www.teiss.co.uk/what-happens-when-a-certificate-authority-is-compromised/`. (acessado em 01 Fevereiro 2021).

[47] Myers, M.; Tschofenig, H. Online Certificate Status Protocol (OCSP) Extensions to IKEv2. RFC 4806, Feb. 2007.

[48] Nagarajan, A.; Palmintier, B.; Baggu, M. Advanced inverter functions and communication protocols for distribution management. *IEEE/PES Transmission and Distribution Conference and Exposition (T&D)* (2016), 1–5.

[49] Obert, J.; Cordeiro, P.; Johnson, J.; Lum, G.; Tansy, T.; Pala, M.; Ih, R. Recommendations for trust and encryption in der interoperability standards. Tech. rep., Sandia National Laboratories, 2019.

[50] Obi, M.; Slay, T.; Bass, R. Distributed energy resource aggregation using customer-owned equipment: A review of literature and standards. *Energy Reports 6* (2020), 2358–2369.

[51] Ogle, J. P.; Touhiduzzaman, M.; Nguyen, Q. H.; Thekkumparambath Mana, P. CReST-VCT System Integration Framework. Tech. rep., Pacific Northwest National Lab.(PNNL), 2020.

[52] Pala, D.; Proserpio, G. Model-driven development of a standard-compliant Customer Energy Manager. *International Symposium on Smart Electric Distribution Systems and Technologies (EDST)* (2015), 324–328.

[53] Renjit, A. Chapter 10 - Communications, cybersecurity, and the internet of things for microgrids. In *Distributed Energy Resources in Microgrids*. Academic Press, 2019, pp. 275–290.

[54] Rescorla, E. HTTP Over TLS. RFC 2818, May 2000.

[55] Rescorla, E.; Dierks, T. The Transport Layer Security (TLS) Protocol Version 1.2. RFC 5246, Aug. 2008.

[56] Richter, A.; van der Laan, E.; Ketter, W.; Valogianni, K. Transitioning from the traditional to the smart grid: Lessons learned from closed-loop supply chains. *International Conference on Smart Grid Technology, Economics and Policies (SG-TEP)* (2012), 1–7.

[57] Ronacher, A. Flask. `https://flask.palletsprojects.com/en/1.1.x/`. (acessado em 01 Fevereiro 2021).

[58] Sanfilippo, S. Hping. `http://www.hping.org/`. (acessado em 01 Fevereiro 2021).

[59] Sarker, P. S.; Venkataramanan, V.; Cardenas, D. S.; Srivastava, A.; Hahn, A.; Miller, B. Cyber-Physical Security and Resiliency Analysis Testbed for Critical Microgrids with IEEE 2030.5. *Workshop on Modeling and Simulation of Cyber-Physical Energy Systems* (2020), 1–6.

[60] Sebastian, D. J.; Agrawal, U.; Tamimi, A.; Hahn, A. DER-TEE: Secure distributed energy resource operations through trusted execution environments. *IEEE Internet of Things Journal 6*, 4 (2019), 6476–6486.

[61] Sebastian, D. J.; Hahn, A. Exploring emerging cybersecurity risks from network-connected DER devices. *North American Power Symposium (NAPS)* (2017), 1–6.

[62] Seitenfus, F.; Barriquello, C.; Neves Canha, L.; Pedretti, A.; Silva Santana, T.; Nadal, Z. Simulation and analysis of OpenADR agents using VOLTTRON platform. *IEEE PES Conference on Innovative Smart Grid Technologies, ISGT Latin America 2019* (2019), 1–6.

[63] Simpson, R. IEEE 2030.5-2013 (Smart Energy Profile 2.0) - An Overview for KSGA. `https://www.robbysimpson.com/prezzos/IEEE_2030_5_Seoul_Simpson_20150424.pdf`, 2015. (acessado em 10 Junho 2020).

[64] Slicker, M. IEEE-2030.5-Client. `https://github.com/epri-dev/IEEE-2030.5-Client`. (acessado em 01 Fevereiro 2021).

[65] Soyoye, O. T.; Stefferud, K. C. Cybersecurity Risk Assessment for California's Smart Inverter Functions. *IEEE CyberPELS (CyberPELS)* (2019), 1–5.

[66] Sun, C.; Zhu, R.; Liu, C. Cyber Attack and Defense for Smart Inverters in a Distribution System. *CIGRE Study Committee D2 Colloquium, Helsinki, Finland* (2019).

[67] van Kerkhoven, J.; Charlebois, N.; Robertson, A.; Gibson, B.; Ahmed, A.; Bouida, Z.; Ibnkahla, M. IPv6-Based Smart Grid Communication over 6LoWPAN. *IEEE Wireless Communications and Networking Conference (WCNC)* (2019), 1–6.

[68] Zhen Zhao; Agbossou, K.; Cardenas, A. Connectivity for Home Energy Management applications. *IEEE PES Asia-Pacific Power and Energy Engineering Conference (APPEEC)* (2016), 2175–2180.

APÊNDICE A – Conjuntos de funções e recursos IEEE 2030.5-2018

A.1 Support Resources

A.1.1 Conjunto de funções Device Capabilities

É composto por um único recurso denominado *DeviceCapability* e tem por finalidade enumerar os recursos e os conjuntos de funções suportados por um dispositivo. Solicitantes podem consultar esse recurso para descobrir se um outro determinado recurso ou conjunto de funções está disponível e qual é a sua localização no dispositivo que está sendo consultado.

O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /dcap
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Erro;

A.1.2 Conjunto de funções Self Device

É composto de um único recurso denominado *SelfDevice* e tem por finalidade prover uma interface para divulgação de informações gerais sobre o dispositivo, tais como versão do *software* e o número de identificação. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /sdev
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Erro;

A.1.3 Conjunto de funções End Device

Localizado em um dispositivo servidor/controlador da HAN, é basicamente um contêiner que tem por finalidade prover uma interface para a troca de informações, tanto gerais quanto específicas de determinado dispositivo. Semelhante a uma ficha de registro a qual contém informações sobre determinado ativo. Tal ficha fica armazenada em um dispositivo servidor/controlador juntamente com outras fichas de outros dispositivos presentes na rede. É composto de quatro recursos sendo eles:

- **EndDevice** – instância própria do *EndDevice* que provê a interface para a troca de informações. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;
- **EndDeviceList** – lista dos recursos *EndDevice* hospedados em um determinado dispositivo. O dispositivo que hospeda tal recurso tem a função de persistir a informação dos *EndDevice* ao longo do tempo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;
- **Registration** – apresenta informações relacionadas ao processo de registro do dispositivo em questão, como por exemplo o PIN e a data de registro. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/rg
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;
- **DeviceStatus** – apresenta o atual estado operacional do dispositivo em questão. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/dstat
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Opcional;

A.1.4 Conjunto de funções Function Set Assignments

Define uma coleção de referências para instâncias de conjuntos de funções. Tal coleção é usada por um recurso *EndDevice* para indicar ao dispositivo qual recurso ele deve usar para um propósito específico. Por exemplo, determinado dispositivo controlador de medidores inteligentes deseja que um grupo específico de medidores inteligentes de uma área/região usem um determinado conjunto de funções específico de tempo e preço. O recurso *Function Set Assignments* é quem irá conter as instâncias das referências para os conjuntos de funções que devem ser usados pelo grupo. O conjunto de funções *Function Set Assignments* é composto por dois recursos:

- **FunctionSetAssignments** – fornece uma coleção de recursos para um determinado dispositivo consumir. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/fsa/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;
- **FunctionSetAssignmentsList** – lista todos os recursos *FunctionSetAssignments* presentes no dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/fsa
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

A.1.5 Conjunto de funções Subscription/Notification

Tem por finalidade promover um mecanismo de notificação para que um dispositivo seja informado sobre alterações em um recurso. O conjunto de funções permite que dispositivos façam subscrição em um determinado recurso e caso o recurso sofra alterações todos os dispositivos que estão subscritos são notificados. Por exemplo, vários dispositivos podem fazer a subscrição para acompanhar o preço de determinada *commodity*. Ao ser alterado o preço, todos os dispositivos subscritos serão notificados dessa alteração. É composto pelos seguintes recursos:

- **Subscription** – contém as informações relacionadas a uma assinatura de um dispositivo que deseja receber atualizações de um recurso automaticamente. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/sub/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Obrigatório;

- **SubscriptionList** – lista as instâncias *Subscription* relacionadas ao dispositivo em questão, ou seja, lista todas as suas subscrições. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/sub
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Obrigatório; DELETE: Erro;

- **Notification** – instância que recebe a notificação sobre a alteração ocorrida em um recurso ao qual o dispositivo possui assinatura. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento : /ntfy/{id1}
  - Métodos HTTP: GET/HEAD: Desencorajado; PUT: Erro; POST: Erro; DELETE: Erro;

- **NotificationList** – lista todas as notificações do dispositivo em questão que serão enviadas. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /ntfy
  - Métodos HTTP: GET/HEAD: Desencorajada; PUT: Erro; POST: Obrigatório; DELETE: Erro;

A.1.6 Conjunto de funções Response

Repositório de dados contendo campos de resposta genérico (*e.g.*, data de criação, LFDI e status) que é estendido para conjunto de funções específicos. Por exemplo, ao ocorrer determinado evento (*e.g.*, alteração de preço) e esse ser enviado aos demais dispositivos, pode ser requisitado que os dispositivos alvos do evento enviem uma resposta confirmando que o evento foi recebido. A resposta deverá ser informada (via POST) na URI indicada

no campo *replyTo* (*e.g.*, `replyTo=''{hostname}/rsps''`) presente no evento original. As respostas de um mesmo dispositivo serão diferenciadas uma das outras pela forma de representação, por exemplo, uma resposta a um evento relacionado ao preço será feita pela representação `<PriceResponse>`, já um evento relacionado a resposta de uma mensagem de texto será feito pela representação `<TextResponse>`. O conjunto de funções é composto dos seguintes recursos:

- **ResponseSetList** – lista das instâncias *ResponseSet*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Desencorajado; DELETE: Erro;
- **ResponseSet** – instância específica de *ResponseSet* relacionada a determinado dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Erro; DELETE: Desencorajado;
- **ResponseList** – lista das instâncias *Response* de determinado dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}/rsp
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Obrigatório; DELETE: Erro;
- **Response** – instância de um repositório de dados de resposta genérico que é estendido para conjuntos de funções específicos. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}/rsp/{id2}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Erro; DELETE: Opcional;
- **PriceResponse** – resposta relacionada a mensagem de preço. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}/rsp/{id3}

- Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Erro; DELETE: Opcional;

- **TextResponse** – resposta a uma mensagem de texto. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}/rsp/{id4}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Erro; DELETE: Opcional;

- **DERControlResponse** – resposta a um controle DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}/rsp/{id5}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Erro; DELETE: Opcional;

- **FlowReservationResponse** – resposta a um *FlowReservation*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}/rsp/{id6}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Erro; DELETE: Opcional;

- **DrResponse** – resposta a demanda e controle de carga. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /rsps/{id1}/rsp/{id7}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Erro; POST: Erro; DELETE: Opcional;

A.2 Common Resources

A.2.1 Conjunto de funções Time

É composto por um único recurso denominado *Time*. Ele tem por finalidade fornecer a base de tempo para que todos os dispositivos estejam sincronizados. Dispositivos que implementam esse recurso podem obter a base de tempo de uma fonte externa via NTP ou indiretamente por meio de uma fonte imprecisa (*e.g.*, inserida manualmente via interface

de usuário). Cada fonte de tempo possui uma métrica de qualidade conforme definido no documento `sep.xsd`, sendo a fonte NTP a de melhor qualidade. Sincronização de tempo é necessária para uma correta execução de outras funcionalidades tais como mudança de preço, resposta a demanda e controle de carga e medição. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /tm
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Erro;

A.2.2 Conjunto de funções Device Information

Tem por finalidade fornecer a identificação e as informações de determinado dispositivo que são disponibilizadas pelo fabricante. Por exemplo, por meio desse recurso é possível ter a informação de quem é o fabricante e qual o número de série de determinado dispositivo. O conjunto de funções é composto pelos seguintes recursos:

- **DeviceInformation** – instância que fornece a identificação e as informações do dispositivo disponibilizadas pelo fabricante. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/di
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Opcional;
- **SupportedLocale** – informa uma localidade (região) a qual o dispositivo é compatível. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/di/loc/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Obrigatório;
- **SupportedLocaleList** – lista de localidades para a qual o dispositivo é compatível. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/di/loc/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Obrigatório; DELETE: Erro;

A.2.3 Conjunto de funções Power Status

É composto de um único recurso denominado *PowerStatus*. Ele fornece informações sobre a fonte de alimentação atual do dispositivo, bem como de outras utilizadas como reserva de energia. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /edev/id1/ps
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Erro;

A.2.4 Conjunto de funções Network Status

Fornece informações a respeito da camada de rede (IP) do dispositivo e, potencialmente, da camada de enlace. O conjunto de funções é composto dos seguintes recursos:

- **IPInterface** – informa o status de rede de uma interface específica. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;
- **IPInterfaceList** – lista das instâncias *IPInterface* associadas ao dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;
- **IPAddr** – instância que contém o endereço IP de uma interface específica. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns/{id2}/addr/{id3}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;
- **IPAddrList** – lista de instâncias *IPAddr* de determinada interface. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /edev/{id1}/ns/{id2}/addr
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **RPLInstance** – instância que provê informação sobre o status de uma rede de baixa potência e com perdas (RLP) associada a um determinado *IPAddr*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;

- **RPLInstanceList** – lista as instâncias RPL das quais o *IPAddr* é membro. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **RPLSourceRoutes** – informa uma rota específica RPL. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}/srt/{id5}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;

- **RPLSourceRoutesList** – lista das rotas RPL. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns/{id2}/addr/{id3}/rpl/{id4}/srt
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **LLInterface** – instância de uma camada de enlace específica do dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/ns/{id2}/ll/{id3}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;

- **LLInterfaceList** – lista das camadas de enlace do dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/ns/{id2}/ll
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **Neighbor** – instância que contém atributos específicos de um vizinho que faz uso do IEEE 802.15.4. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/ns/{id2}/ll/{id3}/nbh/{id4}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;

- **NeighborList** – lista dos vizinhos que fazem uso do IEEE 802.15.4. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/ns/{id2}/ll/{id3}/nbh
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

A.2.5 Conjunto de funções Log Event

O conjunto de funções *Log Event* tem por finalidade registrar os eventos gerados pelos dispositivos. Os eventos de registros geralmente são assíncronos e podem ser provenientes de avisos, mensagens de erro e detecções de eventos ou condições não usuais. Por exemplo, se a bateria de um dispositivo está abaixo do limite aceito, isso pode gerar um evento e tal informação ser registrada. O conjunto é composto por dois recursos:

- **LogEvent** – instância com registro de data e hora de um evento significativo detectado pelo dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/lel/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Obrigatório;

- **LogEventList** – lista de instâncias *LogEvent* do dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/lel
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Obrigatório;

A.2.6 Conjunto de funções Configuration

Fornece um acesso centralizado de leitura e gravação da configuração operacional dos dispositivos. É composto dos seguintes recursos:

- **Configuration** – instância que contém os parâmetros de configuração de um dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/cfg
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Erro;

- **PriceResponseCfg** – especifica a forma que os dispositivos devem responder às mudanças de preço. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/cfg/prcfg/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Obrigatório;

- **PriceResponseCfgList** – lista de instâncias *PriceResponseCfg*. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /edev/{id1}/cfg/prcfg
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Obrigatório; DELETE: Erro;

A.2.7 Conjunto de funções File Download

Fornece os mecanismos para realização de suporte remoto seguro, interoperabilidade e *download* de arquivos nos dispositivos. É composto pelos seguintes recursos:

- **File** – instância que contém vários metadados sobre um determinado arquivo em um dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /file/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Desencorajado;
- **FileList** – lista de instâncias *File*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /file
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;
- **FileStatus** – fornece o status das operações de carga e ativação de um arquivo em um dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/fs
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Erro;

A.3 Smart Energy Resources

A.3.1 Conjunto de funções Demand Response and Load Control

Fornece uma interface para o *Demand Response and Load Control*. Dispositivos tais como termostatos e dispositivos que oferecem suporte ao controle de carga fazem uso desse tipo de conjunto de função enquanto HEMS e sistemas de gestão de energia disponibilizam tais conjuntos de funções. O conjunto de função é composto pelos seguintes recursos:

- **DemandResponseProgram** – instância que identifica um programa de resposta a demanda. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /dr/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;
- **DemandResponseProgramList** – lista de instâncias *DemandResponseProgram*. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /dr
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **EndDeviceControl** – fornece parâmetros de controle para instruir um *EndDevice* a executar uma ação específica. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /dr/{id1}/edc/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **EndDeviceControlList** – lista de instâncias *EndDeviceControl*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /dr/{id1}/edc
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **ActiveEndDeviceControlList** – lista de instâncias *EndDeviceControl* que no momento estão ativas. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /dr/{id1}/actedc
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Erro;

- **LoadShedAvailability** – instância que indica o status atual de consumo e a capacidade de rejeitar carga do dispositivo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/lsl/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Obrigatório;

- **LoadShedAvailabilityList** – lista das instância *LoadShedAvailability*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/lsl

- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Obrigatório; DELETE: Erro;

A.3.2 Conjunto de funções Metering

Fornece interfaces para trocar informações sobre medição e leituras das *commodities*. É composto dos seguintes recursos:

- **UsagePointList** – lista de instâncias *UsagePoint*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;
- **UsagePoint** – instância que representa um ponto lógico na rede onde consumo e/ou produção são medidos fisicamente. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;
- **MeterReadingList** – lista de instâncias *MeterReading* com suas respectivas URIs de acesso. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt/{id1}/mr
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;
- **MeterReading** – instância contendo conjunto de valores obtidos dos medidores. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt/{id1}/mr/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;
- **ReadingType** – instância dos tipos de dados fornecidos por uma leitura específica. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /upt/{id1}/mr/{id2}/rt
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Erro;

- **ReadingSetList** – lista de instâncias *ReadingSet*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt/{id1}/mr/{id2}/rs
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **ReadingSet** – instância que contém uma lista de *Reading(s)*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt/{id1}/mr/{id2}/rs/{id3}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;

- **ReadingList** – lista de conjunto de leituras *Reading* de um medidor em específico. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt/{id1}/mr/{id2}/rs/{id3}/r
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **Reading** – instância que contém um valor específico medido por um medidor ou outro ativo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /upt/{id1}/mr/{id2}/rs/{id3}/r/{id4}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;

- **MirrorUsagePointList** – lista de instância *MirrorUsagePoint*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /mup
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Obrigatório; DELETE: Erro;

- **MirrorUsagePoint** – instância que oferece, na forma de espelhamento, suporte idêntico ao *UsagePoint* para outros dispositivos que possuem alguma restrição para comunicação. Por exemplo, um medidor de gás pode ter restrição de energia e com isso ele desperta em intervalos de tempos para enviar suas medições a um dispositivo que ofereça o recurso de espelhamento, o qual não possui restrições de energia. Uma vez que as leituras estão armazenadas no dispositivo que fornece o espelhamento, a concessionária de gás realiza o acesso ao mesmo a qualquer momento tendo em vista que ele não possui nenhuma restrição para a comunicação. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /mup/{id1}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Obrigatório; POST: Obrigatório; DELETE: Obrigatório;

A.3.3 Conjunto de funções Pricing

Fornece modelos de estruturas tarifárias com suporte a uma variedade de tipos tais como: preço fixo, blocos de consumo, preço por hora do dia seguinte e preço em tempo real. O conjunto de funções também oferece suporte a tarifas específicas para aplicações presentes nos dispositivos (*e.g.*, DER) e preços baseados em eventos especiais (*e.g.*, pico de consumo). É composto pelos seguintes recursos:

- **TariffProfileList** – lista de instâncias *TariffProfile*. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /tp
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **TariffProfile** – instância que disponibiliza o código tarifário por meio de uma tabela de cobrança. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /tp/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **RateComponent** – instância que especifica os encargos aplicáveis a um componente da tarifa. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /tp/{id1}/rc/{id2}
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **ActiveTimeTariffIntervalList** – apresenta as instâncias ativas de *TimeTariffInterval* para um *TariffProfile* específico. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /tp/{id1}/rc/{id2}/acttti
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Erro;

- **TimeTariffIntervalList** – lista das instâncias ativas de *TimeTariffInterval* juntamento com o *ConsumptionTariffInterval* associado. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /tp/{id1}/rc/{id2}/tti
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **TimeTariffInterval** – instância que especifica o intervalo de tempo para uso de *RateComponent* específico. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /tp/{id1}/rc/{id2}/tti/{id3}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: erro; DELETE: Desencorajado;

- **ConsumptionTariffIntervalList** – lista de instâncias *ConsumptionTariffInterval*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /tp/{id1}/rc/{id2}/tti/{id3}/cti
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **ConsumptionTariffInterval** – instância que apresenta limites definidos em termos de quantidade de consumo de um serviço ou *commodity* (*e.g.*, eletricidade, gás e água) para a aplicação de uma estrutura tarifária. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /tp/{id1}/rc/{id2}/tti/{id3}/cti/{id4}
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

A.3.4 Conjunto de funções Messaging

Fornece a interface para o serviço de mensagens de texto. Monitores locais e outros dispositivos de exibição de texto são exemplos dos dispositivos que fazem uso desse tipo de conjunto de funções. O conjunto de funções *Response* é usado em conjunto para permitir a confirmação da entrega da mensagem e também para permitir o uso de respostas mais complexas. É compostos pelos seguintes recursos:

- **MessagingProgramList** – lista de instâncias *MessagingProgram*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /msg
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;
- **MessagingProgram** – instância para coleções de mensagens de texto. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /msg/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;
- **ActiveTextMessageList** – lista de mensagens que estão atualmente ativas. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /msg/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Erro;
- **TextMessageList** – lista de instâncias *TextMessage*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /msg/{id1}/txt
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **TextMessage** – instância de mensagem de texto individual usada na notificação. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /msg/{id1}/txt/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

A.3.5 Conjunto de funções Billing

Fornece o consumo, custo, estimativa de consumo futuro e histórico de consumo das mais variadas *commodities*, principalmente energia elétrica. É composto pelos seguintes recursos:

- **CustomerAccountList** – lista de instâncias *CustomerAccount*. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /bill
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **CustomerAccount** – instância de atribuição de um grupo de produtos e serviços a um dispositivo utilizando meio de cobrança e pagamento. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /bill/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **CustomerAgreementList** – lista de instâncias *CustomerAgreement*. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /bill/{id1}/ca
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **CustomerAgreement** – instância de acordo entre o cliente e o fornecedor de serviço/*commodity*. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /bill/{id1}/ca/{id2}
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **ActiveBillingPeriodList** – lista de instâncias ativas de *BillingPeriod*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ca/{id2}/actbp
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Erro;

- **BillingPeriodList** – lista de instâncias *BillingPeriod*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ca/{id2}/bp
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **BillingPeriod** – instância que contém um período de tempo para a cobrança. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ca/{id2}/bp/{id3}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Desencorajado;

- **ProjectionReadingList** – lista de instâncias *ProjectionReading*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ca/{id2}/pro
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **ProjectionReading** – instância que contém valores previstos em uma leitura futura para um intervalo de tempo especificado. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ca/{id2}/pro/{id3}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Desencorajado;

- **BillingReadingSetList** – lista de instâncias *BillingReadingSet*. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /brs

  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **BillingReadingSet** – instância que contém o intervalo de tempo em que ocorreu determinada leitura. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /brs/{id1}

  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **BillingReadingList** – lista de instâncias *BillingReading*. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /brs/{id1}/br

  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **BillingReading** – instância que contém dados capturados em intervalos regulares de tempos. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /brs/{id1}/br/{id2}

  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Desencorajado;

- **TargetReadingList** – lista de instâncias *TargetReading*. O elemento de URI e os métodos HTTP referentes ao recurso são:

  - Elemento URI: /bill/{id1}/ca/{id2}/tar

  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **TargetReading** – instância que contém uma meta ou objetivo de consumo a qual podem ser associadas a incentivos contratuais. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /bill/{id1}/ca/{id2}/tar/{id3}
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **HistoricalReadingList** – lista de instâncias *HistoricalReading*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ca/{id2}/ver
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **HistoricalReading** – instância usada para apresentar leituras que já foram processadas. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ca/{id2}/ver/{id3}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **ServiceSupplier** – instância que contém a empresa que presta serviço ou fornece a *commodity*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /bill/{id1}/ss
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

A.3.6 Conjunto de funções Prepayment

Fornece o mecanismo para a entrega condicionada do serviço ou *commodity* baseado em crédito ou dívida pendente do usuário. É composto pelos seguintes recursos:

- **PrepaymentList** – lista de instâncias *Prepayment*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /ppy
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **Prepayment** – instância que provê informações de pré-pagamento localizadas nos *links* *AccountBalance*, *CreditRegister* e *OperationStatus*. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /ppy/{id1}
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

- **AccountBalance** – instância que contém informações de saldo para o pré-pagamento do serviço/*commodity*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /ppy/{id1}/ab
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Erro;

- **PrepayOperationStatus** – instância que contém informação do status do serviço/*commodity* que está sendo fornecido por meio condicional. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /ppy/{id1}/os
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Erro;

- **ActiveSupplyInterruptionOverrideList** – lista de instâncias *SupplyInterruptionOverride* ativas. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /ppy/{id1}/actsi
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Erro;

- **SupplyInterruptionOverrideList** – lista de instâncias *SupplyInterruptionOverride*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /ppy/{id1}/si
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **SupplyInterruptionOverride** – instância que informa o período de tempo durante o qual o fornecimento do serviço/*commodity* não deve ser interrompido, mesmo que haja problemas de crédito. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /ppy/{id1}/si/{id2}
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Desencorajado; POST: Erro; DELETE: Desencorajado;

- **CreditRegisterList** – lista de instâncias *CreditRegister*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /ppy/{id1}/cr
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **CreditRegister** – instância que contém uma transação para modificação de crédito, seja ela para adição ou decremento. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /ppy/{id1}/cr/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;

A.3.7 Conjunto de funções Flow Reservation

Fornece uma interface para a reserva de eventos de troca de fluxo de energia (*e.g.*, carga ou descarga). Carros elétricos, DER e outros acumuladores de carga são exemplos de dispositivos que utilizam tal conjunto de funções, pois exigem um tempo considerável para completar toda a troca de fluxo energético, seja para carga ou descarga, o que poderia causar uma alta demanda ou sobrecarga na rede. O conjunto de funções é composto pelos seguintes recursos:

- **FlowReservationRequestList** – lista de instâncias *FlowReservationRequest*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/frq
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Obrigatório; DELETE: Erro;

- **FlowReservationRequest** – instância que informa a necessidade de reserva de um período de tempo para realizar uma operação de troca de fluxo. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /edev/{id1}/frq/{id2}
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Opcional;

- **FlowReservationResponseList** – lista de instâncias *FlowReservationResponse*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/frp
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Desencorajado; DELETE: Erro;

- **FlowReservationResponse** – instância de resposta a uma solicitação de reserva de fluxo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/frp/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Desencorajado;

A.3.8 Conjunto de funções Distributed Energy Resources

Fornece uma interface para gerenciar DERs. Dispositivos que geram e armazenam energia são exemplos de dispositivos que utilizam tais conjuntos de funções. O conjunto de funções é composto pelos seguintes recursos:

- **DERList** – lista de instâncias *DER*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **DER** – instância que contém informação sobre a DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Opcional;

- **AssociatedUsagePoint** – informa o *link* do *UsagePoint* associado com determinada DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der/{id2}/upt
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;
- **AssociatedDERProgramList** – lista de instâncias *DERProgram* associadas a determinada DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der/{id2}/derp
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;
- **CurrentDERProgram** – informa o *link* da instância *DERProgram* que está sendo executada pela DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der/{id2}/cdp
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;
- **DERSettings** – instância que contém as configurações da DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der/{id2}/derg
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Erro;
- **DERStatus** – instância que contém informação do status da DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der/{id2}/ders
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Erro;
- **DERAvailability** – instância que contém informações sobre a disponibilidade de produção da DER. O elemento de URI e os métodos HTTP referentes ao recurso são:

- Elemento URI: /edev/{id1}/der/{id2}/dera
- Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Erro;

- **DERCapability** – instância que contém informações sobre a capacidade de produção da DER. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /edev/{id1}/der/{id2}/dercap
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Obrigatório; POST: Erro; DELETE: Erro;

- **DERProgramList** – lista de instâncias *DERProgram*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **DERProgram** – instância que contém informações sobre o programa de distribuição ao qual a DER está associado. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp/{id1}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;

- **ActiveDERControlList** – lista de instâncias ativas *DERControl*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp/{id1}/actderc
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Erro;

- **DERControlList** – lista de instâncias *DERControl*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp/{id1}/derc
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;

- **DERControl** – instância de controle de DERs baseado em eventos/tempo. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp/{id1}/derc/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;
- **DefaultDERControl** – instância a ser considerada como *DERControl* padrão caso nenhum outro seja encontrado. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp/{id1}/dderc
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Opcional; POST: Erro; DELETE: Erro;
- **DERCurveList** – lista de instâncias *DERCurve*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp/{id1}/dc
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Opcional; DELETE: Erro;
- **DERCurve** – instância que especifica um conjunto de um ou mais pontos que são definidos por um variável independente (`x`) e uma variável dependente (`y`). Cada instância contém um parâmetro *curveType* cujo valor define o significado dos pontos conforme descrito no documento `sep.xsd`. Por exemplo, caso o *curveType* seja 3, isso significa que o conjunto de pontos representa uma relação Volt-Watt onde o tesão (Volt) representa o eixo independente (`x`) e a potência (Watt) representa o eixo dependente (`y`). O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /derp/{id1}/dc/{id2}
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Erro; DELETE: Opcional;

A.3.9 Conjunto de funções Metering Mirror

Fornece um mecanismo para dispositivos que tenham restrições (*e.g.*, limitações de bateria) postem seus dados de medição de maneira eficiente em um dispositivo ao qual eles

estão associados. Por exemplo, determinado medidor de gás pode ter limitações de bateria e por isso não fica ativo enviando e recebendo informações o tempo todo. Desse modo, ele "dorme" e "desperta" em intervalos regulares para comunicar-se com o dispositivo ao qual tem um relacionamento *mirror* para troca de informações.

- **MirrorUsagePointList** – lista de instância *MirrorUsagePoint*. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /mup
  - Métodos HTTP: GET/HEAD: Obrigatório; PUT: Erro; POST: Obrigatório; DELETE: Erro;
- **MirrorUsagePoint** – instância que oferece, na forma de espelhamento, suporte idêntico ao *UsagePoint* para outros dispositivos que possuem alguma restrição para comunicação. O elemento de URI e os métodos HTTP referentes ao recurso são:
  - Elemento URI: /mup/{id1}
  - Métodos HTTP: GET/HEAD: Opcional; PUT: Obrigatório; POST: Obrigatório; DELETE: Obrigatório;