
Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

Dissertação de Mestrado

# Identidades Seguras Para o Núcleo 5G: Uma abordagem não intrusiva

Anderson Altair Tomkelski

Campina Grande, Paraíba, Brasil
01/2024

Universidade Federal de Campina Grande
Centro de Engenharia Elétrica e Informática
Coordenação de Pós-Graduação em Ciência da Computação

# Identidades seguras para o núcleo 5G: Uma abordagem não intrusiva

Anderson Altair Tomkelski

Dissertação submetida à Coordenação do Curso de Pós-Graduação em Ciência da Computação da Universidade Federal de Campina Grande - Campus I como parte dos requisitos necessários para obtenção do grau de Mestre em Ciência da Computação.

Área de Concentração: Ciência da Computação
Linha de Pesquisa: Sistemas de Computação

Andrey Elísio Monteiro Brito
(Orientador)
Reinaldo Cézar de Morais Gomes
(Coorientador)

Campina Grande, Paraíba, Brasil

T658i Tomkelski, Anderson Altair.
Identidades seguras para o núcleo 5G: uma abordagem não intrusiva / Anderson Altair Tomkelski. – Campina Grande, 2024.
70 f. : il.

Dissertação (Mestrado em Ciência da Computação) – Universidade Federal de Campina Grande, Centro de Engenharia Elétrica e Informática, 2024.
"Orientação: Prof. Dr. Andrey Elísio Monteiro Brito, Prof. Dr. Reinaldo Cézar de Morais Gomes".
Referências.

1. Segurança da Informação. 2. Confiança Zero. 3. IMT-2020. 4. Núcleo 5G. 5. Sistemas Distribuídos. I. Brito, Andrey Elísio Monteiro. II. Gomes, Reinaldo Cézar de Morais. III. Título.

CDU 004.72:004.56:621.395(043)

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECÁRIA MEIRE EMANUELA DA SILVA MELO CRB-15/568

MINISTÉRIO DA EDUCAÇÃO
**UNIVERSIDADE FEDERAL DE CAMPINA GRANDE**
POS-GRADUACAO EM CIENCIA DA COMPUTACAO
Rua Aprígio Veloso, 882, Edifício Telmo Silva de Araújo, Bloco CG1, - Bairro Universitário, Campina Grande/PB, CEP 58429-900
Telefone: 2101-1122 - (83) 2101-1123 - (83) 2101-1124
Site: http://computacao.ufcg.edu.br - E-mail: secretaria-copin@computacao.ufcg.edu.br / copin@copin.ufcg.edu.br

## FOLHA DE ASSINATURA PARA TESES E DISSERTAÇÕES

**ANDERSON ALTAIR TOMKELSKI**

**IDENTIDADES SEGURAS PARA O NÚCLEO 5G: UMA ABORDAGEM NÃO INTRUSIVA**

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação como pré-requisito para obtenção do título de Mestre em Ciência da Computação.

Aprovada em: 08/01/2024

Prof. Dr. ANDREY ELÍSIO MONTEIRO BRITO, UFCG, Orientador

Prof. Dr. REINALDO CÉZAR DE MORAIS GOMES, UFCG, Orientador

Prof. Dr. FÁBIO JORGE ALMEIDA MORAIS, UFCG, Examinador Interno

Prof. Dr. PAULO DITARSO MACIEL JÚNIOR, IFPB, Examinador Externo

Documento assinado eletronicamente por **ANDREY ELISIO MONTEIRO BRITO**, **PROFESSOR 3 GRAU**, em 09/01/2024, às 09:32, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **FABIO JORGE ALMEIDA MORAIS**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 09/01/2024, às 09:43, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **REINALDO CEZAR DE MORAIS GOMES**, **PROFESSOR(A) DO MAGISTERIO SUPERIOR**, em 09/01/2024, às 11:01, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

Documento assinado eletronicamente por **Paulo Ditarso Maciel Júnior**, **Usuário Externo**, em 09/01/2024, às 13:30, conforme horário oficial de Brasília, com fundamento no art. 8º, caput, da Portaria SEI nº 002, de 25 de outubro de 2018.

A autenticidade deste documento pode ser conferida no site https://sei.ufcg.edu.br/autenticidade, informando o código verificador **4119906** e o código CRC **E7F5CD5E**.

**Referência:** Processo nº 23096.000602/2024-26 SEI nº 4119906

# Resumo

O surgimento de redes 5G permitiu uma revolução nas comunicações móveis, proporcionando maior velocidade de banda, comunicação massiva entre máquinas e aplicações de Internet das Coisas e comunicação altamente confiável de baixa latência. Para alcançar esses objetivos foram necessárias mudanças na arquitetura do 5G, novas tecnologias de rádio, computação de borda e mudanças no núcleo do 5G.

O núcleo do 5G é responsável por gerenciar conexão e mobilidade dos dispositivos, controlar sessão, prover serviços como a banda larga, entre outras funções. Nessa nova geração, o núcleo 5G foi projetado para suportar as tecnologias nativas da nuvem. Uma arquitetura baseada em serviços foi planejada onde cada serviço do núcleo constitui um microsserviço independente. Os serviços se comunicam utilizando uma API REST.

Esse novo design expõe ainda mais os serviços do núcleo da rede 5G. É necessário abordar os riscos que o núcleo 5G está envolvido e procurar mitigá-los para garantir um ecossistema 5G seguro. Segundo a 3GPP, uma recomendação é que a interface base de serviços utilize autenticação mútua na comunicação entre os serviços do núcleo 5G. A autenticação necessita que identidades sejam emitidas e entregues aos serviços que compõem o núcleo 5G e essa necessidade traz alguns desafios.

Neste trabalho é proposto uma abordagem não intrusiva para emitir e distribuir identidades aos serviços do núcleo 5G utilizando o *framework SPIFFE*. Essa abordagem segue os princípios do paradigma de confiança zero. É abordado um estudo de caso utilizando o Free5GC como núcleo de rede 5G, o *SPIRE*, que é uma implementação do *framework SPIFFE*, e o *proxy envoy*. Essas ferramentas são dispostas de forma que não seja necessário fazer alterações no núcleo 5G. Outras implementações de núcleo 5G que sigam os padrões ditados pela 3GPP podem se beneficiar dessa abordagem.

Para a validação do estudo de caso, uma série de testes foi planejada, e os resultados são comparados com uma execução padrão do núcleo 5G que não emprega *mTLS*, tampouco utilizam as ferramentas empregadas.

**Palavras chave:** Segurança da informação, Confiança Zero, IMT-2020, Núcleo 5G, Sistemas distribuídos.

# Abstract

The emergence of 5G networks has revolutionized mobile communications, offering increased bandwidth, massive machine-to-machine communication, Internet of Things applications, and highly reliable low-latency communication. Achieving these goals required a shift in the 5G architecture, new radio technologies, edge computing, and changes in the 5G core.

The 5G core is responsible for managing device connections and mobility, controlling sessions, and providing services such as broadband, among other functions. In this new generation, the 5G core has been designed to support cloud-native technologies. A service-based architecture was planned, where each core service is an independent microservice. Services communicate using a REST API, which serves as the service's basic interface.

This new design, based on cloud-native technologies, further exposes the 5G core services. Addressing the risks associated with the 5G core is crucial, and mitigating these risks is necessary to ensure a secure 5G ecosystem. According to 3GPP recommendations, one approach is to use mutual authentication in the communication between 5G core services. Authentication requires that identities be issued and delivered to the services composing the 5G core, posing some challenges.

This work proposes a non-intrusive approach to issue and distribute identities to 5G core services using the SPIFFE framework. This approach aligns with the principles of the zero-trust paradigm. A case study is presented using Free5GC as the 5G core, SPIRE as an implementation of the SPIFFE framework, and the envoy proxy. These tools are configured in a way that avoids direct modifications to the 5G core, limited to configurations. Other 5G core implementations following 3GPP standards can benefit from this approach.

To validate the case study, a series of tests was planned, and the results are compared with a standard execution of the 5G core that does not employ mTLS nor utilize the employed tools.

**Keywords**: Information Security, Zero Trust, IMT-2020, 5G Core, Distributed Systems.

# Agradecimentos

Agradeço à minha família, em especial à minha mãe, à minha vó e à minha irmã, que me deram todo apoio durante esse processo. Pelo período ausente, peço desculpas.

Aos meus orientadores, Andrey e Reinaldo por todos os ensinamentos, inspiração e toda a disponibilidade que tiveram.

Ao longo do período de mestrado esta pesquisa foi parcialmente financiada pela CAPES e por diferentes projetos: Zero Trust Provisioning and Operations (ZTPO) uma parceria entre a UFCG e a HPE Brasil; Projeto "Ecossistema baseado em IoT para gerência de água e energia" (edital 09/2021, Demanda Universal), uma parceria entre a UFCG e a FAPESQ-PB; e 5G-SEC (Projeto de Estudos sobre Segurança Cibernética em Redes 5G) uma parceria entre a UFCG e a ANATEL.

Sou grato aos meus amigos Ricardo e Adriano por toda ajuda e suporte; a Leonardo e Genildo que me receberam muito bem; aos meus colegas do LSD (Laboratório de Sistemas Distribuídos) pelos momentos vividos e pela cooperação.

A todos, meu sincero muito obrigado.

Invento mais que a solidão me dá

---

Bituca

# Conteúdo

# Lista de Figuras

# Lista de Códigos Fonte

# Capítulo 1

# Introdução

A quinta geração de redes móveis, também conhecida como 5G, surge como uma promessa revolucionária para a comunicação global. A capacidade de oferecer conectividade de alta velocidade, baixa latência e suporte a um número massivo de dispositivos conectados impulsiona uma gama de aplicações, como a Internet das Coisas (IoT — do inglês *Internet of Things*), realidade virtual e aumentada e até mesmo a possibilidade de aplicações para carros autônomos.

O *3rd Generation Partnership Project* (3GPP) é uma colaboração entre diversas organizações de padrões de telecomunicações e desenvolve especificações para sistemas de telecomunicações móveis. O processo de desenvolvimento dessas especificações ocorre através de *releases*. Essas *releases* fornecem uma base sólida para o desenvolvimento do ecossistema 5G, abrangendo desde melhorias nas redes existentes até introdução de novos conceitos.

O 3GPP inicia a regulamentação do 5G a partir da *release* 15 e nela define os componentes fundamentais da Interface Baseada em Serviços (SBI — do inglês *Service Based Interface*) e da Arquitetura Baseada em Serviços (SBA — do inglês *Service Based Architecture*). Esses componentes desempenham um papel fundamental na implementação de serviços baseados em rede e na criação de uma arquitetura flexível e escalável.

A definição de SBI e SBA introduz uma abordagem baseada em serviços no ecossistema do 5G. A SBA é o *framework* necessário para o fornecimento eficiente e dinâmico de serviços no 5G. A SBI são as interfaces padronizadas por onde diferentes componentes da rede se comunicam. A Figura 1.1 mostra a disposição dos serviços na arquitetura baseada

em serviço. A linha vermelha é a interface baseada em serviços que é uma interface criada a partir da exposição dos serviços de uma função de rede (NF — do inglês *Network Function*).

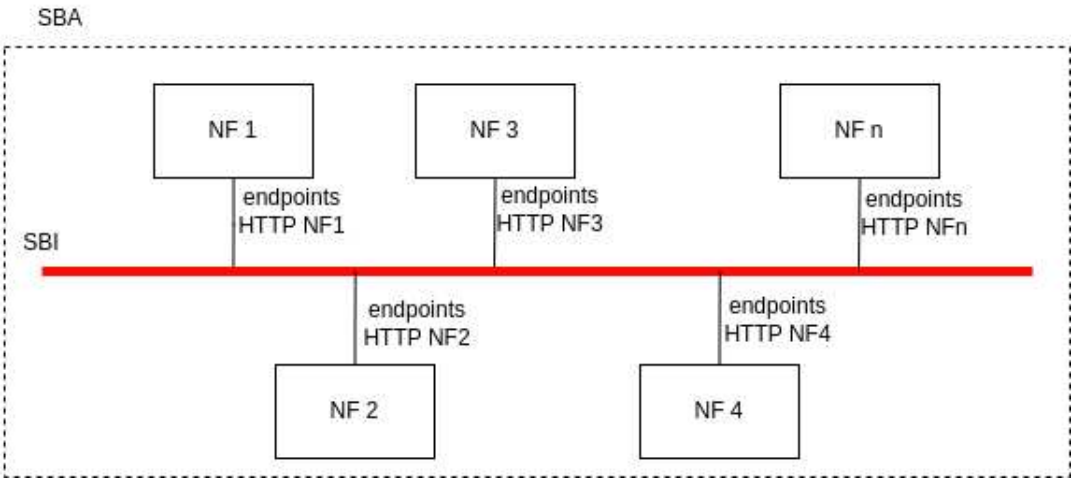


Figura 1.1: SBA e SBI

À medida que a dependência dos serviços de rede continua a crescer em ambientes domésticos, industriais e em meio ao surgimento de novas e diversas aplicações, bem como a expansão da superfície de ataque, a garantia de confidencialidade, integridade e autenticidade das comunicações para o núcleo 5G torna-se uma prioridade crítica. A identidade digital desempenha um papel fundamental para garantir esses requisitos, sendo de suma importância explorar soluções eficazes para a distribuição e gerenciamento seguros de identidades dentro do ecossistema 5G.

Um conceito que deve ser levado em consideração é o de confiança zero que se baseia na premissa que não se deve confiar automaticamente em nenhuma entidade, mesmo as que estejam dentro da rede, sendo necessário verificar e autenticar todas as solicitações de acesso que ocorrem na rede independentemente da origem. Essa abordagem assume que ameaças podem existir tanto interna quanto externamente à rede, sendo necessário adotar um modelo de segurança mais granular e cuidadoso [24].

## 1.1 Problema

A partir das definições de SBA os serviços são capazes de executar de maneira independente a sua funcionalidade específica, consumindo informações de outros serviços a partir da SBI.

A necessidade de um serviço consultar outro depende da operação requisitada, essa operação pode ser uma autenticação, estabelecimento de uma sessão, interceptação de tráfego, entre outros.

A Figura 1.2 apresenta uma visão geral da cooperação entre os serviços do núcleo 5G para estabelecer conexão de um equipamento de usuário à internet. O equipamento de usuário estabelece a comunicação com a antena no passo 1, que encaminha a requisição de autenticação para o núcleo 5G a partir de uma interface de acesso no passo 2. No momento 3 a autenticação do equipamento de usuário é realizada pelos serviços do núcleo responsáveis pela autenticação. Em caso de sucesso, no passo 3, a sessão é estabelecida no passo 4 e um serviço de ancoragem é escolhido no passo 5. Após isso há um túnel indicado pelo número 6 por onde o equipamento pode acessar a internet.

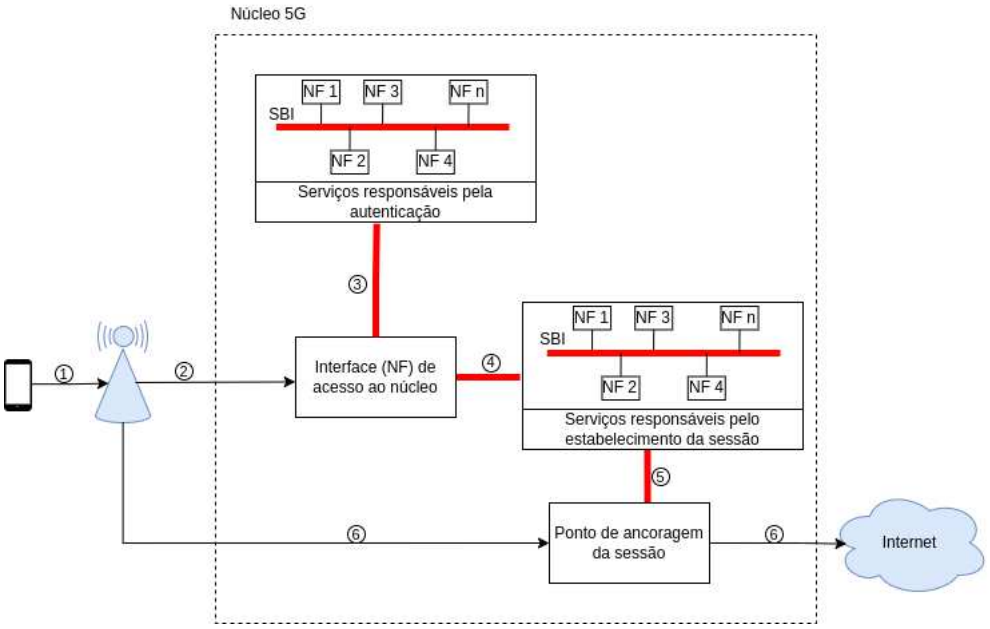


Figura 1.2: Visão geral do processo de autenticação

Nas *releases* da 3GPP são apresentadas especificações técnicas referentes ao desenvolvimento de redes 5G. Na *release* 15 no documento TS 33.501 [2] é apresentado uma seção que descreve os requisitos para registro, descoberta e autorização de serviço. Há também uma seção sobre comunicação direta e indireta entre Funções de Rede (NF — do inglês *Network Function*) e Função de Repositório de Rede (NRF — do inglês *Network Repository Function*), onde é orientado o uso de autenticação mútua entre NF e NRF e todas as

funções de rede devem suportar o protocolo de comunicação *TLS* (do inglês *Transport Layer Security*). Porém, o documento ressalta que a distribuição de chaves para o *TLS* é de responsabilidade do operador de segurança e que não está no escopo do documento.[1]

A distribuição de certificados e identidades de forma dependente de um administrador que tem total controle sobre a geração e distribuição dos certificados apresenta alguns desafios significativos, especialmente em ambientes distribuídos e heterogêneos, como a complexidade de escalar e gerenciar os certificados para um grande número de dispositivos, serviços e usuários. A dependência da rotatividade de chaves e certificados pode ser um problema quando esses artefatos são gerenciados por um administrador ou de uma forma não automatizada.

Como essas especificações técnicas dependem do modo como a implementação é feita, pode ainda não haver suporte a *mTLS* (do inglês *Mutual Transport Layer Security*), como é o caso dos projetos de código aberto *Free5GC* e *Open5GS*. Casos como esses dependem do suporte da comunidade ou das pessoas interessadas em aplicar esse protocolo ao projeto. Ambos os projetos suportam os protocolos HTTP e HTTPS para a comunicação entre os serviços, implementando a funcionalidade definida com a SBI, porém os certificados usados quando a comunicação HTTPS é configurada deve ser forjado e gerenciado pelo administrador da rede.

Um conjunto de ataques à API REST apresentados por Coldwell et al. [8] mostrou que a falta de autenticação da API permite que um atacante utilize as rotas da SBI. Dessa forma é possível fazer consultas sobre serviços da rede, cadastrar novos serviços no NRF e até mesmo deletar serviços. Esses ataques levam em consideração a falta de configuração do núcleo da rede, o que permite facilmente esse acesso não privilegiado às rotas do SBI. Este problema de autenticidade está diretamente relacionado à configuração de confiança estabelecido entre os serviços de rede do núcleo 5G.

[1] Mais especificamente, o documento aborda o assunto da seguinte forma [2]: "The structure of the PKI used for the certificates is out of scope of the present document. The identities in the end entity certificates shall be used for authentication and policy checks. The key distribution of pre-shared keys for *TLS* is up to the operator's security policy and out of scope of the present document."

## 1.2 Objetivo

Seguindo a recomendação da 3GPP, é possível mitigar problemas de segurança relacionados ao SBI adotando o protocolo de comunicação *mTLS*. O *mTLS* proporciona autenticação mútua entre o cliente e o servidor, utilizando certificados digitais durante o *handshake* do *TLS*. Ao implantar o *mTLS* na comunicação do núcleo 5G é possível fortalecer a segurança, garantindo a autenticidade de todos os serviços que compõem o núcleo 5G e a integridade das mensagens trocadas. Essa implantação afeta apenas a comunicação entre os serviços do núcleo, que compõem a SBA, não envolvendo a autenticação do UE com o núcleo 5G, essa autenticação está fora do escopo deste trabalho.

Neste trabalho, temos como objetivo propor uma abordagem não intrusiva para implantação da comunicação *mTLS* no núcleo 5G. Para realizar essa comunicação *mTLS* não intrusiva, utilizamos um *proxy envoy* para cada serviço que compõe o núcleo 5G. A distribuição dos certificados, que são usados para averiguar autenticidade dos *proxies* e, consequentemente, dos serviços que eles protegem, é de responsabilidade do *Secure Production Identity Framework for Everyone (SPIFFE)*.

Este trabalho apresenta um caso de uso que proporciona a entrega dos certificados para serviços de rede do núcleo 5G por meio de um *proxy envoy* intermediário, estabelecendo, dessa forma, uma comunicação *mTLS* segura. Além disso, os experimentos propostos têm como objetivo encontrar limitações de desempenho e segurança, que permitirão o planejamento de pesquisas futuras na área.

## 1.3 Estrutura do documento

Este trabalho está estruturado da seguinte forma. A seção 2 apresentará a fundamentação teórica, destacando os principais componentes envolvidos no desenvolvimento deste trabalho. A Seção 3 detalhará a metodologia adotada nesta pesquisa, descrevendo os procedimentos e ferramentas utilizados para conduzir os experimentos e a análise. Será apresentada em detalhes a arquitetura proposta e os mecanismos de autenticação e autorização implantados. A Seção 4 trará a avaliação dos resultados obtidos a partir da implantação da solução proposta. Finalmente, a seção 5 concluirá esta dissertação destacando as princi-

pais contribuições alcançadas, as limitações identificadas e as perspectivas para pesquisas futuras.

# Capítulo 2

# Fundamentação Teórica

## 2.1 Redes Móveis de Quinta Geração

A rede móvel de quinta geração (5G) tem como objetivo prover alta taxa de transferência, baixa latência, alta densidade e maior faixa de mobilidade sem comprometer a confiabilidade. A revolução dessa tecnologia permitirá a existência de cenários de ambientes IoT, cidades inteligentes, computação tátil, entre outros.

Para a disponibilização de uma rede móvel existem diversas tecnologias envolvidas. O dispositivo que o usuário usa, a tecnologia de rádio com a qual o dispositivo se comunica com a antena, o núcleo que funciona como o último elo para a conexão com borda ou Internet.

O núcleo do 5G assume diversos papéis, agindo na autenticação do dispositivo e gerenciamento de políticas. Ou seja, o que o usuário pode acessar ou quanto ele pode acessar, permitindo auditoria de pacotes e o acesso à Internet e os serviços de borda se disponíveis. É necessário que o núcleo garanta também mobilidade, de modo que o dispositivo não perca o acesso enquanto se movimenta por diferentes regiões. Para resolver os diversos problemas relacionados a esses papéis, uma nova arquitetura baseada em serviços para o núcleo é proposta.

### 2.1.1 Arquitetura Baseada em Serviços

A arquitetura do núcleo 5G permite que os serviços se comuniquem através de interfaces conectadas através de uma arquitetura baseada em serviços (SBA) em comum. Os serviços

que compõem a rede podem se comportar como consumidores e como produtores. A partir da SBA, um serviço consumidor é capaz de localizar e interagir com serviços produtores expostos. As funcionalidades suportadas por uma função de rede são disponibilizadas e acessíveis através de uma API [21].

A arquitetura necessita que os serviços consumidores sejam capazes de selecionar uma instância de serviço produtor e determinar seu endereço. Isso é possível a partir de um repositório de serviços que mantém registro de todas as funções de rede instanciadas disponíveis e os serviços expostos por essas instâncias. Esses registros são mantidos de forma dinâmica pelas funções de rede produtoras que se registram no repositório [21].

Quando o serviço de rede consumidor descobre as instâncias do serviço de rede produtor, a escolha da instância é feita a partir dos critérios de serviços desejados, considerando características como capacidade e disponibilidade da instância.

A comunicação entre os serviços ocorre através de uma API RESTful com protocolo HTTP2, baseada no modelo de requisições e respostas, ou no modelo de inscrição e notificação [18]. Essa interface para a comunicação entre os serviços é a chamada SBI [16].

### 2.1.2 Registro e descoberta de serviços

Quando dois serviços se comunicam na arquitetura SBA, eles podem possuir dois papéis distintos. A função de rede que envia a requisição assume o papel de consumidor. A função de rede que oferece o serviço e dispara ações baseadas nas requisições assume o papel de produtor.

Para que um consumidor possa localizar e enviar mensagens para um produtor é necessário que os serviços possuam alguma forma de descobrir uns aos outros. A descoberta de serviço garante que todos os produtores mantenham registro dos serviços que eles oferecem. Dessa maneira, é necessário que um serviço produtor seja configurado para encontrar apenas o serviço de repositório e não precisando conhecer nenhum outro serviço da rede a partir da sua configuração inicial [21].

Todo o serviço que é iniciado precisa fazer o registro junto ao repositório. Nessa primeira etapa, o serviço age como consumidor e o repositório como produtor. O repositório oferece o serviço de registro de serviço para o serviço consumidor.

Vamos considerar um exemplo prático que envolve os serviços Função de Repositório da

Rede (*Network Repository Function* — NRF), Função de Gestão de Acesso e Mobilidade (*Access and Mobility Management Function* — AMF) e Função de Controle de Política (*Policy Control Function* — PCF). A Figura 2.1 mostra o fluxo que ocorre desde o registro do serviço PCF realizado à NRF, o serviço AMF fazendo uma requisição para descoberta de serviço e enfim, a requisição do serviço PCF pela AMF [21].

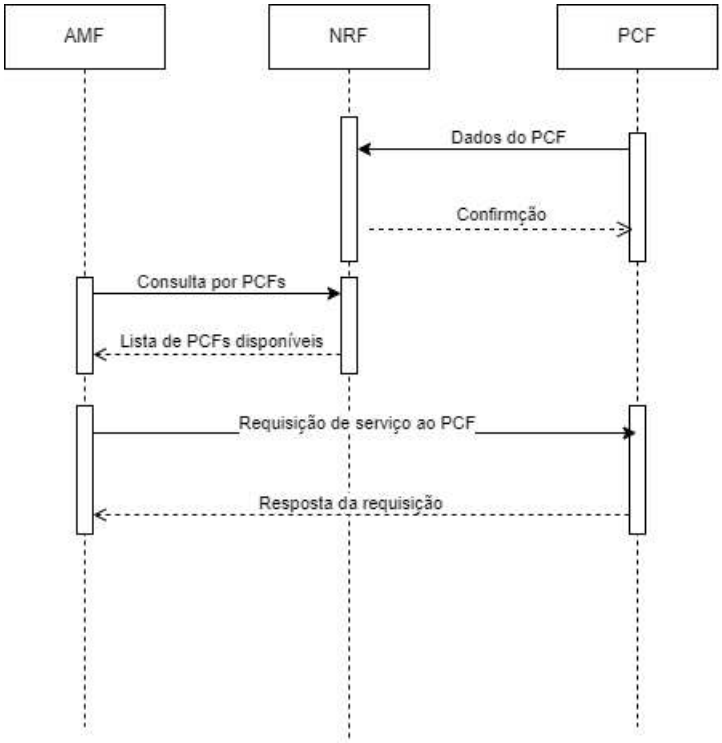


Figura 2.1: Registro de NF — Adaptado de Rommer et al. 2019.

A PCF inicia fazendo uma requisição do tipo *PUT* para a NRF, que inclui informações sobre a PCF, como os serviços disponíveis, o endereço e a identidade. A NRF verifica a validade da PCF e armazena as informações referentes ao serviço. A partir desse momento a PCF se torna disponível para outras funções da rede através de consulta à NRF.

A AMF precisa utilizar um serviço da PCF e, para isso, faz uma consulta *GET* à NRF requisitando uma lista de PCFs que disponibilizam o serviço necessário. Essa etapa é chamada de descoberta de serviço, onde a AMF age como consumidor e a NRF como produtor.

Quando a etapa de descoberta é concluída, a AMF está apta a se comunicar com a PCF fazendo requisições de serviço. Nesta etapa a AMF age como consumidor e a PCF age como produtor.

### 2.1.3 Núcleo da rede 5G

Para desempenhar as diferentes tarefas das quais é responsável o núcleo 5G possui diversas funções de rede que se comunicam através de diferentes interfaces, com diferentes protocolos trocando as mais diversas informações. Além disso, é possível que essas funções de rede sejam executadas em diferentes máquinas, distribuídas geograficamente, ou ainda, em diferentes ambientes, como por exemplo contêineres e máquinas virtuais. A distribuição por si só pode levar a mais pontos de entrada para atacantes, sendo necessário garantir que a nuvem e a distribuição desse sistema esteja segura. A forma como os serviços se comunicam também pode ser alvo de ataque e deve ser investigada. É muito comum falhas de segurança em rotas acontecerem por falta de autenticação, criptografia, ou até mesmo após o domínio de uma máquina legítima por um atacante.

De acordo com o sistema de categorias para pontos fracos e vulnerabilidades de hardware e software CWE (do inglês — *Common Weakness Enumeration*) é possível observar uma tendência consistentemente ascendente de falhas relacionadas a falta de autenticação. De acordo com a comunidade isso pode indicar a falta de implementação consistente de técnicas relacionadas a autenticação por parte dos desenvolvedores de software [9].

O núcleo 5G é descrito em um série de especificações pela 3GPP a partir da *release* 15 (Rel-15). Esses padrões e recomendações podem ser implementados de diferentes formas, por diferentes companhias ou até mesmo no formato de código aberto. Existem alguns projetos de código aberto que implementam o núcleo do 5G, como é o caso do *Free5GC* [2]. Mais informações sobre o *Free5GC* podem ser vistas no Apêndice A.

O principal objetivo do núcleo da rede é estabelecer sessões de uma maneira segura e encaminhar os dados do usuário provendo conexão. Somado à rede de acesso via rádio, obtém-se a rede móvel de telecomunicações. A seguir são detalhadas as principais funções de rede que compõem o núcleo 5G.

A Função de Gestão de Acesso e Mobilidade (*Access and Mobility Management Function* — AMF) interage com a rede de rádio e dispositivos móveis. Suporta comunicação criptografada permitindo que os dispositivos se registrem, sejam autenticados e mudem geograficamente entre diferentes células de rádio. A AMF também permite encontrar e ativar dispositivos que estão em modo de espera. A AMF não faz gerenciamento da sessão dos dispositivos conectados no núcleo 5G, assim como não gerencia a autenticação. Serviços

específicos são utilizados para essas funcionalidades e a AMF apenas encaminha os pacotes do dispositivo para esses serviços conforme demandado [5, 21].

A Função de Gerenciamento de Sessão (*Session Management Function* — SMF) é a responsável por gerenciar as sessões dos dispositivos. Responsável por estabelecer, modificar, liberar sessões e atribuir endereços IPs para os dispositivos de acordo com cada sessão. A SMF se comunica indiretamente com os dispositivos através da AMF. Também é responsável por selecionar e controlar as UPFs, que inclui a configuração da direção e aplicação de tráfego para cada sessão [5, 21].

A principal tarefa da Função do Plano de Usuário (*User Plane Function* — UPF) é processar e encaminhar os dados do usuário. A funcionalidade da UPF é controlada pela SMF. A UPF conecta com IP externos e age como um ponto de ancoragem de IP estável para os dispositivos em redes externas, ocultando a mobilidade. A UPF realiza diversos tipos de processamento nos dados encaminhados, gera relatórios de uso para a SMF e pode aplicar inspeção de pacotes, analisando o conteúdo dos pacotes encaminhados. Também executa políticas de usuário como bloqueio, redirecionamento ou controle de acesso, o que possibilita o uso de tecnologias como computação em borda, por exemplo. Adicionalmente, pode aplicar qualidade de serviço (*Quality-of-Service* — QoS) para marcar pacotes com alta prioridade em caso de congestionamento da rede [5, 21].

O Repositório de Dados Unificado (*Unified Data Repository* — UDR) é o banco de dados que armazena diversas informações como dados de assinaturas, dados que definem vários tipos de rede ou políticas de usuário [21].

O Gerenciamento Unificado de Dados (*Unified Data Management* — UDM) age como uma interface para os dados de assinatura do usuário armazenados no UDR e executa diversas funções solicitadas pela AMF. É responsável por gerar dados de autenticação usados para autenticar dispositivos [5, 21].

A Função do Servidor de Autenticação (*Authentication Server Function* — AUSF) provê o serviço de autenticação para dispositivos. Esse processo utiliza as credenciais de autenticação geradas pelo UDM. Também é responsável por gerar material criptográfico para permitir atualização segura de informações *roaming* e outros parâmetros no dispositivo [5, 21].

As funções de rede adicionais que estendem a capacidade do Core são: PCF (*Policy*

*Control Function*), NSSF (*Network Slice Selection Function*), N3IWF (*Non-3GPP Inter Working Function*), AF (*Application Function*), NEF (*Network Exposure Function*), SMSF (*Short Message Service Function*), LMF (*Location Management Function*), entre outras.

## 2.2 Protocolos de comunicação

### 2.2.1 Transport Layer Security — TLS

O primeiro padrão para comunicação criptografada entre clientes e servidor foi desenvolvido pela Netscape. A primeira versão do *Secure Sockets Layer* (SSL) foi amplamente implementado porém não foi lançada publicamente. A versão publicamente lançada foi o SSL 2.0 que continha falhas severas de segurança. A terceira versão do SSL, o SSL 3.0, foi renomeada para *TLS 1.0* e apoiada pela Força Tarefa de Engenharia de Internet (*Internet Engineering Task Force — IETF*).

O *TLS* opera entre duas entidades, nomeadas cliente e servidor, sobre a camada de transporte TCP, garantindo confidencialidade, integridade e autenticidade dos dados transmitidos. O *TLS* é composto por uma série de etapas.

A etapa de *handshake* envolve a troca de mensagens entre o cliente e o servidor para iniciar a comunicação criptografada. A primeira mensagem é enviada pelo cliente com a versão do *TLS* suportada, algoritmos de chave pública, entre outros parâmetros. O servidor retorna a mensagem indicando a versão de *TLS* selecionada, o algoritmo de criptografia e de chave pública a serem utilizados e fornece um certificado digital contendo a chave pública do servidor.

Na etapa de autenticação e troca de chaves, o cliente verifica a autenticidade do certificado recebido. Se necessário, verifica a cadeia de certificação até uma autoridade certificadora confiável. O cliente gera uma chave de sessão e criptografa essa chave com a chave pública do servidor para enviar ao servidor. Com sua chave privada o servidor consegue descriptografar a chave de seção gerada pelo cliente.

Por fim, o cliente envia uma mensagem ao servidor indicando que as mensagens futuras serão criptografadas com a chave de sessão compartilhada [20].

Uma extensão do protocolo *TLS* é o *mTLS*, que garante a autenticação mútua entre as duas

entidades envolvidas, cliente e servidor. Tanto o cliente quanto o servidor possuem chaves privadas e certificados. Dessa forma, na etapa de *handshake*, ambos trocam certificados e realizam verificação de autenticidade [7].

## 2.3 Modelo Confiança Zero

Dado os riscos relacionados às tecnologias de informação e comunicação, é importante construir uma infraestrutura confiável para minimizá-los. Segundo a ITU, confiança pode ser vista como a expectativa de algo cumprir uma tarefa de uma maneira esperada, cumprindo o propósito pretendido [26]. O ETSI define segurança como o estado de proteção, bem como as medidas aplicadas para alcançar, manter e validar a proteção [19].

O documento ETSI GS NFV-SEC 003 V1.1.1 [19] define relações de segurança para e entre as NFs. Para que o conceito de confiança seja construído, componentes-chave são a autoridade certificadora e o gerenciamento de identidades e acesso. Esses componentes são essenciais para ações e funções entre as NFs, de maneira criptografada, assinada e segura. Entretanto, é necessário que as chaves e identidades sejam assinadas ou pelo menos validadas por um terceiro confiável, nesse caso a autoridade certificadora.

A ETSI reconhece a necessidade de NFs possuírem uma cadeia de confiança inicial, chamada autoridade certificadora (Certificate Authority — CA), para que uma entidade possa confiar em outra entidade. Também cita que o estabelecimento de uma CA, como cadeia de confiança inicial, leva ao problema conhecido como problema da tartaruga de baixo. Um problema circular onde a CA também depende de uma cadeia de confiança inicial, outra CA [19].

O modelo confiança zero é um conjunto de conceitos e ideias que busca adotar uma postura de verificação contínua das interações entre os componentes de uma rede. Em vez de confiar cegamente na integridade e segurança dos dispositivos e usuários, o modelo de confiança zero exige uma validação constante, garantindo que cada interação seja autenticada e autorizada antes de ser considerada confiável. O paradigma de confiança zero leva em consideração que a rede pode estar comprometida internamente e que mecanismos de defesa para o tráfego externo já não são mais válidos [25].

Uma arquitetura de confiança zero diz respeito ao plano utilizado por uma organização

para implementar os conceitos do modelo de confiança zero, definindo como os componentes se relacionam, o fluxo de trabalho e políticas de acesso.

O foco é na autenticação, autorização e encolhimento das zonas de confiança implícitas, garantindo disponibilidade e reduzindo atrasos nos processos de autenticação. As regras de acessos são criadas de forma mais específica possível, de modo que menos privilégios sejam necessários para realizar ações.

A implementação do modelo de confiança zero pode variar entre as organizações, mas existem dois componentes principais que compõem o modelo: Ponto de Aplicação de Políticas de Confiança Zero (*Policy Enforcement Point* — PEP) e Mecanismo de Confiança Zero (*Policy Engine* — PE). O PEP é responsável por permitir a comunicação entre um participante da rede e um recurso, encaminhando o tráfego para o mecanismo de confiança zero, além de monitorar e gerenciar registros do tráfego. O mecanismo de confiança zero, por sua vez, decide se permite o acesso com base nas informações da requisição. Um conjunto de princípios e recomendações propõem que o PEP seja implantado o mais próximo do recurso possível para reduzir o tamanho da área considerada implicitamente segura.

O Ponto de Decisão de Política (*Policy Decision Point* — PDP) pode ser subdivido em mecanismo de política e administrador de política (*Policy Administrator* — PA). O mecanismo de política é o componente responsável pela decisão final, garantindo acesso ou não ao recurso e gerando registros sobre a decisão. A decisão é baseada em políticas da organização e artefatos de entrada externos. O administrador de política, por sua vez, confia na decisão do mecanismo de política. Em caso de sucesso, o administrador de política pode gerar sessões ou credenciais que garantam o acesso ao recurso, configurando o ponto de aplicação da política para permitir o início da sessão. Em caso de falha na decisão, o ponto de aplicação da política é sinalizado para derrubar a conexão [24].

O ponto de aplicação da política é responsável por habilitar, monitorar e terminar eventualmente as conexões aos recursos da organização. O PEP se comunica com o administrador de política (PA) para encaminhar as requisições e para receber atualizações das políticas [24].

Os acessos permitidos podem depender do nível de autorização que o serviço ou usuário possui. A verificação não deve depender de artefatos únicos como senha, mas sim considerar múltiplos fatores como senha, localização, dispositivo utilizado pelo usuário para acessar um recurso interno e privilégios. Uma vez que for verificado e validado o acesso, é necessário

que toda a comunicação seja criptografada e proteções de integridade sejam fornecidas [22].

A adoção de um modelo de confiança zero traz diversos benefícios, incluindo melhor proteção e controle dos recursos e dados, além da capacidade de descartar tráfego inútil imediatamente, evitando congestionamento na rede. Com o controle do tráfego também é possível monitorar e melhorar a segurança com base no histórico obtido ao longo do tempo.

## 2.4 SPIFFE/SPIRE

Em ambientes distribuídos, como microsserviços e ambientes em nuvem, a gestão de identidades e a segurança da rede se tornam mais complexas. Por serem ambientes altamente dinâmicos, o uso de abordagens como credenciais estáticas ou autenticação centralizada apresentam desvantagens.

Uma identidade deve representar o propósito lógico específico do serviço e uma associação estabelecida com uma autoridade ou raiz de confiança. Uma vez que todos os serviços de uma rede recebem uma identidade, elas podem ser usadas para que os serviços se autentiquem, autorizem acesso e transmitam informações de forma segura.

Documentos de identidade digitais como X.509 são usados como identidades de serviços e podem ser verificados usando técnicas criptográficas. Uma técnica comumente usada é a Infraestrutura de Chave Pública (*Public Key Infrastructure* — PKI). A PKI define um conjunto de regras, políticas e procedimentos necessários para criar, gerenciar, armazenar, distribuir e revogar certificados digitais, além de gerenciar chaves de criptografia públicas. Identidades digitais podem ser validadas localmente com um conjunto de pacotes de raiz de confiança [17].

O *Secure Production Identity Framework For Everyone — SPIFFE*, surge com o objetivo de fornecer identidades seguras e exclusivas independente da localização, melhorar a autenticação e a autorização em ambientes distribuídos, oferecer gerenciamento e emissão de identidades de forma federada e descentralizada, ser compatível e integrável com as tecnologias criptográficas consolidadas, bem como fornecer identidades independente da infraestrutura subjacente [10].

O *SPIFFE* é um conjunto de padrões que define interfaces e documentos necessários para identificação de cargas de trabalho (executáveis em geral) em ambientes dinâmicos e

heterogêneos, de maneira agnóstica. O *SPIFFE* consiste em cinco partes.

- *SPIFFE ID*: A representação única do nome de uma carga de trabalho, a identidade propriamente dita. É composto por um prefixo "spiffe://", seguido pelo nome do domínio de confiança e o nome do carga de trabalho específico. Um *SPIFFE ID* possível é "spiffe://organizacao.com/servico1". O domínio de confiança é usado para gerenciar limites administrativos e de segurança dentro e entre organizações. Cada domínio de confiança possui uma autoridade emissora diferente.

- *SPIFFE Verifiable Identity Document (SVID)*: O documento de identidade criptograficamente verificável usado pelo serviço para provar sua identidade a outros serviços. O *SVID* possui um *SPIFFE ID* e é assinado por uma autoridade emissora que representa o domínio de confiança ao qual a carga de trabalho pertence. O *SPIFFE* suporta atualmente dois tipos de documento de identidade para o *SVID* que são o X.509 e o *JWT*.

- *SPIFFE API* de carga de trabalho: A interface local utilizada pelas cargas de trabalho para recuperar suas identidades, pacotes de confiança e informações necessárias para averiguar a autenticidade das identidades de outros serviços. A *SPIFFE API* de carga de trabalho é disponibilizada como um serviço *gRPC* e usa um fluxo bidirecional, permitindo que atualizações sejam enviadas às cargas de trabalho conforme necessário. Ela também entrega *SVIDs* e pacotes de confiança às cargas de trabalho e os rotaciona de acordo com a necessidade.

- *SPIFFE Trust Bundle*: A coleção de chaves públicas em uso por uma autoridade emissora *SPIFFE*. Cada domínio de confiança possui um pacote associado a ele e esse pacote é usado para validar os *SVIDs* que afirmam pertencer a esse domínio de confiança ou federados. A coleção de chaves públicas pode ser compartilhada publicamente por não possuir nenhum segredo, porém o pacote está sujeito a modificações em seu conteúdo, o que requer que a distribuição ocorra de maneira segura de forma a garantir integridade.

- *SPIFFE Federation*: Meio pelo qual os *SPIFFE Trust Bundle* podem ser compartilhados entre diferentes domínios de confiança.

O *SPIRE* (do inglês, *SPIFFE Runtime Environment*) é uma implementação de código aberto das cinco partes do *SPIFFE*. O *SPIRE* possui dois componentes principais, o *SPIRE Server* e o *SPIRE Agent*. O *SPIRE Server* é responsável por autenticar os *SPIRE Agents* e por forjar os *SVIDs*. O *SPIRE Agent*, por sua vez, é responsável por disponibilizar a *SPIFFE API* de carga de trabalho para as cargas de trabalho. Esses dois componentes são concebidos em uma arquitetura orientada a *plugin* que permite fácil extensão e adaptação para diferentes configurações e plataformas [10].

### 2.4.1 SPIRE Server

O *SPIRE Server* gerencia e entrega todas as identidades de um domínio de confiança *SPIFFE*. Ele armazena informações sobre seus *SPIRE Agents* e cargas de trabalho. O *SPIRE Server* conhece as cargas de trabalho que ele gerencia a partir de registros de entrada, que são regras flexíveis usadas para atribuir *SPIFFE IDs* a nós e cargas de trabalho. O armazenamento do *SPIRE Server* é usado para guardar informações sobre esses registros de entrada e também o status dos *SVIDs* forjados e entregues.

Todos os certificados em um domínio de confiança são assinados pelo *SPIRE Server*. Por padrão, o *SPIRE Server* gera um certificado auto assinado, que posteriormente é usado para assinar *SVIDs*. É possível usar diferentes *plugins* para obter esse certificado assinado a partir de outra autoridade certificadora. A parte do *SPIRE Server* responsável pelas assinaturas dos certificados é chamada *Upstream Certificate Authority* [10].

### 2.4.2 SPIRE Agent

A função do *SPIRE Agent* é disponibilizar a API para as cargas de trabalho. Ele determina a identidade das cargas de trabalho e se introduz ao *SPIRE Server*. O *SPIRE Agent* recebe informações sobre o domínio de confiança e as cargas de trabalho que possivelmente farão chamadas à API de carga de trabalho. Quando novos registros de entrada sobre cargas de trabalho são criados no *SPIRE Server*, essas informações são propagadas para os *SPIRE Agents*.

O *SPIRE Agent* usa a identidade obtida durante a atestação de nó para se autenticar com o *SPIRE Server* e receber os *SVIDs* para as cargas de trabalho que ele é responsável por

gerenciar. O *SPIRE Agent* é responsável por renovar os pacotes de confiança e os *SVIDs*, de acordo com o tempo de vida que eles possuem, e atualizar as cargas de trabalho [10].

### 2.4.3 Atestação

Atestação é o processo no qual as cargas de trabalho e o ambiente ao qual elas pertencem são descobertos, no fim desse processo é possível provar a identidade de uma carga de trabalho usando informações disponíveis como evidencias. No *SPIRE* existem dois tipos de atestação, a atestação de nó e a atestação de carga de trabalho. A atestação de nó trata dos atributos que descrevem os nós e a atestação de carga de trabalho trata dos atributos que descrevem as cargas de trabalho. Esses atributos são conhecidos como seletores.

A atestação de nó inicia quando o *SPIRE Agent* executa pela primeira vez. O *SPIRE Agent* se comunica com o *SPIRE Server* e troca informações sobre o local onde o Agent está executando. Para executar essa tarefa, um *plugin* específico de plataforma é usado. Esse *plugin* é executado tanto no lado do *SPIRE Agent* quanto no lado do *SPIRE Server*. As informações são coletadas e enviadas para o *SPIRE Server*, o *SPIRE Server* verifica a validade dessas informações de acordo com a definição do *plugin* e os seletores relacionados ao nó [10].

Como exemplo, um *SPIRE Agent* executando em uma máquina virtual da *AWS* se comunica com o *SPIRE Server* iniciando a atestação. O *SPIRE Agent* envia as informações coletadas na máquina virtual para o *SPIRE Server*. O *SPIRE Server* é capaz de verificar essas informações fazendo requisições para a API da AWS. O sucesso da atestação de nó resulta na entrega de identidade ao *SPIRE Agent*. Essa identidade é utilizada para todas as comunicações posteriores [23].

A atestação de carga de trabalho determina a identidade do serviço. A atestação ocorre quando uma carga de trabalho estabelece uma conexão com a API. Quando essa conexão é estabelecida, o *SPIRE Agent* utiliza as funcionalidades do sistema operacional para determinar qual processo abriu essa conexão. A atestação ocorre através de *plugins* para atestação de cargas de trabalho, que coleta informações adicionais sobre o serviço e retorna para o *SPIRE Agent* como seletores. O *SPIRE Agent* determina a identidade da carga de trabalho comparando os seletores descobertos com o registro de entrada, entregando o *SVID* correspondente ao serviço.

O registro de entrada é o meio pelo qual são cadastrados as cargas de trabalho que são permitidas no ambiente, definindo quais serviços são esperadas e quais serão seus respectivos *SPIFFE IDs*. Cada registro de entrada possui três atributos principais, o *parent ID*, que indica onde uma carga de trabalho deve executar, o *SPIFFE ID* e as informações necessárias para identificar a carga de trabalho. O registro de entrada pode descrever nós ou cargas de trabalho, o *parent ID* do registro de entrada da carga de trabalho faz referência ao registro de entrada do nó. O *parent ID* do registo de entrada dos nós é o *SPIFFE ID* do *SPIRE Server* [10].

### 2.4.4 Modelo de Ameaça do SPIFFE/SPIRE

O modelo de ameaça do *SPIFFE/SPIRE* assume que a comunicação de rede é hostil ou completamente comprometida. Também assume que o hardware onde os componentes executam e os próprios componentes são confiáveis. As partes terceiras envolvidas na atestação, como os serviços de nuvem ou tecnologias nos quais os nós e cargas de trabalho executam, também são considerados confiáveis.

Os limites de segurança considerados por Feldman et al. [10] envolvem a comunicação entre os componentes cargas de trabalho e *agents*, entre *agents* e *servers*, e entre *servers* em diferentes domínios de confiança. As cargas de trabalho não são confiáveis bem como os *servers* de outros domínios de confiança e a comunicação da rede.

O limite entre carga de trabalho e *agent* define que o *agent* não confia em nenhuma informação enviada pela carga de trabalho ou qualquer informação que possa ser manipulada pela carga de trabalho. Toda a afirmação tomada pelo *agent* em relação a identidade da carga de trabalho deve ser feita de maneira independente.

Os *agents* são considerados mais confiáveis que as cargas de trabalho e menos confiáveis que os *servers*. Para limitar o impacto de um nó *Agent* comprometido o *SPIRE* apenas entrega as informações necessárias para o *Agent* resolver a sua tarefa. A partir do *parent ID* e dos registros de entrada, o *Agent* recebe apenas as identidades das cargas de trabalho que devem executar no nó ao qual ele pertence.

O limite entre *servers* em diferentes domínios de confiança define que os *servers* apenas podem emitir *SVIDs* de seus respectivos domínios de confiança. Dessa maneira as chaves públicas trocadas durante a federação correspondem apenas ao domínio de confiança de

onde essas chaves partiram. Esse limite se aplica apenas à federação, todo o *SPIRE Server* no mesmo domínio de confiança tem acesso as chaves de assinatura com as quais podem forjar *SVIDs* do domínio de segurança ao qual faz parte.

## 2.5 Envoy Proxy

O *envoy* é um *proxy* de código aberto projetado para arquiteturas baseadas em serviço. É uma aplicação que é executada junto ao processo alvo, oferecendo uma comunicação *mesh* no qual cada aplicação envia e recebe suas requisições a partir da interface local, sem necessariamente ter qualquer conhecimento a respeito da topologia da rede.

O *envoy* oferece uma série de benefícios e recursos. O balanceamento de carga permite distribuir o tráfego de entrada entre vários serviços. Permite que serviços se registrem dinamicamente e sejam descobertos pelo *envoy*. Possui roteamento flexível baseado em regras, permitindo que as solicitações de entrada sejam direcionadas a determinados serviços tendo como critérios URL, cabeçalhos HTTP ou metadados. Possui recursos de segurança incluindo suporte à *TLS* e *mTLS*, autenticação de serviços e filtragem de conteúdo. O *envoy* também coleta métricas detalhas sobre o tráfego.

A solicitação recebida pelo *proxy envoy* passa por vários filtros configuráveis, permitindo a transformação e roteamento do tráfego. O *envoy* pode lidar com diversos protocolos, como HTTP, gRPC, entre outros. A comunicação com o serviço ao qual o *envoy* está relacionado é persistente, o que aumenta o desempenho e reduz a latência [12].

## 2.6 Trabalhos Relacionados

No trabalho de Vasoukolaei, Sattar e Matrawy [28], identificam a recomendação da 3GPP de fazer uso do protocolo *TLS* entre diferentes NFs no núcleo 5G. Os autores propõem um estudo para verificar o custo de operações criptográficas do *TLS* como latência, sobrecarga do tamanho do pacote e o número de mensagens adicionadas em comparação a comunicação sem *TLS*. Também é analisado o impacto na rede do núcleo 5G ao utilizar diferentes cifras de criptografia disponíveis no *TLS*.

No trabalho de Xinxin Hu et al. [15], os autores apresentam uma análise de segurança so-

bre o protocolo HTTP/2 usado no núcleo 5G. O protocolo HTTP/2, por ser largamente usado em ambientes da internet sem relações com telecomunicações, permite que a arquitetura base de serviços seja implantada usando as tecnologias nativas de nuvem e da web, aumentando a velocidade de implantação e integração de novos componentes ou a atualização desses componentes. Como o objetivo do 5G é expor o núcleo para outras verticais, como a Internet das Coisas massiva e veículos autônomos, é necessário que o protocolo de comunicação não seja de uso específico para redes de telecomunicações móvel, como nas gerações anteriores.

Os autores discutem os tipos de ataques *Stream Reuse Attack*, *Flow Control DoS*, *Dependency cycle DoS*, *HPACK Bomb*, *Man-in-the-Middle Attack* e *Interconnect Attacks*, analisando como esses ataques podem afetar as redes 5G. Os ataques podem prejudicar o funcionamento da rede, sobrecarregando as funções da rede e causando negação de serviço. O ataque *Man-in-the-Middle* pode permitir ao atacante forjar funções de rede e controlar o tráfego, mesmo que o SBI utilize *TLS*. É possível que o atacante forje certificados e se passe por uma função de rede legítima, nesse caso é importante observar como a criação e distribuição de certificados ocorre para evitar esse tipo de ataque [15].

O estudo de Syed et al. [25] apresenta uma discussão aprofundada do estado da arte das técnicas de autenticação e controle de acesso buscando encontrar opções disponíveis que sigam os princípios da confiança zero. Cada componente da arquitetura de confiança zero é analisada para verificar se a existência de técnicas é suficiente para a realização de confiança zero em infraestruturas críticas. Os autores abordam autenticação, controle de acesso, criptografia, segmentação e perímetro baseados em software.

Mecanismos de autenticação convencionais possuem limitações ou até mesmo são vulneráveis. Além disso, se torna cada vez mais necessário que esses mecanismos de autenticação sejam leves e escaláveis para permitir confiança para todos os recursos organizacionais, inclusive distribuídos ou limitados. Várias técnicas de autenticação são observadas por Syed et al. [25], como autenticação contínua, autenticação consciente baseada em alguma informação momentânea e autenticação de dispositivos.

A autenticação de dispositivos leva em consideração serviços ou dispositivos que muitas vezes não possuem interação humana. Esse tipo de autenticação faz necessária a utilização de identidades únicas para esses recursos. Uma possível categoria de identificação apontada pela pesquisa é através de *gateways* que não possuem modificações regulares. A Infraestru-

tura de chave Pública (PKI) tem sido largamente adotada para prover identidades únicas e, nesse cenário, se torna necessário o gerenciamento de certificados [25].

A discussão sobre microssegmentação e perímetro definido por software também apresenta a utilização de *gateways* e *firewalls* para desempenhar o papel do Ponto de Aplicação de Política (PEP), aplicando as políticas definidas no Mecanismo de Política (PE). A microssegmentação define medidas de segurança e onde elas serão aplicadas na rede, o princípio fundamental é a aplicação de políticas de segurança próximas do recurso a ser protegido. Os autores também apontam a importância das tecnologias de Redes Definidas por Software (SDN) e Virtualização de Funções de Rede (NFV), inclusive para redes 5G, e a possibilidade da utilização de políticas de segurança dinâmicas e funções de segurança de rede virtual para aplicar controle de acesso.

No trabalho de Buyakar, Tulja Vamshi Kiran et al. [6], os autores propõem reduzir a latência das funções de rede pelo uso do protocolo *gRPC* no lugar de API REST HTTP e uma configuração distribuída no serviço de rede NRF para registro e descoberta de serviços usando o projeto de código aberto Consul. Os autores propõem também o uso de um *look-aside load balancer* para garantir alta escalabilidade e baixa latência, como alternativa para um balanceador de carga baseado em *proxy*.

Os autores Benzaïd, Tabel e Farooqi [4] reforçam que tradicionalmente os mecanismos de criptografia e assinatura digital são utilizados para garantir proteção de integridade de dados. Porém, um desafio encontrado nesses mecanismos é a dependência de confiança em terceiros para geração de chaves. Um gerador de chaves comprometido poderá comprometer toda a infraestrutura. Os autores esclarecem o conceito de confiança para redes 5G e emergentes, apontando também a necessidade de uma gestão eficaz de confiança nos componentes do ecossistema 5G, implicando na identificação de indicadores-chave de desempenho e métricas para cada uma das entidades envolvidas.

Um *framework* de integridade de dados baseado em *blockchain* é proposto, com o objetivo de garantir a integridade de dados derivados de componentes de rede, como funções de rede virtuais, para o uso em modelos de aprendizagem de máquina. Os dados são associados a IDs na *blockchain* e a integridade pode ser verificada antes do uso por algoritmos de aprendizagem de máquina.

Uma arquitetura de confiança zero em duas camadas é proposta por Feng, Zhou et

al. [13], com o objetivo de avaliar a confiança do equipamento de usuário e das aplicações de borda oferecidas pela arquitetura de Computação de Borda de Acesso Múltiplo (do inglês, *Multi-Access Edge Computing* — MEC). Na abordagem proposta, as entidades de confiança zero são introduzida na rede 5G MEC. Essas entidades são responsáveis por fazer a avaliação do equipamento de usuário em tempo real e tomar decisões de confiança baseadas no resultados dessas avaliações. Com essa abordagem é possível monitorar em tempo real o comportamento do usuário e mitigar vulnerabilidades relacionadas ao acesso indevido às aplicações MEC.

Para Bello, Hussein et al. [3], a literatura oferece muitos estudos relacionados a comunicação entre equipamentos de usuários e o núcleo da rede a partir da rede de acesso de rádio (RAN). Porém, poucos estudos abordam a segurança da comunicação dentro do núcleo da rede. Entretanto é necessário observar com mais cautela essa parte do ecossistema de redes móveis, principalmente com a tendência crescente da implantação dos componentes do núcleo da rede em ambientes de nuvem e maior exposição na Internet. Os autores argumentam a necessidade de construir *fameworks* de segurança baseados no conceito de confiança zero para endereçar problemas como autenticação mútua, confidencialidade do plano de dados, proteção de integridade e privacidade nessas redes móveis.

Na arquitetura proposta, as funções de rede são colocadas atrás de *gateways*, responsáveis por proteger as funções de rede de ataques externos. Essa configuração garante que a comunicação entre as funções de rede apenas ocorrerá mediante o uso de um certificado. A validade desses certificados é verificada a partir de consultas ao sistema de segurança chamado Perímetro Definido por Software (SDP). A validação dos certificados resulta no estabelecimento de uma comunicação *mTLS* entre as funções de rede. No modelo proposto pelos autores, toda a comunicação é baseada nas regras definidas no SDP e os certificados gerados por ele. A configuração do SDP depende de um administrador e não é claro a política de rotatividade desses certificados.

No trabalho de Duong, Van-Binh, e Younghan Kim [11], os autores propõem um serviço *mesh* para o núcleo 5G. A arquitetura proposta executa cada serviço de rede em um contêiner junto com um *proxy envoy*. Para gerenciar a comunicação entre os componentes é utilizado o *Istio*, que gerencia as políticas utilizadas pelo *envoy* para aplicar decisões sobre o tráfego que entra e sai de cada serviço. Essa comunicação ocorre utilizando *mTLS*. O Cilium é

usado como interface de rede de contêiner (CNI) para conectar todos os *proxies* que agem como *sidecars*. Uma limitação com relação a utilização do *Istio* é a falta de flexibilidade relacionada a atestação de cargas de trabalho, que pode ser resolvida com a utilização do *SPIRE* integrada ao *Istio*.

# Capítulo 3

# Metodologia

Para alcançar uma abordagem não intrusiva, isto é, sem modificação no código fonte do núcleo 5G disponível para nós, é necessário pensar em uma abordagem que permita que um outro *software*, que possa ser acoplado às funções de rede, seja utilizado. Esse software será responsável por receber identidades através do *SPIFFE* e fazer uma tradução da comunicação entre as funções de rede para utilizar *mTLS*. Nossa abordagem utiliza *proxies* para cumprir esse papel, adicionados como uma camada externa que gerencia a segurança da comunicação, sem exigir alterações no código interno do núcleo 5G. Dessa forma, a estrutura existente do núcleo 5G permanece intacta, e as funcionalidades de segurança necessárias para o *mTLS* são adicionadas de maneira independente por meio dos *proxies*.

Um caso de uso é proposto utilizando o *proxy envoy*, o *SPIRE* e o *Free5GC* como núcleo 5G. Ao integrar o *envoy* e o *SPIRE* ao *Free5GC*, busca-se implementar uma solução eficiente para a implementação do *mTLS*, permitindo autenticação mútua entre os componentes do sistema, garantindo confidencialidade e um padrão para a entrega e gerenciamento de identidades e certificados. A abordagem busca permitir que outros projetos de núcleo 5G também possam ser configurados da mesma forma, desde que sigam os padrões definidos a partir *release* 15 da 3GPP [2], como é feito no projeto *Free5GC*.

## 3.1 Núcleo 5G com mTLS

Sabendo que os serviços do núcleo 5G se comunicam através de uma interface padrão utilizando API REST, queremos garantir que essa comunicação seja criptografada e além disso

que ocorra autenticação em todos os serviços do núcleo.

Para garantir que a comunicação seja criptografada basta que a API REST passe a utilizar o protocolo HTTPS, porém isso não garantiria atestação mútua. Ao utilizar protocolos criptografados, outro problema que surge é o armazenamento, a distribuição e a renovação dos certificados.

Levando em consideração projetos de núcleo 5G que não implementam o protocolo *mTLS*, pode ser útil a aplicação de *proxies* como o *envoy* ou *Ghostunnel*. Desse modo, as aplicações alvo não precisam ser alteradas, elas continuam se comunicando utilizando seus protocolos padrões, porém passam a se comunicar com o *proxy*. A Figura 3.1 mostra como a comunicação ocorre quando os *proxies* são adicionados. O serviço A deseja se comunicar com o serviço B e envia a mensagem utilizando o protocolo HTTP para o *proxy* A. O *proxy* A deve transmitir a mensagem ao serviço B, porém o serviço B não se comunica utilizando *mTLS*. O *proxy* A deve enviar a mensagem ao *proxy* B para que faça a tradução para o serviço B.

Para a escolha do *proxy* a ser utilizado na abordagem proposta foi analisada a facilidade em configurar a integração entre os *proxies* e os serviços de rede. Além do requisito de possuir a comunicação com o *SPIRE* desenvolvida, é necessário que o *proxy* consiga recuperar os certificados através da *workload API* e utiliza-los para estabelecer comunicação. O *Ghostunnel* precisa de uma instância para cada rota de comunicação estabelecida, o que o torna inviável. O *envoy*, por sua vez, consegue criar um mapeamento de destino baseando-se em informações como a URL do método HTTP. Dessa forma, é necessário apenas um *envoy* por serviço possuindo o mapeamento de rotas que o serviço correspondente poderá usar.

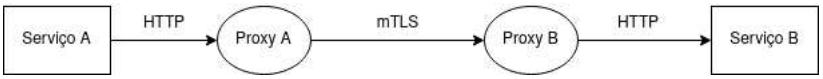


Figura 3.1: Fluxo do proxy — Fonte: Do autor

## 3.2 Distribuição de identidades

É necessário garantir a provisão de chaves para que a comunicação *mTLS* entre os *proxies* funcione. Gerar as chaves manualmente utilizando ferramentas e empregar autoridades cer-

tificadoras externas é uma alternativa, porém necessita diversas garantias como armazenamento seguro das chaves, renovação periódica, revogação.

Uma ferramenta que pode contribuir para facilitar o cenário de gerenciamento das chaves e certificados é o *SPIRE*. O *proxy* pode ser configurado para recuperar e utilizar os certificados disponibilizados pelo *SPIRE*. Nesse caso o *SPIRE* atua como uma autoridade certificadora, gerando chaves privadas e certificados e fornecendo-os aos *proxies*. Essas chaves são usadas para autenticar a identidade do *proxy* e criptografar a comunicação *mTLS*. Quando um serviço precisa se comunicar com outro ocorre a troca de certificados, verificação da validade desses certificados e decisão do algoritmo de criptografia que será utilizado na comunicação. Nessa etapa da autenticação mútua, os certificados recebidos pelos *proxies* podem ser verificados na cadeia de certificados até a autoridade certificadora do *SPIRE*.

Na Figura 3.2 é apresentado o cenário em que os *proxies* estabelecem uma comunicação com o *SPIRE* para o recebimento de identidades. O processo de atestação ocorre para verificar se o *proxy* é confiável, baseado nas informações coletadas pelo *SPIRE*. O *proxy* de um serviço, o serviço e o *SPIRE Agent*, que entrega o certificado ao *proxy*, sempre executam no mesmo local. Um *SPIRE Agent* pode entregar certificados para mais de um *proxy* desde que seja executado na mesma máquina que o *SPIRE Agent*, como mostrado na Figura 3.3.

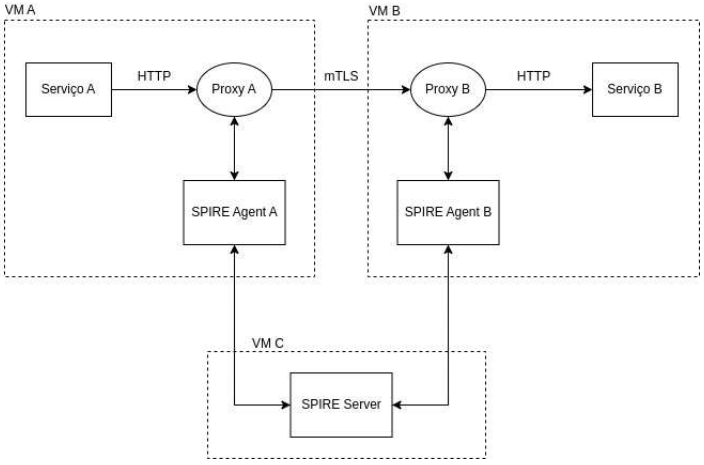


Figura 3.2: Fluxo do proxy utilizando SPIRE — Fonte: Do autor

O processo para a disponibilização das identidades inicia quando o *SPIRE Server* é iniciado. O *SPIRE Server* gera um certificado autoassinado, que será utilizado para assinar os *SVIDs* para todas as cargas de trabalho que estão no domínio de confiança desse *SPIRE*

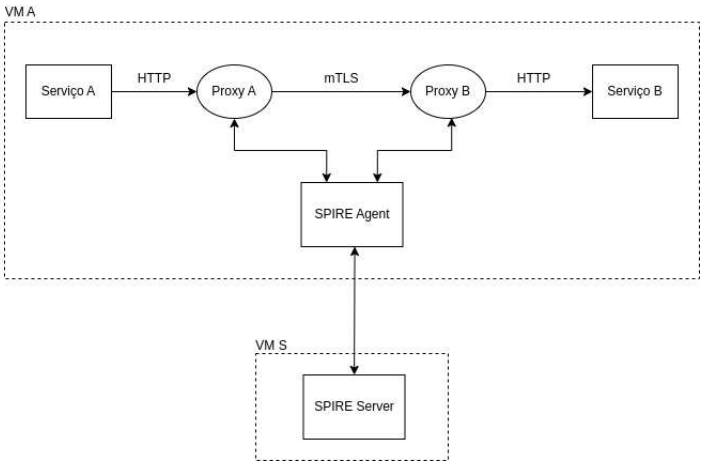


Figura 3.3: Dois serviços para um SPIRE Agent — Fonte: Do autor

*Server*. Na primeira execução é criado também o *trust bundle*, utilizado no momento da verificação da validade dos *SVIDs* entre as cargas de trabalho. A partir desse momento é possível registrar cargas de trabalho através da API de registro do *SPIRE Server*.

O *SPIRE Agent* é executado na mesma máquina em que a carga de trabalho está executando. O processo de atestação remota é realizado entre o *SPIRE Agent* e o *SPIRE Server* para provar ao último a identidade do nó onde o primeiro está executando. Nessa etapa, os *plugins* de atestação de nó são utilizados. O *SPIRE Agent* coletará as informações necessárias para a validação do *SPIRE Server*. O sucesso da atestação de nó resulta na entrega de um *SVID* para a instância do *SPIRE Agent*.

O *SPIRE Agent* requisita ao *SPIRE Server* os registros de entrada que ele é responsável por manter, ou seja, aqueles registros de carga de trabalho que possuem como *parent ID* essa instância do *SPIRE Agent*. O *SPIRE Agent* faz a requisição de assinatura dos *SVIDs* das cargas de trabalho ao *SPIRE Server* e as armazena em *cache*. Após esse processo o *SPIRE Agent* começa a esperar requisições de certificados de carga de trabalho através da *workload API*.

Quando a carga de trabalho inicia, um *proxy* no caso da figura 3.2, o *SPIRE Agent* realiza o processo de atestação da carga de trabalho utilizando os *plugins* de atestação de carga de trabalho. O *SPIRE Agent* compara as informações coletadas pelos *plugins* de atestação de carga de trabalho com as informações presentes nos registros de entrada. Se houver um casamento, o *SVID*, já em *cache*, é entregue ao *proxy*.

## 3.3 Caso de Uso

Nesta sessão é apresentada a forma como foi realizada a integração do *Free5GC* com o *SPIRE*, cujo objetivo é proporcionar uma integração não intrusiva, ou seja, sem alterações no código fonte do *Free5GC*. Para controlar a comunicação entre as funções de rede do *Free5GC* foi usado o *proxy envoy*. Os *proxies* têm se mostrado alternativas frutíferas para autenticação no paradigma de confiança zero.

### 3.3.1 Configuração dos NFs

Cada serviço do *Free5GC* possui um arquivo YAML que define os parâmetros relacionados à identificação, à localização e à autenticação do NF. Isso inclui informações como os endereços de IP, portas que o NF estará escutando, tipo do protocolo utilizado, nome dos *endpoints* que o NF possui, informações relacionadas à telecomunicação móvel como código de área, entre outros. Nesse arquivo YAML é indicado o endereço do NRF e esse é o único serviço de rede conhecido pelos NFs inicialmente.

A configuração inicial do serviço AMF pode ser visto no Código B.1 do Apêndice B. Essa configuração é muito semelhante para as funções de serviço AUSF, NSSF, PCF, SMF, UDM, UDR e PCF. Todos os serviços de rede configuram a interface base de serviços (SBI) da mesma maneira, com seus respectivos IPs, possuindo a lista de nomes de serviço correspondente e o endereço do NRF.

Entre as linhas 1 e 3, o arquivo possui informações gerais do serviço. A partir da linha 4 são listadas as configurações iniciais do serviço. A linha 6 indica o IP que a gNB usará para se comunicar com o AMF. A partir da linha 8 é configurada a interface base de serviço, indicando o tipo de protocolo que será usado na linha 9, qual IP será registrado no NRF na linha 10, qual IP que o serviço estará usando na linha 11 e a porta na linha 12. A diferença entre IP de registro e IP usado de fato pode ser útil em alguns cenários com o kubernetes, onde o que será registrado é um nome de nó. Entre as linhas 13 e 18 o serviço indica quais serão os *endpoints* que serão registrados, ou seja, quais funções esse serviço irá atender. Na linha 20 é indicado o endereço do NRF, único serviço conhecido pelo AMF desde o início.

### 3.3.2 Integração do Envoy com SPIRE

Para que um serviço receba um *SVID* é necessário faça uma requisição para o a API de carga de trabalho do *SPIRE Agent* que está executando no mesmo local que o serviço. Os serviços do *Free5GC* não possuem nativamente essa funcionalidade. Para uma abordagem não intrusiva, é necessário o uso de um *proxy* que se comunique com o *SPIRE Agent*. Nessa abordagem o *proxy envoy* é o responsável por receber o *SVID* e trocar as mensagens entre as funções de rede. O *envoy* funciona como um tradutor que recebe a mensagem do NF usando o protocolo HTTP e transmite no protocolo *mTLS*.

O *envoy* é configurado a partir de um arquivo YAML. Nesse arquivo são indicados os *listeners* e *clusters* do *envoy*. Os *listeners* são os pontos de extremidade onde o *envoy* aceita conexão de serviços, monitorando e aceitando conexões de entrada, gerenciando a comunicação entre os serviços externos e o *envoy*. Cada *listener* está associado a um endereço de IP e porta específicos utilizados para receber as requisições. Os *clusters* representam conjuntos de instâncias de serviços de destino, que o *envoy* utiliza para rotear as solicitações para esses serviços de destino. A configuração de *clusters* mapeia o endereço e porta do serviço alvo, bem como o protocolo utilizado para enviar essas informações para o serviço.

É necessário configurar a comunicação do *envoy* com o *SPIRE Agent* para que possa requisitar o seu certificado *SVID* através da API de carga de trabalho. Essa configuração é feita indicando um *cluster* que aponta para o *socket* disponibilizado pelo *SPIRE Agent*. Esse *socket* é o meio para a comunicação com a *workload API*. O Código C.1 do Apêndice C mostra a configuração do *cluster* responsável por permitir ao *envoy* se comunicar com a *workload API*.

Para receber o certificado *SVID* é necessário fazer uma configuração SDS como mostrado no Código C.2 do Apêndice C. Essa configuração é feita no *listener* se a rota de acesso ao *envoy* depender da validação do certificado *SVID* do serviço externo, ou seja, o *envoy* apenas aceitará a comunicação se o serviço externo possuir um *SVID* válido. Quando o *envoy* deseja estabelecer uma comunicação com um serviço externo que está esperando a apresentação de um certificado *SVID*, essa configuração também ocorrerá no *cluster*. Desse modo é possível configurar o *envoy* para apenas estabelecer comunicação com serviços que estão no mesmo domínio de confiança do *SPIRE Server* e que possuam *SVIDs* válidos. A Figura 3.4 mostra

a sequência do processo de atestação entre o *envoy* o *SPIRE Agent* para recuperar o *SVID*.

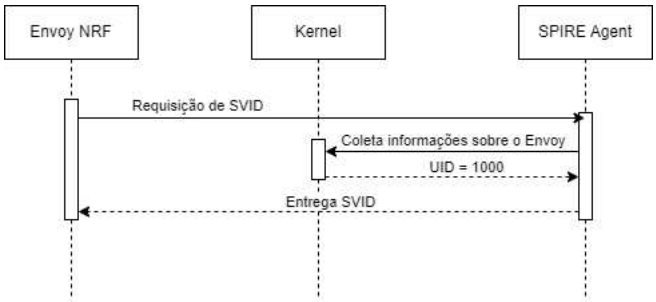


Figura 3.4: NRF integrado ao SPIRE

O único *listener* e *cluster* configurados sem a validação do *SVID* são os utilizados pelo serviço ao qual o *envoy* está associado. Por exemplo, o *envoy* estabelece comunicação com o NRF via HTTP sem essa validação, porém com os demais *envoys* que representam os outros serviços a comunicação ocorre através do protocolo *mTLS*.

### 3.3.3 Integração do Envoy com NRF

O primeiro NF a executar é o NRF, responsável pelo armazenamento de informações sobre os serviços disponíveis na rede e descoberta dessas informações pelos serviços, coordenando e orquestrando os serviços no núcleo 5G.

Os outros serviços do núcleo registram suas informações junto ao NRF durante suas respectivas inicializações. Essas informações incluem o nome do serviço, o tipo, a identidade e a localização. Entre as informações armazenadas está qual tipo de serviço que pode acessar cada *endpoint*, prioridade e capacidade. O armazenamento dos *endpoints* é essencial para permitir que os outros serviços da rede consultem o NRF e encontrem as informações necessárias para se comunicar com o serviço desejado.

Quando um serviço precisa se comunicar com outro serviço na rede, ele faz uma solicitação ao NRF para descobrir o *endpoint* do serviço desejado. O NRF recebe a solicitação de descoberta e consulta seu registro para obter as informações do serviço solicitado. O *endpoint* do serviço solicitado é retornado para o solicitante, permitindo que a comunicação entre os serviços seja estabelecida.

Todos os NFs agem como produtor e consumidor, com exceção do NRF que é um NF

que atua apenas como produtor, ou seja, não consome informações de outros NFs. No caso do NRF apenas se faz necessário que o *envoy* receba as mensagens e reencaminhe ao NRF no IP e porta configurado no arquivo YAML do NRF. A configuração do NRF pode ser vista no Código D.1 do Apêndice D, com dois *endpoints* mostrados nas linhas 16 e 17.

A integração do *envoy* com o NRF é particularmente diferente da integração do *envoy* com os demais NFs, uma vez que o NRF age apenas como servidor. Nesse caso, é necessário configurar somente um *listener* no *envoy* e o *cluster* do NRF. O NRF receberá dados exclusivamente provenientes do seu *envoy*. Por sua vez, o *envoy* do NRF apenas receberá dados provenientes de outros *envoys* que possuem *SVIDs*.

O NRF está usando o IP 127.0.0.10 e a porta 8000 para receber as mensagens. O *envoy* está disponível no IP 127.0.1.10 e toda mensagem recebida que tem como destino os *endpoints* do NRF é encaminhada para o serviço. Esse padrão se repetirá nesse trabalho, o NF que estiver configurado com o IP 127.0.0.x terá um *proxy envoy* configurado para escutar com o IP 127.0.1.x e a porta utilizada na interface SBI sempre é a 8000.

O *listener* do *envoy* do NRF é mostrado no Código D.2 do Apêndice D. Entre as linhas 21 e 30 é configurado qual será o *cluster* alvo para cada requisição. No *envoy* do NRF toda a comunicação será enviada para o *cluster* com o nome nrf que representa a instância do serviço.

A configuração dos *clusters* do *envoy* do NRF pode ser observada no Código D.3 do Apêndice D. Nessa etapa são configuradas as informações de onde o NRF está executando. Quando a requisição para o NRF é recebida pelo *listener* do envoy, ela é encaminhada para esse *cluster* e, a partir do endereço e porta definidos, é encaminhada para o NRF.

A configuração do *envoy* para os demais NFs precisa ser feita nos dois sentidos. Como cada *envoy* é responsável pela comunicação de um NF, todo tráfego que parte do NF precisa ser encaminhado ao seu *envoy* e todo tráfego que tem como destino um NF precisa ser recebido pelo seu *envoy* correspondente.

### 3.3.4 Integração do Envoy com NFs

A configuração dos *envoys* para os outros NFs é semelhante ao apresentado para o *envoy* do NRF quando a requisição está sendo recebida. Quando a requisição está sendo enviada é preciso conhecer o destino dela. O *envoy* é configurado para saber o local de cada serviço.

Usando o *endpoint* que o NF está consultando é possível filtrar a comunicação e decidir qual será o *cluster* que receberá a informação. O *cluster* será o *envoy* do serviço de destino.

Vamos utilizar como exemplo a configuração do *envoy* do AMF. A configuração de rota pode ser visualizada no Código D.4 do Apêndice D. Quando o *listener* do *envoy* recebe a requisição do AMF ele redireciona essa comunicação para algum dos *clusters* configurados. Esse redirecionamento depende do *endpoint* contido na requisição. No Código D.4 do Apêndice D, por exemplo, se o *endoint* for *"/nnrf"* então o redirecionamento será para o *cluster* "nrf"e assim por diante.

A configuração dos *clusters* do *envoy* do AMF pode ser vista no Código D.5 do Apêndice D. É necessário utilizar os certificados do *SPIRE* quando o AMF envia mensagens para outros NFs através dos *envoys*, como é o caso do *cluster* do NRF e do AUSF, mostrado respectivamente nas linhas 15 e 56. Quando a mensagem é proveniente de algum NF com destino ao AMF, ela é recebida pelo *envoy* e os certificados são validados. Se houver sucesso nessa validação a mensagem é encaminhada para o AMF usando o protocolo HTTP sem novas validações. Esse padrão ocorre para todos os serviços do *Free5GC*.

### 3.3.5 Redirecionamento para o Envoy

Quando um NF inicia, com exceção do NRF, ele realiza o processo de registro junto ao NRF usando o endereço indicado no arquivo de configuração YAML. A solução para que a interação ocorra através do *envoy* seria indicar o endereço do *envoy* referente ao serviço. Dessa forma, o *envoy* receberia uma requisição indicando um *endpoint* que possui a rota "/nnrf", que é a rota do NRF. Assim o *envoy* do NF poderia encaminhar a mensagem ao *envoy* do NRF.

O problema dessa abordagem é que ela funciona apenas para a comunicação entre NFs e NRF. No cenário em que o NF A quiser se comunicar com outro NF B, novamente a consulta ao NRF ocorrerá a partir do *envoy*, porém o endereço retornado será o endereço do NF B, com isso o NF A se comunicará diretamente com o NF B. Se configurarmos os NFs para que cadastrem o endereço de seus respectivos *envoys* no NRF o problema persiste mas, dessa vez, a comunicação falhará, pois o NF A receberá o endereço do *envoy* NF B e tentará encaminhar a mensagem para ele sem um *SVID* válido. A Figura 3.5 mostra a sequência das mensagens em que um NF A tentou acessar o *envoy* B cadastrado na NRF e nao obteve sucesso por não

direcionar a saída para seu próprio *envoy*.

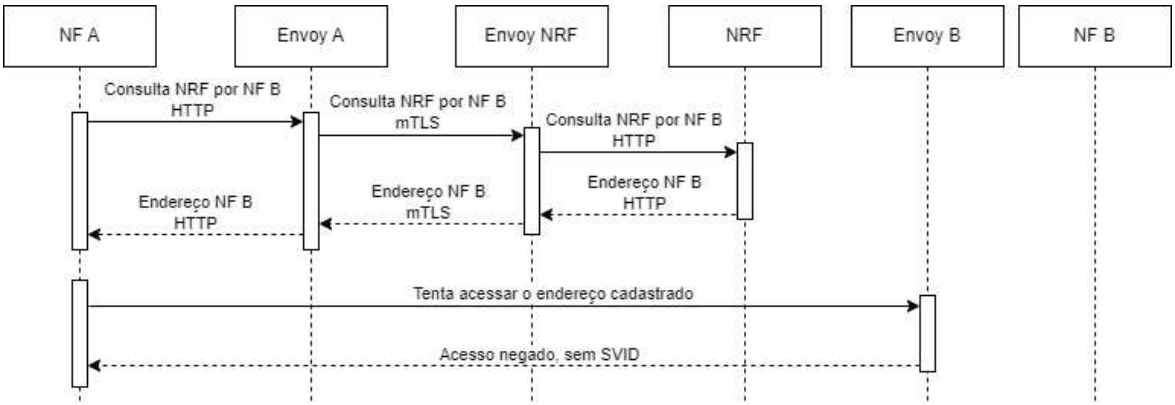


Figura 3.5: NF A falhando ao tentar comunicar com NF B

É necessário garantir que todo o tráfego de um NF seja encaminhado para seu respectivo *envoy*. Isso é possível utilizando regras de *firewall*, tal como o IPTables. O IPTables é um *firewall* em nível de pacotes que funciona comparando regras que indicam se um pacote tem ou não permissão de passar. Porém é necessário isolar as regras para cada NF, de forma que a regra do NF A não pode ser válida para o NF B. As regras também não podem ser aplicadas aos *envoys*, do contrário ocorreria requisições para si mesmo constantemente.

Para resolver o problema do isolamento foi criado um usuário para cada NF. Os NFs executam no ambiente de seus respectivos usuários e a regra de IPTables é aplicada baseada no ID do usuário. Por exemplo, o NF AMF executa como usuário amf com uid 1001 e a regra de IPTables aplicada para ele é a seguinte.

```
1  # iptables -t nat -A OUTPUT ! -d 127.0.0.18 -m owner --uid-owner amf -j
       DNAT --to-destination 127.0.1.18
```

No caso em que os serviços estão distribuídos em diferentes contêineres, VMs, nuvem, etc, é necessário que essa configuração do *proxy IPTables* seja feita da maneira que convém ao caso. Por exemplo, para os contêineres é possível a utilização de *sidecars* para o redirecionamento das informações aos *proxies envoys*. A mudança da configuração para encontrar e fazer o redirecionamento para o *envoy* não afeta a funcionalidade prática da proposta.

A regra indica que todo tráfego que não tenha como destino a própria origem e que o dono é o usuário amf deve ser destinada ao IP referente ao *envoy* do AMF (127.0.1.18). A

exceção da própria origem é para permitir que o serviço faça requisições ao seu próprio IP sem se comunicar com o *envoy*, se esse for o caso.

O exemplo da NF A se comunicando com a NF B com a configuração do IPTables aplicada pode ser vista na Figura 3.6. O endereço que a NF A possuí é o endereço do *envoy* B, recuperado através do NRF como sendo a NF B, porém a regra de IPTables força a saída pelo *envoy A*. O *envoy A* filtra o pacote redirecionando para o endereço do *envoy B* usando mTLS com o *SVID*.

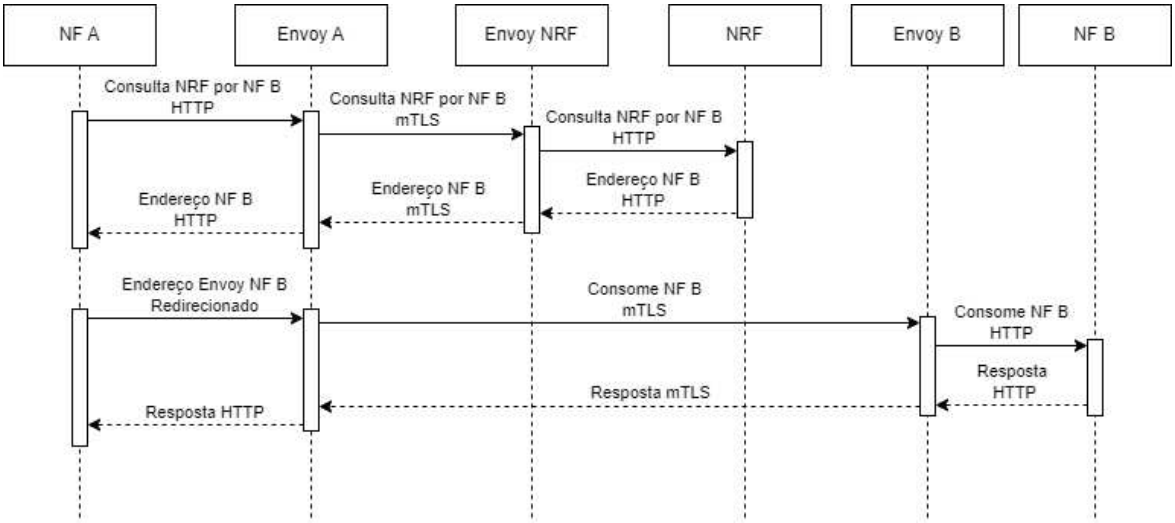


Figura 3.6: NF A redirecionando saída para o Envoy A

## 3.4 Validação do modelo

Para a validação do modelo é necessário garantir que o funcionamento original da implantação do núcleo 5G se mantenha. Além disso, é necessário observar as diferenças de consumo de recursos entre uma implantação que utiliza a abordagem proposta e outra implantação padrão do núcleo 5G. O consumo excessivo de recursos da máquina ou de tempo de resposta alto, em comparação com a implantação padrão, pode indicar inviabilidade da abordagem proposta.

A configuração correta da abordagem proposta deve permitir que a comunicação entre os serviços do núcleo 5G ocorra através de *proxies* utilizando o protocolo *mTLS*. É possível verificar que a troca de mensagens entre os serviços depende dos *proxies* ao remover o *proxy* de algum serviço e observar que o serviço não retorna mais respostas. A falta de identidades do *proxy* deve surtir o mesmo efeito. Também é possível, através dos registros dos *proxies*, observar o recebimento e encaminhamento dos pacotes.

Para comparar o consumo de tempo entre o cenário que utiliza a abordagem proposta e um cenário padrão, faremos uma série de registros e exclusão de equipamentos de usuário. No momento do registro do equipamento de usuário os serviços cooperam para autenticar, iniciar sessão, atribuir endereço, etc. Podemos calcular o tempo que demora para um equipamento de usuário se registrar nos dois cenário e observar a diferença causada pelos usos de recursos extras requeridos pela nossa abordagem.

O consumo de recursos também será observado em ambos os cenários. É possível extrair leituras referente à porcentagem de uso da CPU e à quantidade de memória utilizada nos momentos em que há registro de um equipamento de usuário, ou seja, há troca de comunicação entre os serviços.

A partir desses testes será possível averiguar a viabilidade da abordagem proposta, bem como se há ou não prejuízo para a rede 5G ao utilizar as ferramentas escolhidas.

# Capítulo 4

# Análise de Desempenho

Neste capítulo apresentamos a avaliação relacionada ao uso de recursos e latência no cenário proposto, em comparação com um cenário implementado de maneira semelhante mas sem o uso de *envoy*, *SPIRE* e *mTLS* no núcleo do 5G.

## 4.1 Cenários de Testes

Os testes serão executados em dois cenários, sendo um a implantação de referência e o outro o caso de uso proposto. A implantação de referência é mostrada na Figura 4.1, essa implantação possui apenas a instalação do Free5GC executando em uma VM. Os serviços não possuem nenhum tipo de isolamento, todos os serviços escutam na porta 8000 em um IP local diferente. A comunicação na SBI é HTTP. Esse cenário é modelo para o cenário seguinte, que considera a comunicação entre NFs usando *mTLS* com o auxílio de *envoy* e gerenciamento de certificados pelo *SPIRE*. Para a interação com os serviços do núcleo a VM UERANSIM é utilizada para simular as gNBs e os UEs.

A implantação caso de uso do *Free5GC* usando o *envoy* e o *SPIRE* é mostrada na Figura 4.2. Como descrito no Capítulo 3, o *envoy* recebe os certificados e garante que a comunicação entre os serviços do núcleo seja feita utilizando *mTLS*. O *IPTables* é usado para que toda a comunicação de saída de cada serviço seja redirecionada ao seu respectivo *envoy*. Uma VM é responsável por executar todos os serviços do núcleo e o *SPIRE Agent*, outra VM é responsável por executar o *SPIRE Server* e uma terceira VM é responsável por executar o emulador de antena e equipamento de usuário.

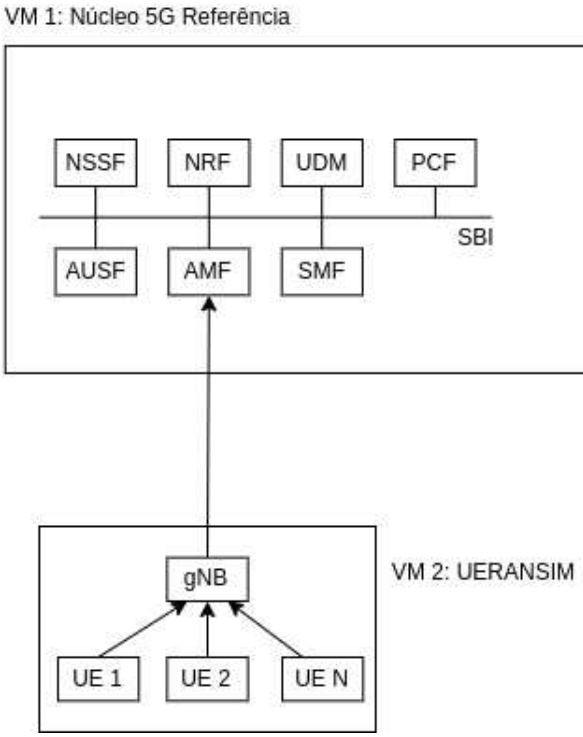


Figura 4.1: Núcleo 5G

A configuração das VMs é apresentada na Tabela 4.1. As VMs do núcleo possuem a mesma configuração. O sistema operacional instalado é o Ubuntu 20.04 com o kernel 5.4.0-162-generic, recomendado nos guias de instalação do Free5GC. A VM onde o UERANSIM é executado possui mais recurso de memória RAM por ser responsável por executar multiplas instâncias de UEs. A mesma VM é usada para os testes na configuração de referência e no caso de uso. A VM que executa o *SPIRE Server* possui 2 CPUs, 2 GB de memória RAM e o sistema operacional Ubuntu 20.04.

| Nome da VM | CPU | RAM | SO |
|---|---|---|---|
| Núcleo 5G Referência | 4 | 8 GB | Ubuntu 20.04 |
| Núcleo 5G Caso de Uso | 4 | 8 GB | Ubuntu 20.04 |
| UERANSIM | 4 | 32 GB | Ubuntu 20.04 |
| SPIRE Server | 2 | 8 GB | Ubuntu 20.04 |

Tabela 4.1: Configuração das VMs

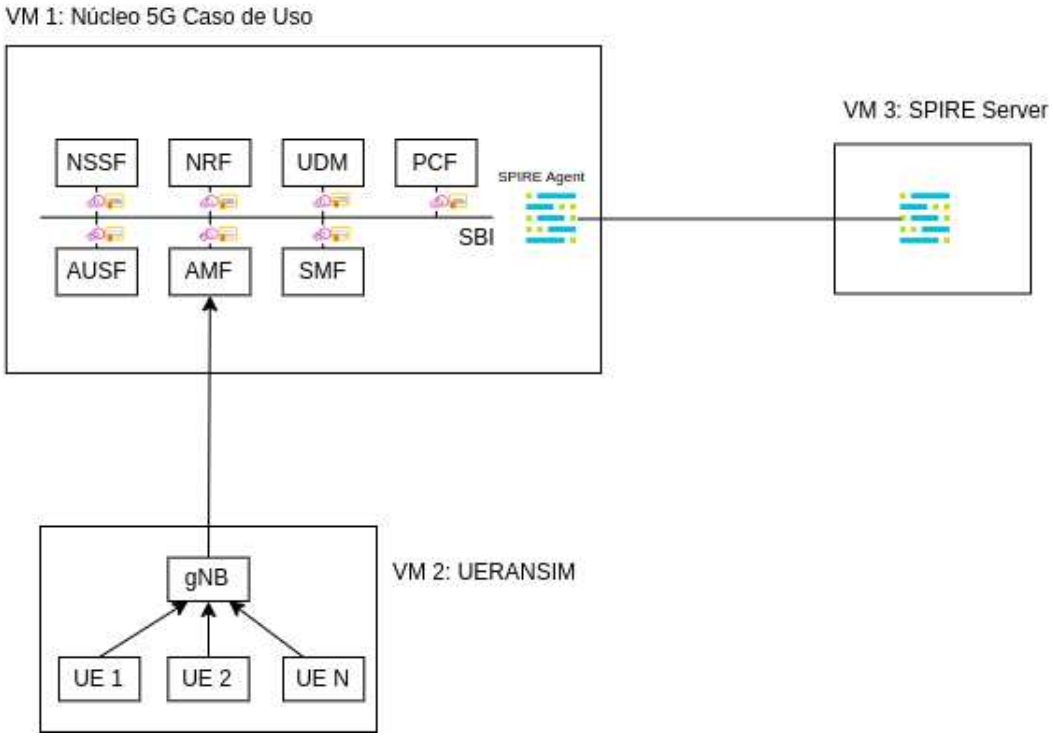


Figura 4.2: Núcleo 5G com SPIRE

Os testes serão separados entre testes de latência, que é a demora que um dispositivo leva para se autenticar no núcleo 5G, e testes de recursos, que indicam a diferença do consumo da CPU e da memória RAM quando o processo de autenticação é executado. O processo de autenticação e a garantia da sessão envolve todos os serviços do núcleo que estão executando na máquina.

O cenários de testes consideram o caso em que um UE apenas irá se autenticar de cada vez e o caso em que múltiplos UEs irão se conectar ao mesmo tempo. De acordo com a RFC 2544 [1] os testes de latência devem ser executados ao menos 20 vezes sendo o valor relatado a médias dos valores registrados. Levando essa recomendação em consideração a autenticação se repetirá 50 vezes para cada caso e, em cada vez, é calculado quanto tempo o UE ou os UEs demoraram para terminar o processo, bem como a porcentagem de memória que estava sendo consumida antes, durante e após esse processo.

No primeiro caso um UE apenas se conecta por vez, a sessão é aberta e, em seguida, o próximo, até que totalizem 50 autenticações. No segundo caso 25 UEs se autenticam ao mesmo tempo, no terceiro caso 50, no quarto caso 75 e no quinto caso 100.

Com mais de 100 UEs foi percebido uma negação de serviço, para os dois cenários. Na maior parte das tentativas, muitos UEs não conseguiam trocar informações com o núcleo em tempo hábil, nas três tentativas que lhes são configuradas. Nesse caso, ele entra em modo de espera. Essa negação de serviço ocorre justamente pela limitação de apenas uma instância do AMF para tratar todas as requisições. Para ampliar a capacidade é necessário aumentar a quantidade de instâncias e balancear a carga entre elas. Todos os UEs utilizam a mesma gNB para se conectar.

## 4.2 Testes de Latência

Os testes de latência podem ser úteis para indicar o impacto do uso dos *proxies envoy* agregado com o *SPIRE* na comunicação entre os serviços no cenário proposto. Como todos os serviços do núcleo 5G estão no mesmo local, o objetivo é saber quanto tempo o processamento e encaminhamento realizado pelo *envoy* acresce na comunicação. Além disso, em cenários distribuídos, cada *envoy* estaria localizado junto ao serviço pelo qual ele é responsável, acrescendo apenas o tempo de latência referente a distância dos serviços. Esse tempo de latência não está relacionado com a solução proposta neste trabalho. Com os testes de latência pretende-se descobrir quanto tempo demora, nos dois cenários, para que um equipamento de usuário (UE) seja autenticado e tenha sua sessão iniciada.

Um *script python* foi criado para executar os UEs. A responsabilidade do *script* é iniciar o UE e o cronômetro. Quando a mensagem indicando que a sessão foi aberta chega, o cronômetro é então paralisado. Esse processo é repetido quantas vezes indicado nos conjuntos de UEs criados. Durante a coleta dos dados foi percebido valores discrepantes que dificultavam a visualização dos gráficos, esses valores superavam muito o desvio padrão e indicam uma anomalia no momento da execução e leitura, os valores fora dos limites de desvio padrão foram removidos.

A Figura 4.3 mostra o gráfico de distribuição do primeiro caso de teste, onde apenas um UE é autenticado por vez, no cenário proposto e no cenário padrão. É possível perceber que não existe uma diferença muito grande entre o tempo que um UE leva para receber sua sessão, desde sua inicialização.

O segundo gráfico observado na Figura 4.4 mostra a distribuição do segundo e terceiro

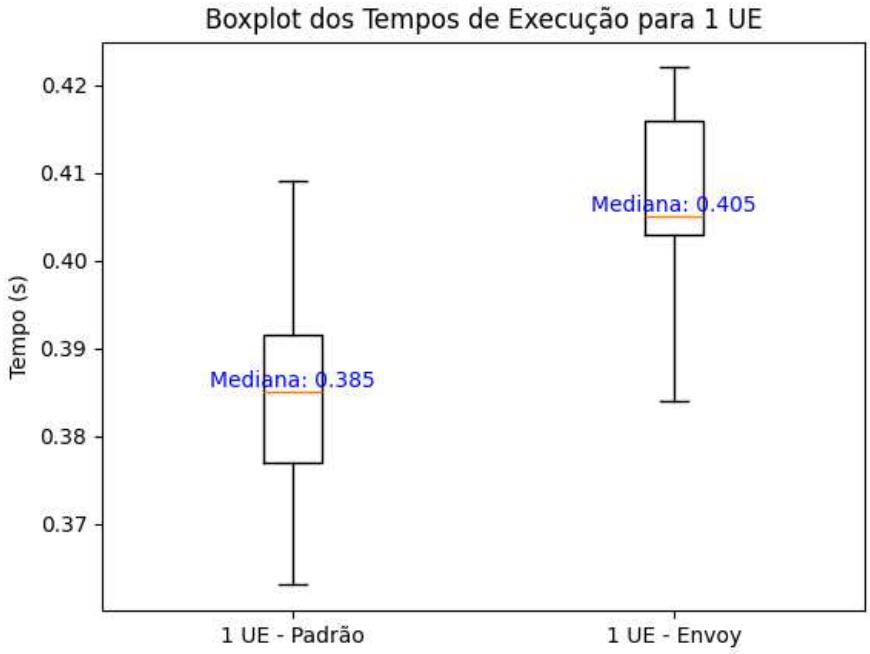


Figura 4.3: Tempo de autenticação caso de teste 1

casos de teste, onde 25 e 50 UEs são autenticados simultaneamente, no cenário proposto e no cenário padrão. É possível perceber um crescimento esperado da latência em relação ao primeiro caso de teste, justamente pela quantidade de UEs que estão fazendo a autenticação ser maior. Também é possível observar uma tendência de crescimento da variância da latência conforme a quantidade de UEs simultâneos aumentam, isso se deve ao fato de que os UEs precisam renovar as trocas de mensagens para realizar a autenticação devido a expiração das mensagens.

Na Figura 4.5 podemos observar o gráfico de distribuição do quarto e quinto casos de teste, onde 75 e 100 UEs são autenticados simultaneamente, no cenário proposto e no cenário padrão. Nesses casos também é possível observar o crescimento da latência e o aumento da variância pelo volume de UEs autenticados.

Analisando os três gráficos dos casos de teste, podemos concluir que a adição dos *proxies* e gerenciamento de certificados a partir do *SPIRE* no núcleo 5G causa alterações sutis no tempo de resposta do núcleo. Entretanto, esse tempo de resposta está mais sensível a outros eventos do que aos recursos que foram adicionados nesse caso de uso. Como os *pro-*

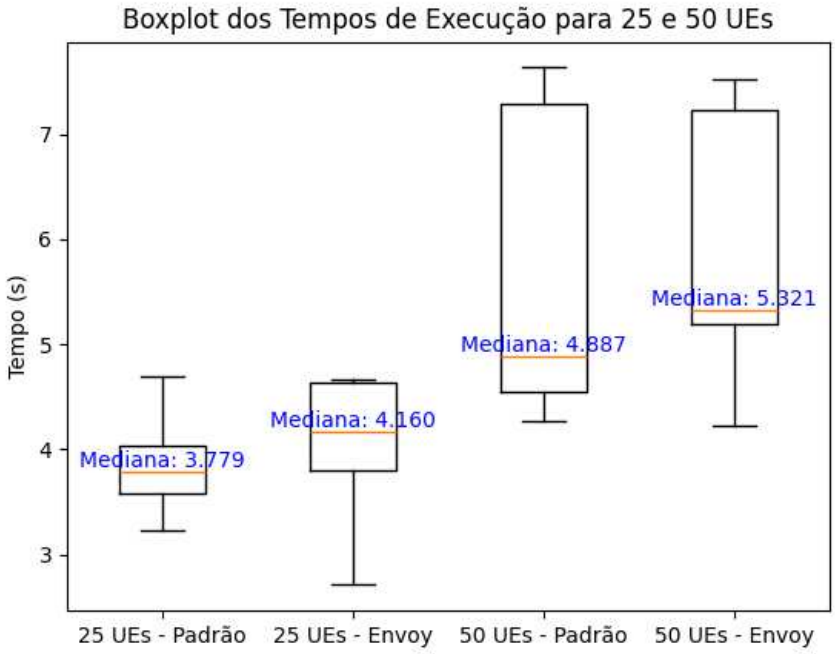


Figura 4.4: Tempo de autenticação caso de teste 2

*xies* executam localmente em relação ao serviço eles não adicionam tempo de transferência significativo, apenas tempo de processamento. O aumento da variância também pode ser observado no cenário de referência, não possuindo relação com a adição dos serviços empregados no caso de uso.

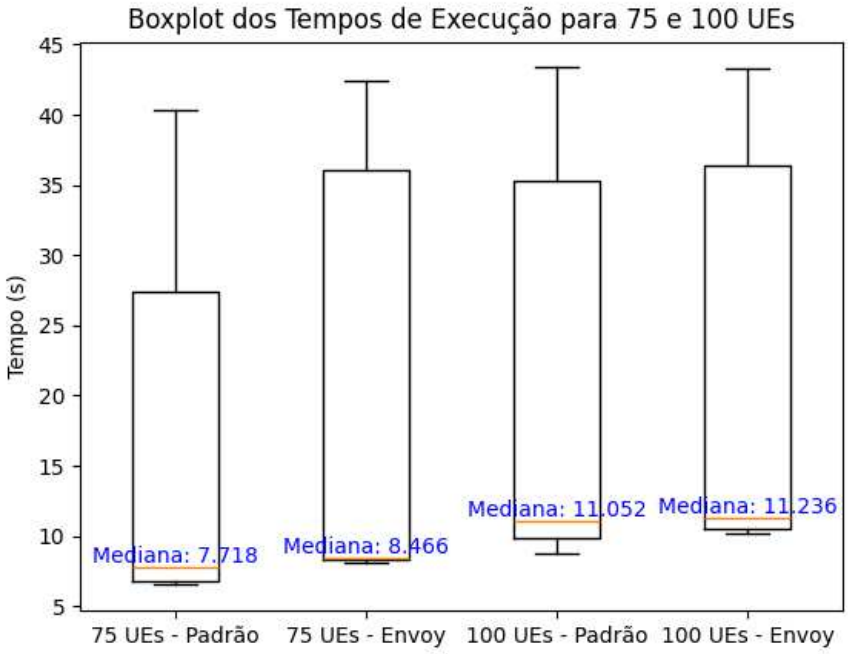


Figura 4.5: Tempo de autenticação caso de teste 3

## 4.3 Testes de Recursos

Os testes de recursos referem-se ao uso de memória RAM e processamento para desempenhar as tarefas do núcleo 5G. A coleta das informações referente à porcentagem de uso da memória RAM e processador foi realizada enquanto as demandas surgiam. No início, os serviços do núcleo estão executando sem trocar mensagens. Quando começa a ocorrer a autenticação de UEs os serviços começam a trocar as mensagens. Por fim, é feito um teste de acesso à Internet usando a interface de rede do UE criada.

Como nessa etapa pretende-se saber apenas o consumo dos recursos, é possível ir além e tentar conectar mais UEs simultaneamente. Mesmo que os UEs não finalizem a autenticação com sucesso obtendo as sessões, podemos ainda observar a porcentagem de uso da CPU e da memória até que os UEs sejam autenticados ou entrem em modo de espera. É apresentado um caso de teste com 250 UEs requisitando autenticação. Todos os UEs utilizam a mesma gNB para se conectar.

A Figura 4.6 apresenta o gráfico de consumo de CPU para a autenticação de um UE por vez totalizando 50 requisições de autenticação. A cor azul representa o cenário padrão e a

cor laranja representa o nosso cenário integrando *envoy* e *SPIRE*. Este gráfico, excepcionalmente, mostra o consumo de CPU para cada autenticação ocorrida. Há uma diferença sutil na porcentagem de uso da CPU entre os dois cenários. O cenário proposto apresenta um consumo um pouco mais acentuado quando o UE faz a requisição de autenticação. A maioria das autenticações no cenário padrão está entre 3% e 4%, no caso de uso há uma maior distribuição, mas uma tendência de crescimento.

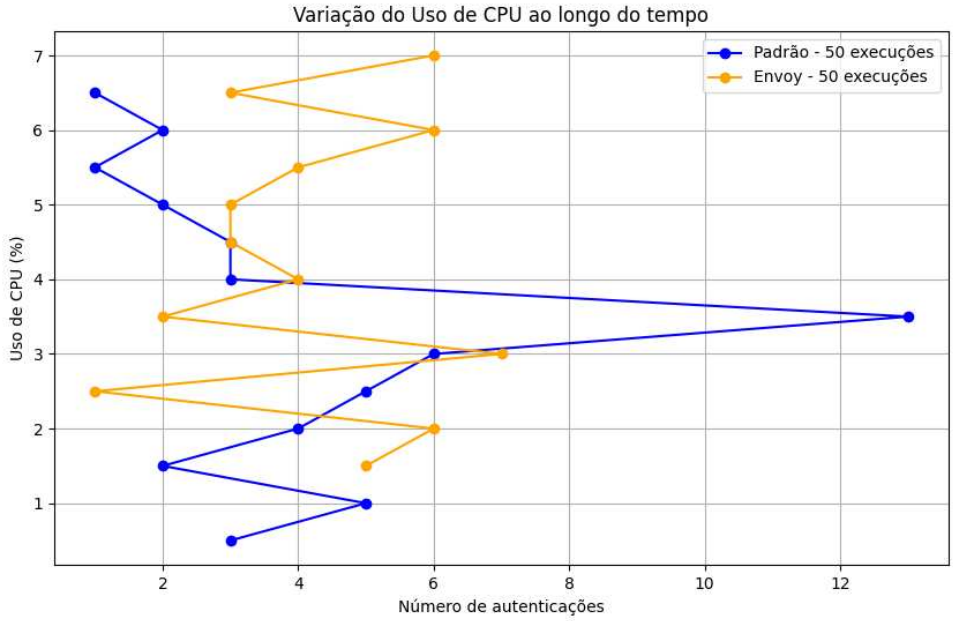


Figura 4.6: Consumo de CPU com 1 UE

Quando os UEs fazem as requisições de forma simultânea a porcentagem de uso da CPU aumenta. Esse comportamento pode ser observado na Figura 4.7, que apresenta o gráfico quando 25 UEs requisitam autenticação ao mesmo tempo. A Figura 4.8 apresenta o gráfico para 50 UEs, a Figura 4.9 apresenta o gráfico para 75 UEs e a Figura 4.10 apresenta o gráfico para o caso em que 100 UEs estão requisitando autenticação simultaneamente. Os gráficos apresentam a variação de uso percentual da CPU ao longo de um tempo de leitura, os momentos antes e depois dos picos observáveis nos gráficos são leituras em instantes que as autenticações não estavam ocorrendo.

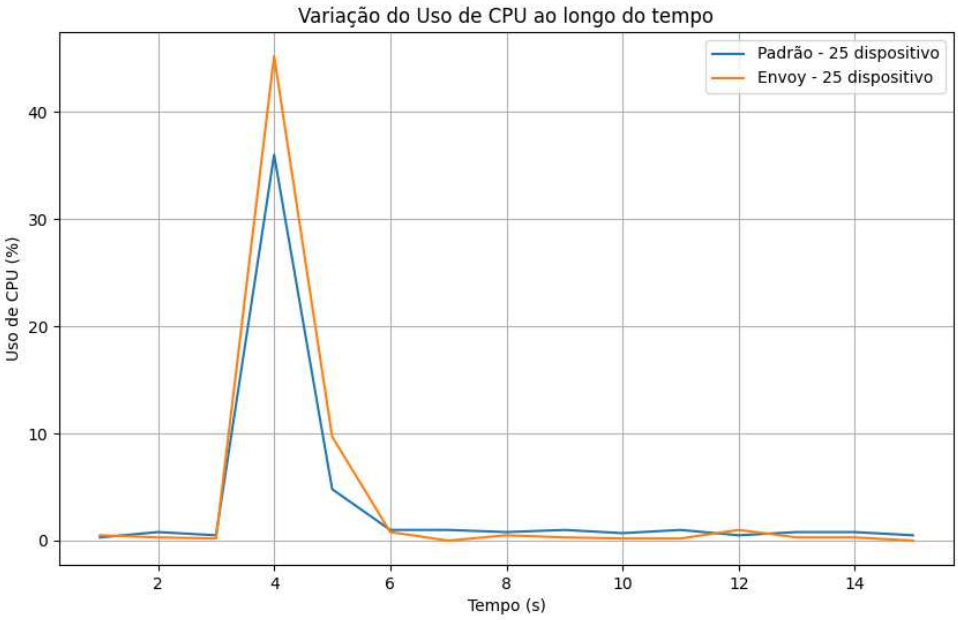


Figura 4.7: Consumo de CPU com 25 UEs

Com o aumento de UEs ainda é possível observar o aumento de consumo da CPU no cenário de caso de uso. Há uma demora maior para concluir a autenticação de todas as UEs conforme o número de UEs simultânea cresce. Além disso, a CPU alcança um pico e não varia muito durante o processamento das autenticações.

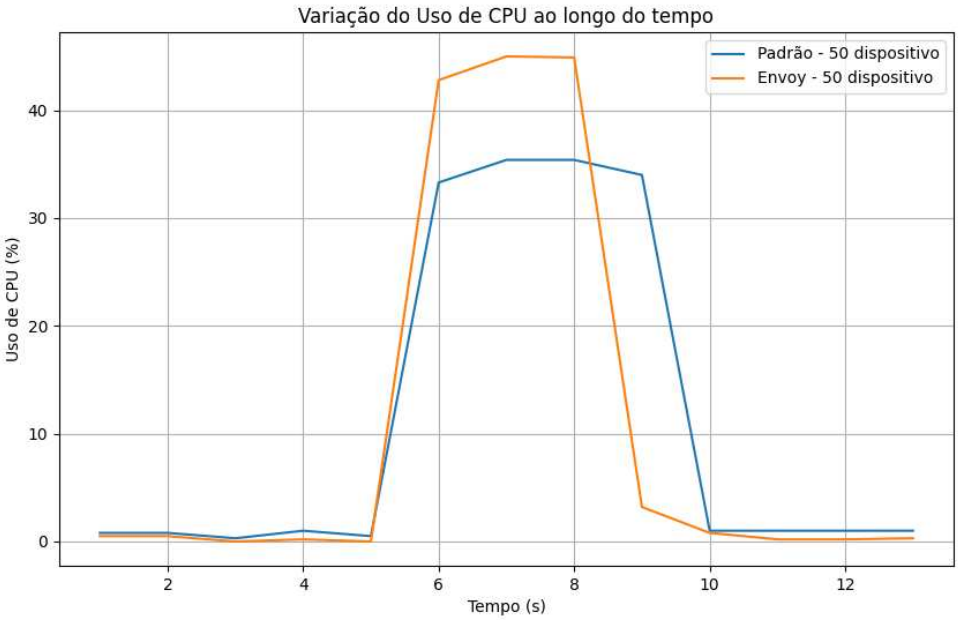


Figura 4.8: Consumo de CPU com 50 UEs

Na Figura 4.11 é apresentado o gráfico referente ao teste onde 250 UEs tentam realizar autenticação simultaneamente. Nesse caso é possível observar também que não há variação de recurso em comparação aos gráficos anteriormente apresentados. No gráfico é possível visualizar uma queda repentina da porcentagem de uso da CPU para o cenário padrão. Essa queda ocorre devido a falhas de autenticação e, nesse momento, o núcleo aguarda que os UEs que não conseguiram se conectar tentem novamente.

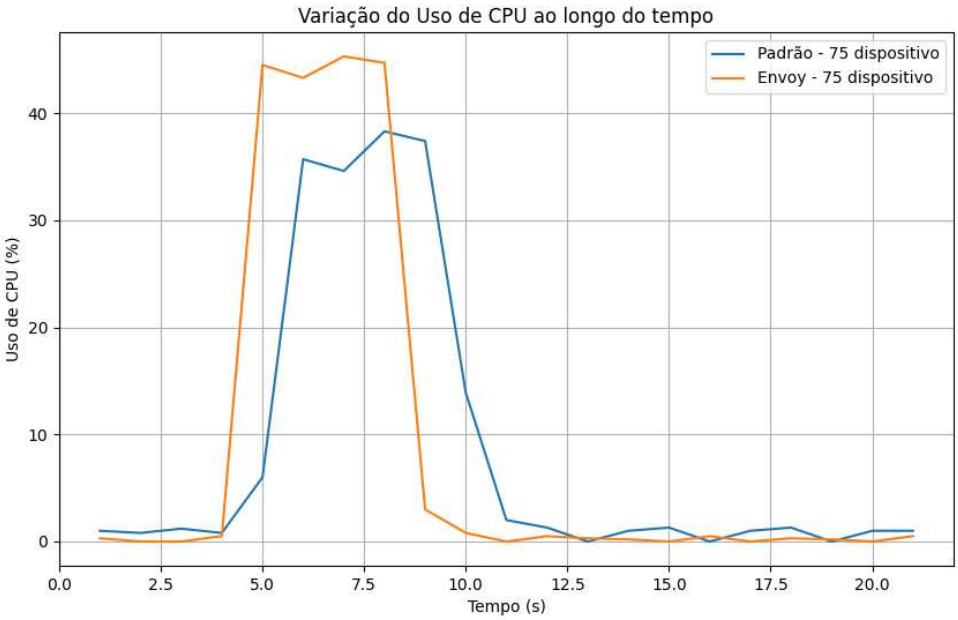


Figura 4.9: Consumo de CPU com 75 UEs

O recurso de memória RAM não varia em nenhum caso de teste, independente da autenticação estar sendo processada ou não. Além disso, também não há variações conforme o número de UEs que tenta se autenticar simultaneamente sobe. Para ilustrar, a Figura 4.12 apresenta o gráfico de variação de memória quando 25 UEs se autenticam simultaneamente e a Figura 4.13 apresenta o gráfico de variação de memória quando 250 UEs tentam se autenticar simultaneamente.

Embora não haja variação de consumo de memória, é importante observar que o cenário proposto necessita de uma porcentagem maior para executar do que o cenário padrão. Isso se deve ao fato que o cenário proposto possui mais componentes executando para realizar a tarefa, o *envoy* e o *SPIRE Agent*.

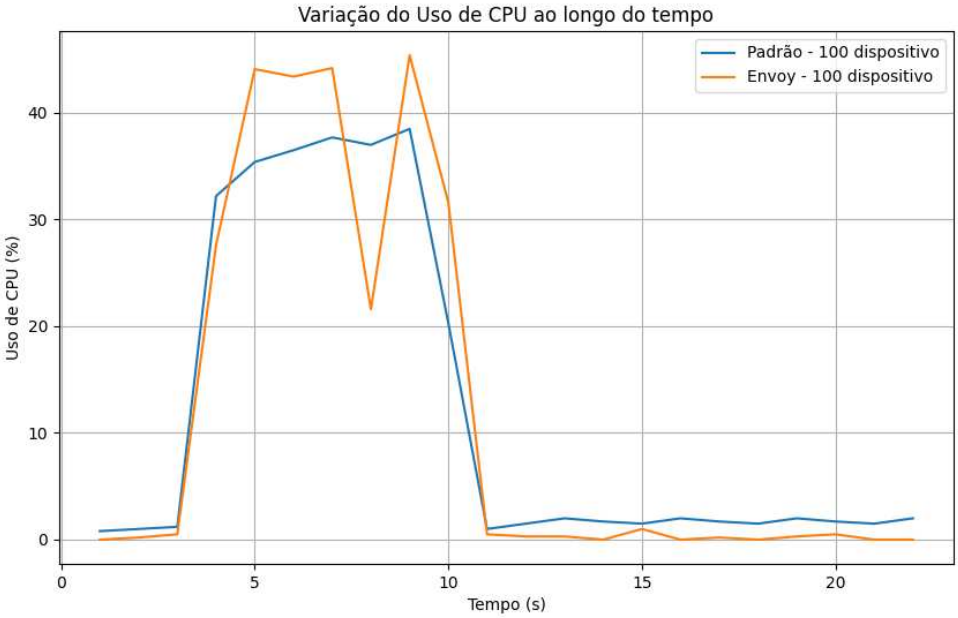


Figura 4.10: Consumo de CPU com 100 UEs

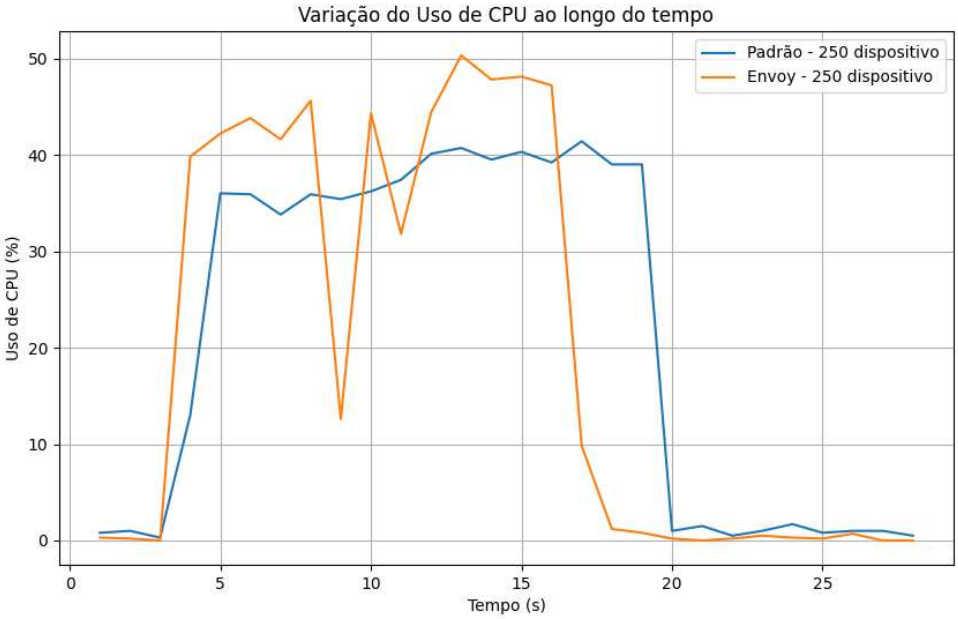


Figura 4.11: Consumo de CPU com 250 UEs

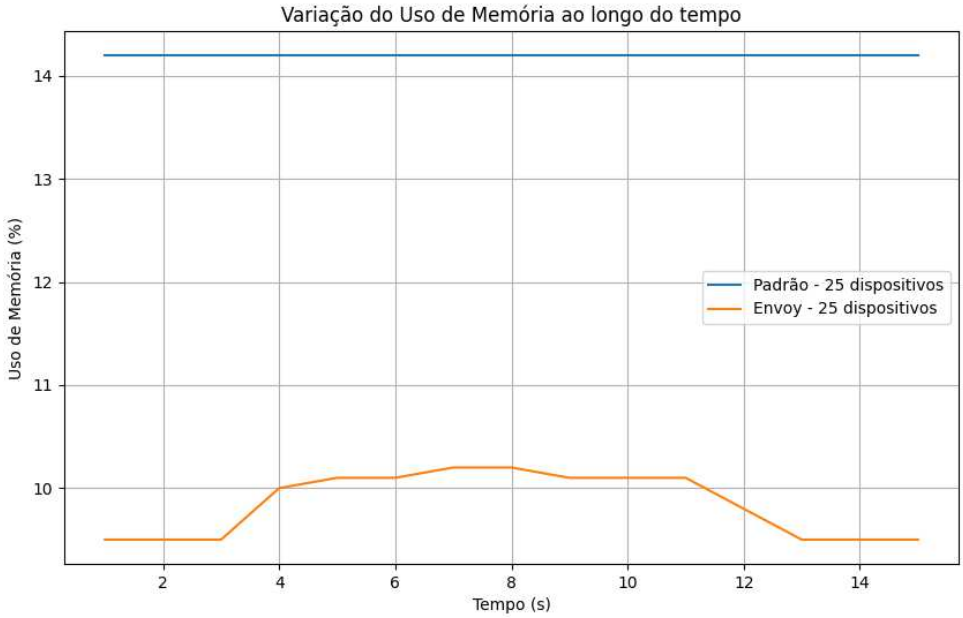


Figura 4.12: Consumo de memória com 25 UEs

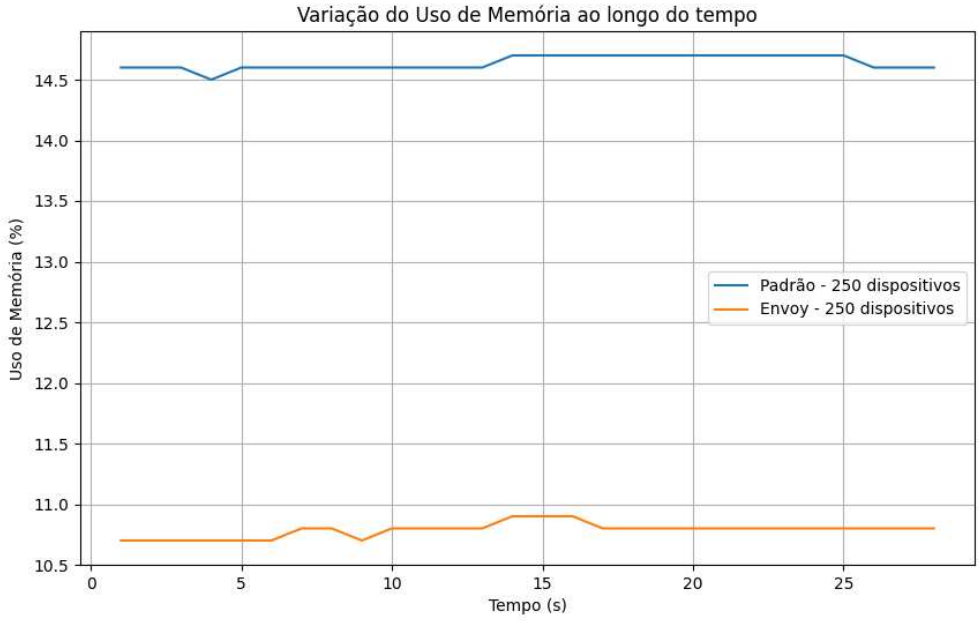


Figura 4.13: Consumo de memória com 250 UEs

# Capítulo 5

# Conclusão

Neste trabalho nós propomos uma abordagem não intrusiva para a integração do núcleo 5G com o *SPIRE*, garantindo autenticação e gerenciamento de identidade para os serviços do núcleo 5G. O gerenciamento de identidades e certificados é essencial para os serviços 5G, tendo em vista a recomendação que a comunicação entre esses serviços seja feita utilizando o protocolo *mTLS*.

Para garantir a abordagem não intrusiva, um *proxy* para cada serviço 5G foi configurado. O papel desses *proxies* é intermediar a comunicação entre os serviços, a comunicação que ocorre na SBI. Além disso, para garantir a utilização do protocolo *mTLS*, os *proxies* também possuem o papel de recuperar certificados e traduzir a comunicação HTTP, usada para se comunicar com o serviço ao qual o *proxy* está anexado para *mTLS* ao *proxy* de destino.

Para os testes da abordagem, o projeto de código aberto *Free5GC* foi utilizado como núcleo da rede 5G. Um *proxy envoy* para cada serviço do plano de controle do *Free5GC* foi configurado. Os *proxies envoy* são as cargas de trabalho que conseguem recuperar *SVIDs* do *SPIRE*. Os *SVIDs* são usados para estabelecer a comunicação *mTLS*. Para simular as antenas e equipamentos de usuários, foi utilizado o emulador *UERANSIM*.

A solução não intrusiva proposta permite flexibilidade da implementação do núcleo 5G, não estando limitada ao projeto *Free5GC*. Em contra partida o caso de uso criado depende da configuração correta dos serviços utilizados, em especial a estratégia de isolamento utilizado para os serviços do núcleo 5G e redirecionamento de pacotes utilizando IPTables. Cenários que utilizam componentes diversos necessitam de configurações semelhantes para garantir o funcionamento da abordagem proposta, como por exemplo uma implantação que considera

o uso de contêineres pode usar *sidecars* para isolar o *proxy* e o serviço. A abordagem não está limitada a infraestrutura apresentada e é compatível com o núcleo 5G a partir da *release* 15 da 3GPP, infraestruturas diferentes podem necessitar de diferentes configurações.

Os experimentos foram realizados em dois cenários, um possuindo um núcleo 5G com nossa abordagem não intrusiva e outro com um núcleo 5G sem nenhum serviço extra na máquina. Foi possível observar que no cenário proposto houve alterações sutis na latência da comunicação entre os serviços do núcleo 5G. Naturalmente, houve acréscimo no uso de memória da máquina no cenário que utiliza os *proxies* em comparação ao cenário que não possui *proxies*, porém não houve variação da taxa de uso da memória com ou sem comunicação entre os serviços. A taxa de uso da CPU não possui diferença perceptível entre os dois cenários, ocorrendo picos quando há troca de mensagens entre os serviços do núcleo. Esses picos não se diferenciaram entre os cenários.

Durante o desenvolvimento deste trabalho, várias maneiras de implantar o núcleo 5G foram analisadas, como a conteinerização desses serviços utilizando diferentes orquestradores. Futuramente, é possível considerar a utilização da abordagem proposta para um núcleo 5G executando em contêineres, onde as políticas de encaminhamento de pacotes seriam diferentes.

O processo de atestação utilizado neste trabalho levou em consideração apenas o identificador do usuário linux que executa o serviço do núcleo 5G. É possível alterar essa atestação para que ocorra utilizando critérios mais rigorosos de segurança, como verificação de informações da nuvem, de hardware, contêiner, entre outros.

Com relação a utilização de *proxies*, é possível alterar a configuração do *envoy* para que as informações do serviço sejam adicionadas ou removidas de maneira dinâmica. Isso permite que os *proxies* tomem conhecimento dos serviços apenas no momento em que o registro ao NRF ocorrer. Dessa maneira, não é necessário que os *proxies* estejam pré-configurados, torando assim a rede mais dinâmica e escalável.

# Bibliografia


[1] Benchmarking Methodology for Network Interconnect Devices. RFC 2544, March 1999.

[2] 3GPP. 3rd Generation Partnership Project (3GPP); Technical Specification Group Services and System Aspects (SA3): TS 33.501: Security Architecture and Procedures for 5G System. Technical Report TS 33.501, 3rd Generation Partnership Project (3GPP), Dezembro 2017.

[3] Yahuza Bello, Ahmed Refaey Hussein, Mehmet Ulema, and Juanita Koilpillai. On sustained zero trust conceptualization security for mobile core networks in 5g and beyond. *IEEE Transactions on Network and Service Management*, 19(2):1876–1889, 2022.

[4] Chafika Benzaïd, Tarik Taleb, and Muhammad Zubair Farooqi. Trust in 5g and beyond networks. *IEEE Network*, 35(3):212–222, 2021.

[5] Gabrial Brown. Service-based architecture for 5g core networks. *Huawei White Paper*, 1, 2017.

[6] Tulja Vamshi Kiran Buyakar, Harsh Agarwal, Bheemarjuna Reddy Tamma, and Antony A Franklin. Prototyping and load balancing the service based architecture of 5g core using nfv. In *2019 IEEE Conference on Network Softwarization (NetSoft)*, pages 228–232. IEEE, 2019.

[7] Brian Campbell, John Bradley, Nat Sakimura, and Torsten Lodderstedt. OAuth 2.0 Mutual-TLS Client Authentication and Certificate-Bound Access Tokens. RFC 8705, Fevereiro 2020.

[8] Cooper Coldwell, Denver Conger, Edward Goodell, Brendan Jacobson, Bryton Petersen, Damon Spencer, Matthew Anderson, and Matthew Sgambati. Machine learning 5g attack detection in programmable logic. In *2022 IEEE Globecom Workshops (GC Wkshps)*, pages 1365–1370, 2022.

[9] MITRE Corporation. CWE - Common Weakness Enumeration, 2023.

[10] Evan Gilman Ian Haken Frederick Kautz Umair Khan Max Lambrecht Brandon Lum Agustín Martínez Fayó Eli Nesterov Andres Vega Michael Wardrop Daniel Feldman, Emily Fox. *Solving the Bottom Turtle — a SPIFFE Way to Establish Trust in Your Infrastructure via Universal Identity*. Booksprints, 1st edition, 2020.

[11] Van-Binh Duong and Younghan Kim. A design of service mesh based 5g core network using cilium. In *2023 International Conference on Information Networking (ICOIN)*, pages 25–28. IEEE, 2023.

[12] Envoy Proxy Contributors. Envoy proxy documentation. `https://www.envoyproxy.io/docs`, Acessado em 25 de junho de 2023.

[13] Zebing Feng, Peng Zhou, Qi Wang, and Weiqiang Qi. A dual-layer zero trust architecture for 5g industry mec applications access control. In *2022 IEEE 5th International Conference on Electronic Information and Communication Technology (ICEICT)*, pages 100–105, 2022.

[14] Free5GC. User guide of free5gc project, Acessado em 17 janeiro de 2023 2023. `https://free5gc.org/guide/`.

[15] Xinxin Hu, Caixia Liu, Shuxin Liu, Wei You, and Yu Zhao. Signalling security analysis: Is http/2 secure in 5g core network? In *2018 10th International Conference on Wireless Communications and Signal Processing (WCSP)*, pages 1–6. IEEE, 2018.

[16] Jiegang Lu, Limin Xiao, Zhigang Tian, Ming Zhao, and Wenbo Wang. 5g enhanced service-based core design. In *2019 28th Wireless and Optical Communications Conference (WOCC)*, pages 1–5, 2019.

[17] Ueli Maurer. Modelling a public-key infrastructure. In *Computer Security—ESORICS 96: 4th European Symposium on Research in Computer Security Rome, Italy, September 25–27, 1996 Proceedings 4*, pages 325–350. Springer, 1996.

[18] Georg Mayer. Restful apis for the 5g service based architecture. *Journal of ICT Standardization*, 6, May 2018.

[19] ETSI Group on Security. Network functions virtualisation (nfv); nfv security; security and trust guidance. Technical report, ETSI, 2014. Acessado em 01 de maio de 2023.

[20] Eric Rescorla. The Transport Layer Security (TLS) Protocol Version 1.3. RFC 8446, Agosto 2018.

[21] Stefan Rommer, Peter Hedman, Magnus Olsson, Lars Frid, Shabnam Sultana, and Catherine Mulligan. *5G Core Networks: Powering Digitalization*. Academic Press, 2019.

[22] William R Simpson and Kevin E Foltz. Network segmentation and zero trust architectures. In *Lecture Notes in Engineering and Computer Science, Proceedings of the World Congress on Engineering (WCE)*, pages 201–206, 2021.

[23] SPIFFE. Spire concepts. `https://spiffe.io/docs/latest/spire-about/spire-concepts/`. Acessado em 23 de junho de 2023.

[24] VA Stafford. Zero trust architecture. *NIST special publication*, 800:207, 2020.

[25] Naeem Firdous Syed, Syed W. Shah, Arash Shaghaghi, Adnan Anwar, Zubair Baig, and Robin Doss. Zero trust architecture (zta): A comprehensive survey. *IEEE Access*, 10:57143–57179, 2022.

[26] International Telecommunication Union. Overview of trust provisioning for information and communication technology infrastructures and services. T-REC Y.3052, International Telecommunication Union, 2017.

[27] National Yang Ming Chiao Tung University. Free5gc compose, Acessado em 17 janeiro de 2023 2023. `https://github.com/free5gc/free5gc-compose`.

[28] Alireza Hosseini Vasoukolaei, Danish Sattar, and Ashraf Matrawy. Tls performance evaluation in the control plane of a 5g core network slice. In *2021 IEEE Conference on Standards for Communications and Networking (CSCN)*, pages 155–160. IEEE, 2021.

# Abreviações e Acrônimos

3GPP - *3rd Generation Partnership Project*

5G SA - *5G Standalone Core*

AF - *Application Function*

AMF - *Access and Mobility Management Function*

API - *Application Programming Interface*

AUSF - *Authentication Server Function*

AWS - *Amazon Web Services*

CA - *Certificate Authority*

CNI - *Container Network Interface*

CUPS - *Control and User Plane Separation*

CWE - *Common Weakness Enumeration*

ETSI - *European Telecommunications Standards Institute*

GCC - *GNU Compiler Collection*

gNB - *Next Generation Node B*

gRPC - *Google Remote Procedure Call*

HTTP - *Hypertext Transfer Protocol*

HTTPS - *Hypertext Transfer Protocol Secure*

IETF - *Internet Engineering Task Force*

IoT - *Internet of Things*

ITU - *International Telecommunication Union*

LMF - *Location Management Function*

MEC - *Mobile Access Edge Computing*

mTLS - *Mutual Transport Layer Security*

N3IWF - *Non-3GPP Inter Working Function*

NEF - *Network Exposure Function*

NF - *Network Function*

NFV - *Network Functions Virtualization*

NRF - *Network Repository Function*

NSSF - *Network Slice Selection Function*

NYCU - *National Yang Ming Chiao Tung University*

PA - *Policy Administrator*

PCF - *Policy Control Function*

PE - *Policy Engine*

PEP - *Policy Enforcement Point*

PKI - *Public Key Infrastructure*

QoS - *Quality-of-Service*

RAN - *Radio Access Network*

REST - *REpresentational State Transfer*

SBA - *Service Based Architecture*

SBI - *Service Based Interface*

SDN - *Software-Defined Networking*

SDP - *Software-Defined Perimeter* SMF - *Session Management Function*

SMSF - *Short Message Service Function*

SPIFFE - *Secure Production Identity Framework for Everyone*

SPIRE - *SPIFFE Runtime Environment* SSL - *Secure Sockets Layer*

SVID - *SPIFFE Verifiable Identity Document* TCP - *Transmission Control Protocol*

TLS - *Transport Layer Security*

UDM - *Unified Data Management*

UDR - *Unified Data Repository*

UE - *User Equipment*

ULCL - *Uplink Classifier*

UPF - *User Plane Function*

VM - *Virtual Machine*

WAN - *Wide Area Network*

YAML - *YAML Ain't Markup Language*

# Apêndice A

# Free5GC

O *Free5GC* [14] é um projeto de código aberto desenvolvido na linguagem Go, com a licença Apache 2.0, que implementa as principais funções de rede do Core 5G, definidas a partir da revisão 15 da 3GPP. Portanto, nem todas as funcionalidades das revisões mais recentes do 3GPP estão disponíveis. Atualmente o principal mantenedor desse projeto é a Universidade Nacional Yang Ming Chiao Tung (NYCU). As funções de rede são separadas entre plano de controle e plano de usuário.

A vantagem de possuir os planos de controle e usuário separados, como proposto na especificação 5G do *Control and User Plane Separation* (CUPS), é a possibilidade de implantação de múltiplos servidores de plano de usuário em um ambiente, mas mantendo as funcionalidades do plano de controle centralizadas. Isso permite a implementação de cenários como o *Mobile Access Edge Computing* (MEC). Nos casos em que o MEC está associado ao plano de usuário, serviços podem ser oferecidos sem a necessidade dos dados gerados pelos usuários trafegarem até os componentes do plano de controle. Os componentes do plano de controle gerenciam características mais pontuais como o plano de dados e as políticas de acesso. As atualizações com os planos de usuários ocorrem de maneira periódica para continuar permitindo ou não o tráfego de determinados usuários, proporcionando um benefício geográfico que diminui a latência de acesso a alguns serviços [14].

O *Free5GC* possui apenas as funções de rede referentes ao núcleo 5G independente (*5G Standalone Core* — 5G SA), que não tem dependência das funções de controle de rede do 4G LTE. As funções de rede do *Free5GC* funcionam como processos separados e independentes, podendo ser modificados e testados individualmente. Seguindo a especificação da 3GPP na

*release* 15 [2], a arquitetura do *Free5GC* é baseada em serviços.

Como o 5G SA Core utiliza uma arquitetura baseada em serviços, onde um conjunto de funções de rede provê serviços para outras funções de rede autorizadas, é necessário que haja um componente responsável por gravar e manter atualizados os registros das funções de rede, no caso a NRF. O componente responsável por gerenciar a conexão entre as gNBs (estações base da rede móvel 5G) é a AMF. As autenticações são gerenciadas por três componentes, o UDM, a AUSF e o UDR. O gerenciamento de sessão é executado pela SMF. As operações de partição da rede são de responsabilidade da NSSF. As políticas de usuário são aplicadas pela PCF. Além disso, o *Free5GC* possui ainda suporte para IPTV e um orquestrador 5G.

O plano de usuário possui apenas a função de gerenciar a troca de pacotes entre os gNBs e a WAN externa. O componente responsável por isso é a UPF. O *Free5GC* também implementa uma função de rede para conexão de redes não móveis, *Non-mobile Access Network* (N3IWF), e a funcionalidade *Uplink Classifier* (ULCL), que reside no componente UPF e é responsável por aplicar filtros ao tráfego do equipamento de usuário, além de redirecionar esse tráfego para acesso a dados de rede local ou de borda.

O projeto também disponibiliza uma versão baseada em Docker, chamada de *Free5GC Compose*, permitindo assim a implantação das funções de rede do núcleo 5G de maneira mais simples e gerenciamento a partir de um única ferramenta. Neste cenário, cada função de rede possui um Dockerfile próprio.

O projeto possui suporte para a maioria dos sistemas operacionais e pode ser também executado em contêineres. As configurações dos componentes ocorrem a partir de arquivos no formato YAML. Após a execução, os gNBs podem ser configurados para se comunicarem com o núcleo a partir do AMF e os dispositivos que se conectam com os gNBs podem também ser cadastrados no núcleo. Para ambientes de teste, os gNBs podem ser emulados a partir de projetos como o UERANSIM[1]. A Figura A.1 ilustra as funções de rede implementadas pelo *Free5GC*.

Para a implantação do *Free5GC*, as seguintes ferramentas são necessárias:

- Sistema operacional baseado em Linux, podendo haver uma ou mais função de rede no mesmo sistema. A versão do Kernel deve ser 5.0.0.23-generic ou 5.4.x;

[1] `https://github.com/aligungr/UERANSIM`

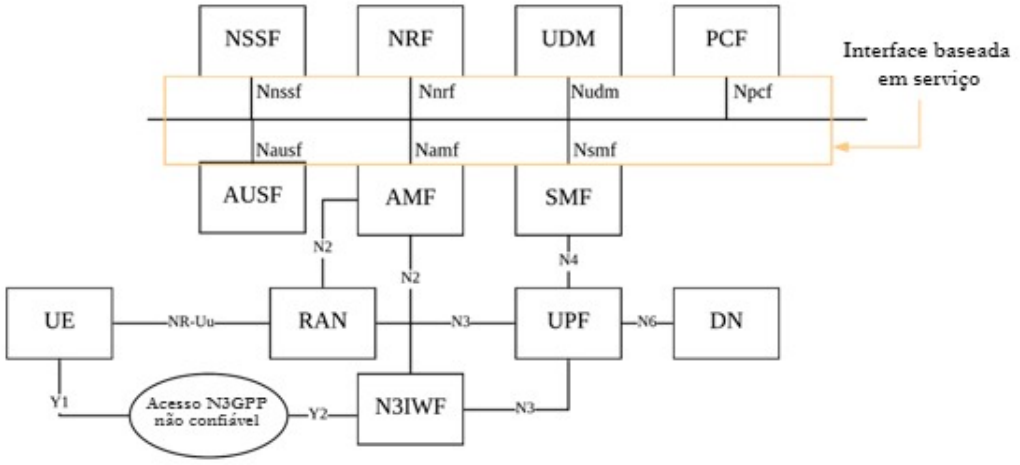


Figura A.1: Funções de rede implementadas pelo Free5GC [14].

- GTP5G Kernel Module deve ser instalado para usar o componente UPF;
- Para executar o *Free5GC Compose* (disponível em [27]) são necessários:
  - GTP5G Kernel Module;
  - Docker Engine necessário para executar os contêineres *Free5GC*;
  - Docker Compose necessário para inicializar a pilha *Free5GC*.
- MongoDB é requerido pelo plano de controle;
- Compiladores GCC 7.3.0+ e Go 1.14.4+ para compilação do projeto.

# Apêndice B

# Configuração dos Serviços

## B.1 Código YAML da AMF

Código B.1: Configuração do AMF

```yaml
info:
    version: 1.0.3
    description: AMF inicial
configuration:
    amfName: AMF
    ngapIpList:
        - 127.0.0.18
    sbi:
        scheme: http
        registerIPv4: 127.0.0.18
        bindingIPv4: 127.0.0.18
        port: 8000
    serviceNameList:
        - namf-comm
        - namf-evts
        - namf-mt
        - namf-loc
        - namf-oam
    ...
    nrfUri: http://127.0.1.18:8000
    ...
```

# Apêndice C

# Configuração dos Proxies Envoy

## C.1 Cluster Envoy SPIRE

Código C.1: Cluster Envoy para SPIRE

```
clusters:
- name: spire_agent
  connect_timeout: 0.25s
  http2_protocol_options: {}
  load_assignment:
    cluster_name: spire_agent
    endpoints:
      - lb_endpoints:
        - endpoint:
            address:
              pipe:
                path: /tmp/spire-agent/public/api.sock
```

## C.2 Recuperando SVID

Código C.2: Recuperando SVID

```
common_tls_context:
    tls_certificate_sds_secret_configs:
    - name: spiffe://example.org/nf
```

```
4           sds_config:
5             resource_api_version: V3
6             api_config_source:
7               api_type: gRPC
8               transport_api_version: V3
9               grpc_services:
10                envoy_grpc:
11                  cluster_name: spire_agent
```

O nome do certificado *SVID* mostrado na linha 3 do Código C.2 é o SPIFFE ID do serviço representado pelo *envoy*.

# Apêndice D

# Integração do Envoy com as NFs

## D.1 Configuração do YAML do NRF

Código D.1: Configuração do NRF

```
info:
    version: 1.0.1
    description: NRF initial local configuration
configuration:
    MongoDBName: free5gc
    MongoDBUrl: mongodb://127.0.0.1:27017
    sbi:
        scheme: http
        registerIPv4: 127.0.0.10
        bindingIPv4: 127.0.0.10
        port: 8000
    DefaultPlmnId:
        mcc: 208
        mnc: 93
    serviceNameList:
        - nnrf-nfm
        - nnrf-disc
```

## D.2 Listener do Envoy NRF

Código D.2: Listener do Envoy NRF

```
listeners:
- name: ingress
  address:
    socket_address:
    address: 127.0.1.10
    port_value: 8000
  filter_chains:
  - filters:
    - name: envoy.filters.network.http_connection_manager
    typed_config:
      ``@type": type.googleapis.com/envoy.extensions.filters.network.
          http_connection_manager.v3.HttpConnectionManager
      codec_type: AUTO
      stat_prefix: ingress_http
      common_http_protocol_options:
        idle_timeout: 1s
      access_log:
      - name: envoy.access_loggers.file
        typed_config:
          ``@type": type.googleapis.com/envoy.extensions.access_loggers.
              file.v3.FileAccessLog
          path: ``/dev/stdout"
      route_config:
        name: nrf
        virtual_hosts:
          - name: nrf
            domains: [``*"]
            routes:
              - match:
                  prefix: ``/"
                route:
                  cluster: nrf
      http_filters:
      - name: envoy.filters.http.router
        typed_config:
```

```
34            ''@type": type.googleapis.com/envoy.extensions.filters.http.
                router.v3.Router
35    transport_socket:
36      name: envoy.transport_sockets.tls
37      typed_config:
38        ''@type": type.googleapis.com/envoy.extensions.transport_sockets.
            tls.v3.DownstreamTlsContext
39        common_tls_context:
40          tls_certificate_sds_secret_configs:
41          - name: spiffe://example.org/nrf
42            sds_config:
43              resource_api_version: V3
44              api_config_source:
45                api_type: gRPC
46                transport_api_version: V3
47                grpc_services:
48                  envoy_grpc:
49                    cluster_name: spire_agent
```

## D.3 Cluster do Envoy NRF

Código D.3: Envoy NRF

```
1  clusters:
2  - name: nrf
3    connect_timeout: 0.25s
4    type: STATIC
5    lb_policy: ROUND_ROBIN
6    load_assignment:
7      cluster_name: nrf
8      endpoints:
9        - lb_endpoints:
10         - endpoint:
11             address:
12               socket_address:
13                 address: 127.0.0.10
14                 port_value: 8000
```

## D.4 Listener Envoy AMF

Código D.4: Envoy AMF

```
route_config:
    name: amf
    virtual_hosts:
      - name: amf
        domains: [“*”]
        routes:
          - match:
              prefix: “/nnrf”
            route:
              cluster: nrf
          - match:
              prefix: “/namf”
            route:
              cluster: amf
          - match:
              prefix: “/nausf”
            route:
              cluster: ausf
          - match:
              prefix: “/nnssf”
            route:
              cluster: nssf
          - match:
              prefix: “/nudm”
            route:
              cluster: udm
          - match:
              prefix: “/npcf”
            route:
              cluster: pcf
          - match:
```

```
32                prefix: ''/nsmf"
33              route:
34                cluster: smf
```

## D.5 Clusters do Envoy AMF

Código D.5: Clusters do Envoy AMF

```
1  clusters:
2    - name: nrf
3      connect_timeout: 0.25s
4      type: STRICT_DNS
5      lb_policy: ROUND_ROBIN
6      load_assignment:
7        cluster_name: nrf
8        endpoints:
9          - lb_endpoints:
10           - endpoint:
11               address:
12                 socket_address:
13                   address: 127.0.1.10
14                   port_value: 8000
15     transport_socket:
16       name: envoy.transport_sockets.tls
17       typed_config:
18         ''@type": type.googleapis.com/envoy.extensions.transport_sockets.
               tls.v3.UpstreamTlsContext
19         common_tls_context:
20           tls_certificate_sds_secret_configs:
21           - name: spiffe://example.org/amf
22             sds_config:
23               resource_api_version: V3
24               api_config_source:
25                 api_type: gRPC
26                 transport_api_version: V3
27                 grpc_services:
28                   envoy_grpc:
```

```
29                    cluster_name: spire_agent
30  - name: amf
31    connect_timeout: 0.25s
32    type: STRICT_DNS
33    lb_policy: ROUND_ROBIN
34    load_assignment:
35      cluster_name: amf
36      endpoints:
37        - lb_endpoints:
38          - endpoint:
39              address:
40                socket_address:
41                  address: 127.0.0.18
42                  port_value: 8000
43  - name: ausf
44    connect_timeout: 0.25s
45    type: STRICT_DNS
46    lb_policy: ROUND_ROBIN
47    load_assignment:
48      cluster_name: ausf
49      endpoints:
50        - lb_endpoints:
51          - endpoint:
52              address:
53                socket_address:
54                  address: 127.0.2.9
55                  port_value: 8000
56    transport_socket:
57      name: envoy.transport_sockets.tls
58      typed_config:
59        ``@type": type.googleapis.com/envoy.extensions.transport_sockets.
            tls.v3.UpstreamTlsContext
60        common_tls_context:
61          tls_certificate_sds_secret_configs:
62          - name: spiffe://example.org/amf
63            sds_config:
64              resource_api_version: V3
```

```
65                  api_config_source:
66                    api_type: gRPC
67                    transport_api_version: V3
68                    grpc_services:
69                      envoy_grpc:
70                        cluster_name: spire_agent
71      ...
```