UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO
CURSO DE GRADUAÇÃO EM SISTEMAS DE INFORMAÇÃO

Henrique Meireles Costa Gomes

Desenvolvimento de um conversor de mensagens do protocolo CMP à luz da ICP-Brasil

Florianópolis
2023

Henrique Meireles Costa Gomes

Desenvolvimento de um conversor de mensagens do protocolo CMP à luz da ICP-Brasil

Trabalho de Conclusão de Curso de Graduação em Sistemas de informação, do Departamento de Informática e Estatística, do Centro Tecnológico da Universidade Federal de Santa Catarina, requisito parcial à obtenção do título de
Bacharel em Sistemas de Informação.
Orientadora: Profa. Dra. Carla Merkle Westphall

Florianópolis
2023

Ficha de identificação da obra elaborada pelo autor,
através do Programa de Geração Automática da Biblioteca Universitária da UFSC.

Gomes, Henrique Meireles Costa
Desenvolvimento de um conversor de mensagens do protocolo CMP à luz da ICP-Brasil / Henrique Meireles Costa Gomes ; orientadora, Carla Merkle Westphall, 2023.
84 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Centro Tecnológico, Graduação em Sistemas de Informação, Florianópolis, 2023.

Inclui referências.

1. Sistemas de Informação. 2. Certificação digital. 3. Protocolo CMP. 4. Infraestrutura de chaves públicas. 5. Segurança. I. Westphall, Carla Merkle. II. Universidade Federal de Santa Catarina. Graduação em Sistemas de Informação. III. Título.

AGRADECIMENTOS

Aos meus pais, Eduardo Gomes e Wania Maria Costa do Couto Gomes. Por todo apoio durante a minha trajetória, por me mostrar os caminhos, por me incentivar e por confiar em mim. Vocês me fizeram chegar até aqui.

À minha avó Darcy e minha tia Wanda. Por todo suporte e incentivo durante toda minha vida.

À minha amada, Maria Tereza, que me deu calma e apoio durante todo o processo de desenvolvimento deste trabalho.

Aos meus amigos e companheiros de curso: Caroline, Caio, Rafaela, João Vitor B., João Vitor M., Bruno e Lino. Pela parceria e pelos bons momentos, durante toda a graduação.

Ao Roberto Marinho, amigo e gerente do projeto em que trabalho, por me auxiliar nos primeiros passos da minha carreira e por me apresentar o universo da certificação digital.

À orientadora deste projeto, Carla Merkle Westphall, por todo auxílio durante o desenvolvimento deste trabalho.

Henrique Meireles Costa Gomes

Desenvolvimento de um conversor de mensagens do protocolo CMP à luz da ICP-Brasil

O presente trabalho em nível de bacharelado foi avaliado e aprovado por banca examinadora composta pelos seguintes membros:

Prof. Jean Everson Martina, Dr.

Prof. José Eduardo de Lucca

_____

Coordenação do Programa de Graduação
em Sistemas de Informação

_____

Prof[a]. Carla Merkle Westphall, Dra.
Orientador

Florianópolis, 2023.

RESUMO

O mercado de certificados digitais brasileiro é regido por normas e políticas definidas pela Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil), uma cadeia hierárquica de confiança, formada por uma Autoridade Certificadora (AC) raiz, Autoridades Certificadoras credenciadas e Autoridades de Registro (AR). Esta hierarquia viabiliza a emissão de certificados digitais para identificar pessoas físicas e pessoas jurídicas (ITI, 2017).

As entidades pertencentes à ICP-Brasil e os clientes precisam se comunicar durante processos como a emissão, renovação e revogação de certificados. Uma forma de realizar esta comunicação é a troca de mensagens do protocolo CMP, que estabelece um padrão para comunicação na criação e gestão de certificados.

As mensagens CMP consistem basicamente em: Uma estrutura ASN.1 assinada, que contém dados sobre o remetente e o destinatário, além de informações relevantes para a operação em questão.

A implementação deste protocolo em aplicações modernas é um desafio. Sistemas desenvolvidos em linguagens como *JavaScript*, *PHP* e *Python* precisam utilizar módulos ou serviços em outras linguagens, a fim de codificar ou decodificar as mensagens.

Este trabalho tem como objetivo facilitar a criação e o consumo de mensagens CMP, dentro das exigências da ICP-Brasil, por meio do desenvolvimento de uma interface de programação de aplicação (API) *web*, capaz de converter o conteúdo de uma mensagem CMP para uma estrutura no formato JSON e vice-versa, uma vez que esta estrutura é mais simples e possui suporte nativo na maioria das linguagens de programação modernas.

Palavras-chave: Certificação digital, Protocolo CMP, Infraestrutura de chaves públicas, Segurança.

ABSTRACT

The Brazilian digital certificate market is governed by norms and policies defined by the Brazilian Public Key Infrastructure (ICP-Brasil), a hierarchical chain of trust consisting of a root Certification Authority (CA), accredited Certification Authorities, and Registration Authorities (RA). This hierarchy enables the issuance of digital certificates to identify individuals and legal entities (ITI, 2017).

Entities belonging to ICP-Brasil and their clients need to communicate during operations such as certificate issuance, renewal, and revocation. One way to facilitate this communication is through the exchange of messages using the CMP protocol, which establishes a standard for communication in certificate creation and management.

CMP messages consist essentially of a signed ASN.1 structure containing data about the sender and recipient, along with relevant information about the operation.

Implementing this protocol in modern applications poses a challenge. Systems developed in languages like JavaScript, PHP, and Python often need to use modules or services in other languages to encode or decode the messages.

This work aims to facilitate the creation and consumption of CMP messages within the requirements of ICP-Brasil by developing a web Application Programming Interface (API) capable of converting the content of a CMP message to a JSON structure and vice versa. JSON is a simpler structure and has native support in most modern programming languages.

Keywords: Digital certification, CMP protocol, Public key infrastructure, Security.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AC	Autoridade Certificadora
API	*Application Programming Interface*
AR	Autoridade de Registro
ASN.1	*Abstract Syntax Notation One*
CMP	*Certificate Management Protocol*
ICP	Infraestrutura de Chaves Públicas
ICP-Brasil	Infraestrutura de Chaves Públicas Brasileira
ITI	Instituto Nacional de Tecnologia da Informação
JSON	*JavaScript Object Notation*
LCR	Lista de Certificados Revogados
RFC	*Request for Comments*

SUMÁRIO

1.INTRODUÇÃO

O certificado digital é um documento eletrônico que possibilita a identificação de pessoas físicas e jurídicas, a troca segura de informações e a integridade e confidencialidade de mensagens. Além disso, com um certificado dentro dos padrões da Infraestrutura de Chaves Públicas Brasileira, podemos realizar assinaturas digitais com validade jurídica e acessar serviços digitais, como os fornecidos pelo governo federal, de forma prática e segura (ITI, 2022a).

O mercado de certificação digital brasileiro está em crescimento. Por isso, cada vez mais empresas surgem ou se adaptam para atuar neste mercado, seja para operar diretamente na emissão e gestão de certificados, ou para prover soluções para entidades que já atuam na área.

O processo para operar como Autoridade Certificadora ou Autoridade de Registro, pertencente à cadeia da ICP-Brasil, é rigoroso. Existe um processo extenso e burocrático, onde é necessário ser aprovado por auditorias que verificam o cumprimento das exigências definidas pelo ITI.

A comunicação entre Autoridades Certificadoras e Autoridades de Registro é um dos pontos auditados, a troca de informações deve ser feita de forma segura. Uma forma de garantir a segurança na comunicação é utilizar o protocolo CMP (*Certificate Management Protocol*), que define um padrão de mensagens para criação e gestão de certificados. Estas mensagens são assinadas e possuem informações sobre remetente e destinatário, além de dados referentes à transação.

A proposta deste trabalho consiste em implementar uma API capaz de converter estruturas JSON em mensagens CMP e vice-versa, ou seja, criar mensagens CMP a partir de entradas no formato JSON e transformar entradas no formato CMP em estruturas JSON, possibilitando a fácil utilização do protocolo em aplicações modernas baseadas em linguagens que não possuem suporte nativo e bibliotecas que consigam interpretar o protocolo.

1.1. OBJETIVOS

1.1.1. OBJETIVO GERAL

O objetivo geral deste trabalho é desenvolver uma API web para conversão de mensagens CMP. A aplicação deverá ser capaz de receber requisições HTTP e,

baseado no conteúdo recebido, retornar uma estrutura JSON com os dados da mensagem CMP recebida ou criar uma nova mensagem CMP a partir dos dados contidos em uma estrutura JSON recebida. As mensagens CMP geradas deverão suportar algumas estruturas estabelecidas pelo ITI, para atender exigências da ICP-Brasil.

Além disso, para demonstração, as mensagens CMP geradas pelo sistema devem ser interpretadas pelo *software* de Autoridade Certificadora “EJBCA Community Edition”, desenvolvido pela empresa PrimeKey, realizando operações de emissão e revogação de certificados digitais.

1.1.2. OBJETIVOS ESPECÍFICOS

Os objetivos específicos que podem ser listados são os seguintes:

- Explicar o protocolo CMP e a estrutura das mensagens, como especificado no documento RFC 4210 (ADAMS et al., 2005), dentro do domínio do trabalho.
- Conversão do conteúdo de mensagens CMP para estruturas JSON.
- Criação de mensagens CMP a partir de estruturas JSON.
- Utilização das mensagens geradas para realizar operações de emissão e gerenciamento de certificados em um software de Autoridade Certificadora.

1.2. ORGANIZAÇÃO DO TRABALHO

O restante deste trabalho está organizado da seguinte maneira: A seção 2 apresenta conceitos necessários para compreensão do assunto do trabalho e do ambiente em que ele está inserido. As seções 3 e 4 apresentam, respectivamente, trabalhos relacionados e detalhes sobre o desenvolvimento do conversor. A seção 5 apresenta os resultados obtidos. Por fim, a seção 6 apresenta as conclusões sobre o trabalho e mostra possibilidades de melhorias, para trabalhos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta conceitos necessários para o entendimento do trabalho.

2.1. CRIPTOGRAFIA

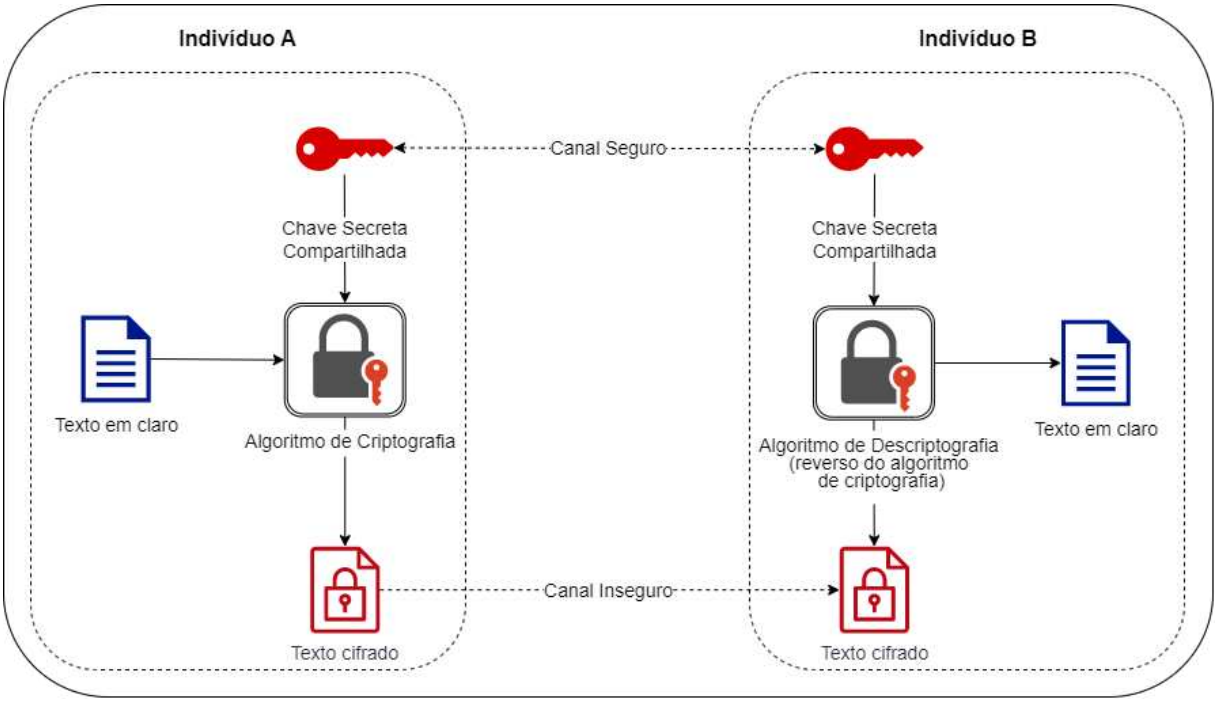
Historicamente, a criptografia surgiu como um meio de permitir que as partes envolvidas mantenham a privacidade das informações que enviam umas às outras, mesmo na presença de um sujeito malicioso com acesso ao canal de comunicação. Embora fornecer privacidade continue sendo um objetivo central, o campo se expandiu para englobar outros, como garantir a integridade e autenticidade das comunicações (BELLARE; ROGAWAEY, 2005). Os dois principais tipos de criptografia são assimétrica e simétrica (IBM Brasil, s.d.).

2.1.1. CRIPTOGRAFIA SIMÉTRICA

A criptografia simétrica utiliza uma única chave (conjunto de valores numéricos utilizados por algoritmos de criptografía) para criptografar e descriptografar dados. Esse sistema necessita de um canal seguro entre as partes envolvidas para transmitir a chave secreta compartilhada, utilizada na operação. As mensagens criptografadas podem ser transmitidas por um canal sem proteção, pois um terceiro malicioso escutando o canal não possui a chave necessária para descriptografar a mensagem e acessar o conteúdo original (FOROUZAN, 2008).

Neste sistema, os indivíduos A e B recebem a chave por meio de um canal seguro. O indivíduo A cifra a mensagem utilizando a chave e envia a mensagem cifrada por um outro canal. O indivíduo B recebe a mensagem cifrada e decifra essa mensagem utilizando a chave, obtendo o conteúdo original cifrado pelo indivíduo A, como mostra a figura 1.

Figura 1 – Criptografia Simétrica



Fonte: O autor (2022).

2.1.2. CRIPTOGRAFIA ASSIMÉTRICA

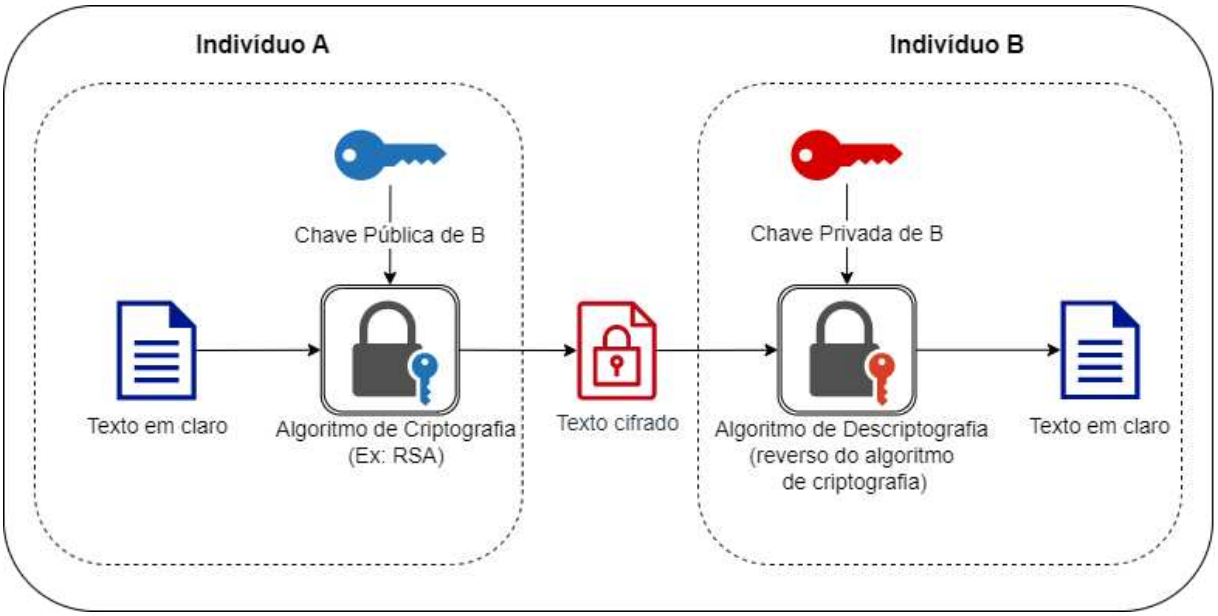
Proposto por Whitfield Diffie e Martin E. Hellman no artigo *New Directions in Cryptography* (DIFFIE; HELLMAN, 1976), se trata de um sistema de criptografia que supera alguns limites existentes na criptografia simétrica, na qual é necessário estabelecer um canal seguro para troca de chaves, o que limita a troca de mensagens criptografadas para a comunicação apenas entre partes que se prepararam previamente para realizá-la.

A criptografia assimétrica, também conhecida como criptografia de chave pública, é baseada em pares de chaves. Um par de chaves é composto por uma chave pública e uma chave privada, sendo a chave privada secreta, ou seja, conhecida apenas pelo seu dono, e a chave pública aberta, que pode ser acessada por outros.

Por meio de funções matemáticas utilizadas na geração do par de chaves, é possível garantir que não é possível descobrir a chave privada a partir da chave pública e que, uma mensagem cifrada por uma chave pública só pode ser decifrada pela chave privada correspondente e vice-versa.

Assim, caso o indivíduo A deseje enviar uma mensagem privada que pode ser lida apenas pelo indivíduo B, ele deve cifrar a mensagem com a chave pública do indivíduo B. Para ler a mensagem, o indivíduo B deve decifrá-la utilizando sua própria chave privada, como mostra a figura 2.

Figura 2 – Criptografia Assimétrica



Fonte: O autor (2022).

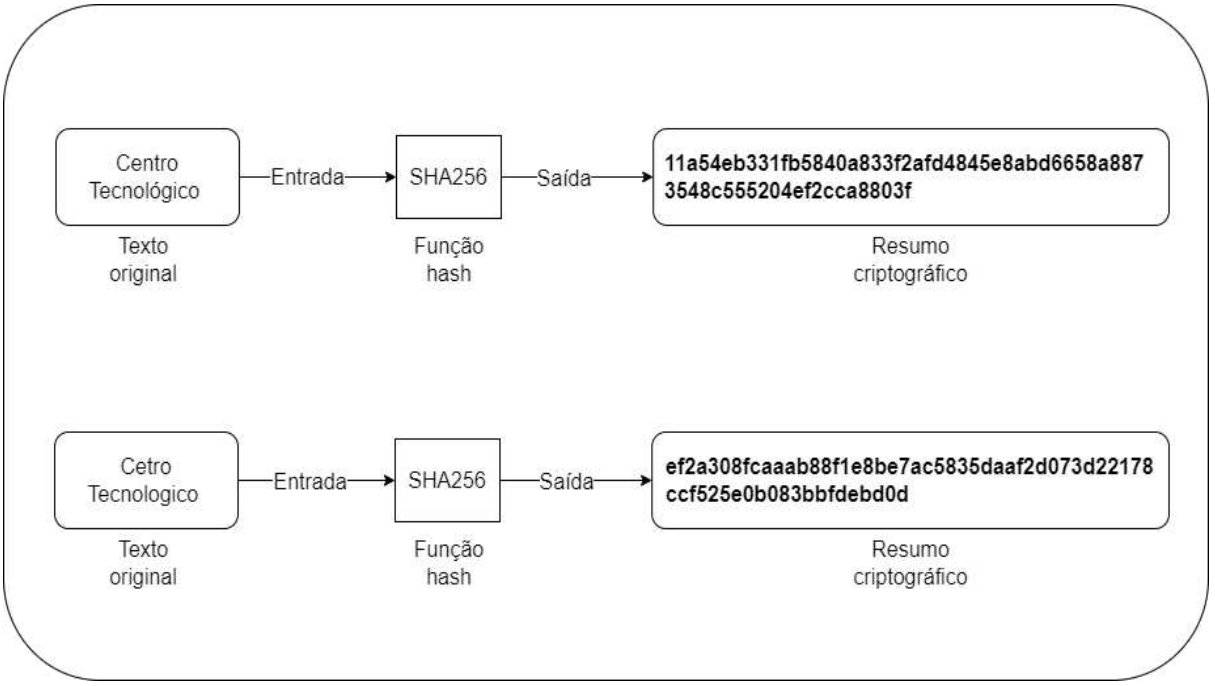
2.1.3. RESUMO CRIPTOGRÁFICO

Resumo criptográfico, ou *hash*, se trata de um conteúdo gerado por uma função *hash*. Este tipo de função recebe como entrada um bloco de dados de tamanho variável e produz uma saída de tamanho fixo. O objetivo do resumo criptográfico é garantir a integridade dos dados. Uma mudança em qualquer bit do conteúdo da entrada resulta, com grandes chances, em uma mudança no resumo criptográfico (STALLINGS, 2013).

Um algoritmo de *hash* utilizado para criptografia é conhecido como função hash criptográfica. Algoritmos deste tipo também devem garantir que é computacionalmente inviável recriar o conteúdo original a partir do seu *hash* e gerar uma colisão (produzir duas entradas que resultam no mesmo *hash*).

A figura 3 mostra o resultado da execução da função *hash* SHA256 recebendo como entrada dois textos parecidos.

Figura 3 – Exemplo SHA256



Fonte: O autor (2022).

No exemplo acima, as entradas, que possuem pequenas diferenças no tamanho e no conteúdo, são submetidas à mesma função *hash* e geraram saídas distintas, porém de mesmo tamanho.

2.1.4. ASSINATURA DIGITAL

A criptografia simétrica resolve problemas relacionados à interceptação de mensagens trocadas em canais inseguros, porém não resolve problemas relacionados à falsificação e adulteração do conteúdo original (IBM, 2022).

Podemos destacar dois cenários preocupantes:

- O indivíduo B pode forjar uma mensagem e alegar que a mesma partiu do indivíduo A, ele simplesmente precisa cifrar o conteúdo usando a chave que os dois indivíduos compartilham.
- O indivíduo A pode alegar não ter enviado uma mensagem, pois a mesma pode ser forjada pelo indivíduo B.

Um exemplo para primeiro cenário: Em uma transação financeira eletrônica, o receptor aumenta a quantia recebida e alega que o remetente enviou aquele valor.

Um exemplo para o segundo cenário: Um indivíduo dá instruções para um corretor de ações, a operação tem um balanço negativo e o indivíduo alega nunca ter dado as instruções (STALLINGS, 2013).

A assinatura digital é uma solução para situações como essas, onde não há plena confiança na outra parte. A assinatura digital deve possuir as seguintes propriedades: Deve checar o autor e a data e hora da assinatura, deve autenticar o conteúdo no momento da assinatura e deve ser verificável por terceiros, para resolver disputas. Baseado nisso podemos definir requisitos para uma assinatura digital (STALLINGS, 2013):

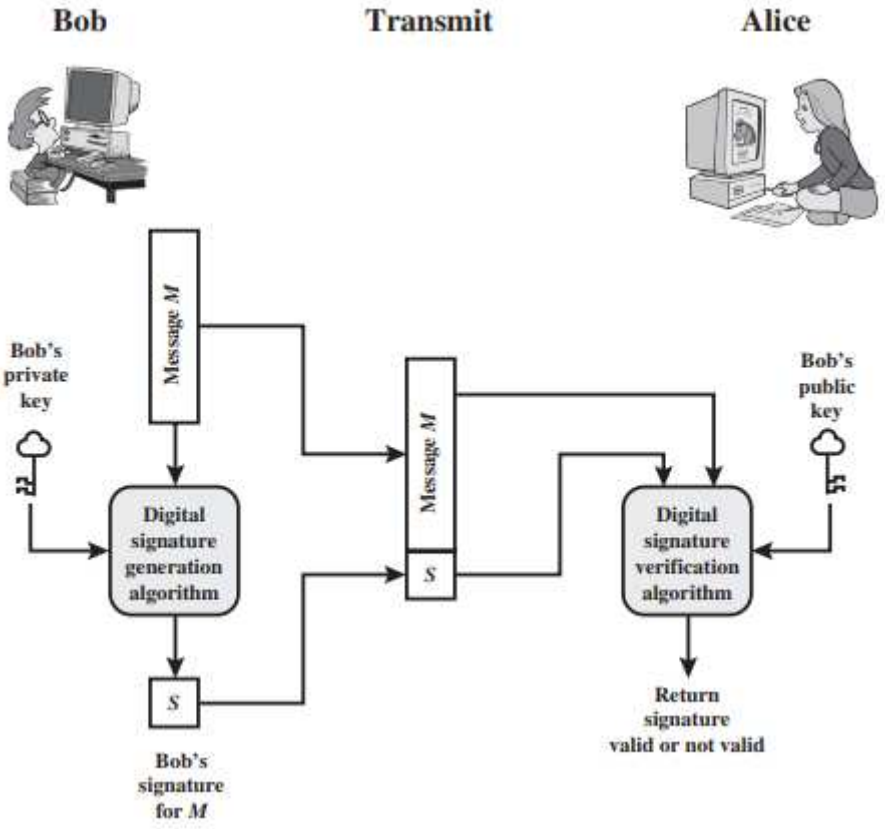
- A assinatura deve ser um padrão de bits que depende do conteúdo assinado.
- A assinatura deve possuir uma informação única do remetente, para evitar falsificações e negação da assinatura.
- Deve ser relativamente fácil gerar uma assinatura digital.
- Deve ser relativamente fácil reconhecer e verificar uma assinatura digital.
- Deve ser computacionalmente inviável falsificar uma assinatura criando um novo conteúdo para uma assinatura existente ou criar uma assinatura fraudulenta para um conteúdo existente.
- Guardar uma cópia de uma assinatura digital em um armazenamento deve ser algo prático.

Uma função segura de resumo criptográfico é uma base para satisfazer esses requisitos. A assinatura é, basicamente, o ato de usar a chave privada para cifrar o resumo criptográfico do conteúdo original, ela consiste no hash criptografado e informações como os algoritmos utilizados para criptografia e *hash* (IBM, 2022).

A verificação da assinatura consiste, basicamente, em decifrar o resumo criptográfico com a chave pública do assinante e comparar o resumo criptográfico assinado com o resumo criptográfico do conteúdo original.

A figura 4 demonstra esses processos.

Figura 4 – Modelo genérico do processo de assinatura digital



Fonte: (STALLINGS, 2013).

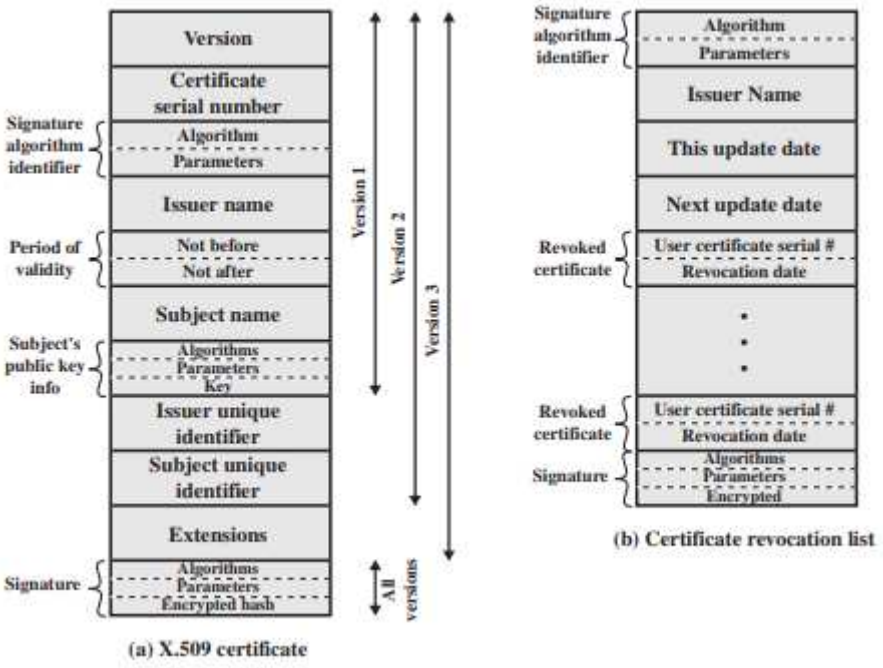
2.2. CERTIFICADO X.509

Também conhecido como certificado digital ou certificado de chave pública, é um arquivo eletrônico assinado por uma Autoridade Certificadora (AC) que contém, além da assinatura, uma chave pública, informações sobre a identidade do dono do par de chaves e informações sobre o emissor do certificado (STALLINGS, 2013).

X.509 é um padrão internacional que define uma estrutura para os certificados de chave pública e listas de certificados revogados (ITU-T, 2008), utilizadas para avisar a revogação dos certificados.

A Figura 5 mostra a estrutura de um certificado digital e de uma lista de revogação. A Infraestrutura de Chaves Públicas Brasileira utiliza a versão 3 do certificado X.509.

Figura 5 – Certificado X.509



Fonte: (STALLINGS, 2013)

Os elementos presentes no certificado são:

- ***Version*****:** Diferencia a versão do formato do certificado. Caso "*issuer unique identifier*" ou "*subject unique identifier*" estejam presentes, o valor deve ser version 2, caso possua outras extensões a versão deve ser version 3.
- ***Serial number*****:** Um valor inteiro e único para a AC emissora que esteja associado, de forma não-ambígua, ao certificado.
- ***Signature algorithm identifier*****:** O algoritmo utilizado para assinar o certificado e os parâmetros associados.
- ***Issuer name*****:** Nome X.500 da AC que emitiu e assinou o certificado.
- ***Period of validity*****:** Consiste em duas datas: "*not before"* e *"not after",* a primeira e última data em que o certificado é válido.
- ***Subject name*****:** Nome X.500 da entidade final, ou seja, aquele que o certificado se refere, o dono da chave privada correspondente a chave pública contida no certificado.
- ***Subject's public-key information*****:** A chave pública da entidade final, junto do identificador do algoritmo para o qual a chave deve ser usada e outros parâmetros associados.

- ***Issuer unique identifier*:** Um campo *bit-string* opcional usado para identificar, de forma única, a AC emissora, caso o nome X.500 tenha sido reutilizado por outras entidades.
- ***Subject unique identifier*:** Um campo *bit-string* opcional usado para identificar, de forma única, a entidade final, caso o nome X.500 tenha sido reutilizado por outras entidades.
- ***Extensions*:** Conjunto de um ou mais campos de extensão.
- ***Signature*:** Assinatura de todos os outros campos do certificado, contém o resumo criptográfico dos outros campos criptografados pela chave privada da AC emissora. Também inclui o identificador do algoritmo utilizado para a assinatura.

O certificado digital, basicamente, atribui uma identidade à uma chave pública. Essa identidade não precisa ser a de uma pessoa física, também é possível emitir certificados para pessoas jurídicas (ITI, 2022a). Além disso, certificados podem ser utilizados por sites, servidores e aplicações para realizar a comunicação segura entre cliente e servidor, por meio de protocolos como o TLS (protocolo de segurança criado para fornecer a comunicação segura).

Por incluir a assinatura do emissor, outras entidades podem verificar a autenticidade do certificado utilizando a chave pública da entidade emissora. Assim, caso confiem no emissor, podem confiar no certificado.

Caso ocorra um roubo, vazamento da chave privada ou outro acontecimento, o dono do certificado pode solicitar a revogação do certificado. A entidade emissora é responsável por publicar a revogação do certificado em uma LCR (Lista de Certificados Revogados), tornando inválidas as operações realizadas pelo certificado a partir daquele momento.

Para um certificado digital possuir validade jurídica reconhecida pela justiça brasileira, ele deve ser emitido por uma Autoridade Certificadora que faz parte da hierarquia da Infraestrutura de Chaves Públicas Brasileira (ICP-Brasil). Desta forma é possível afirmar que as entidades envolvidas na emissão do certificado passaram por processos de regulamentação e auditorias e, por isso, são aptas para garantir a validade dos dados contidos nos certificados que emitem.

Com um certificado digital ICP-Brasil é possível realizar ações como: assinar documentos digitalmente, autenticar-se em serviços do governo brasileiro e de outras entidades, emitir notas fiscais eletrônicas, entre outros (ITI, 2020a).

2.3. INFRAESTRUTURAS DE CHAVES PÚBLICAS

Uma ICP é composta por hardwares, softwares, pessoas, políticas e procedimentos necessários para criar, gerenciar, distribuir, usar, armazenar e revogar certificados digitais (TOORANI e SHIRAZI, 2008).

Dentre as entidades que compõem uma ICP, podemos destacar Autoridades Certificadoras e Autoridades de Registro.

Estas entidades são descritas pelos autores Chokhani, Ford, Sabett, Merrill, e Wu (2003) da seguinte forma:

- Autoridade Certificadora: Responsável por emitir os certificados digitais. Ela é a entidade emissora em relação aos certificados que emite e é a entidade final em relação a AC que emitiu seu certificado. As ACs estão organizadas hierarquicamente, onde ACs de um nível maior emitem certificados para outras ACs que emitem certificados para outras Entidades Finais.
- Autoridade de Registro: Entidades que estabelecem os procedimentos para solicitação de certificado, identificam e autenticam os solicitantes, iniciam ou enviam solicitações de revogação e aprovam ou reprovam solicitações de emissão e renovação de certificados.

As entidades que solicitam e recebem certificados são conhecidas como Entidades Finais ou como Solicitantes. No contexto da ICP-Brasil os solicitantes mais comuns são pessoas físicas e pessoas jurídicas, porém também é possível emitir certificados para outros fins como por exemplo, certificados digitais para objetos metrológicos, como bombas de gasolina.

2.3.1. ICP-BRASIL

Segundo o Instituto Nacional de Tecnologia da Informação (ITI, 2017), a ICP-Brasil segue o modelo de raiz única, onde uma AC, denominada AC-Raiz, é a primeira autoridade da cadeia de certificação. Esta AC é operada pelo próprio ITI, além de exercer o papel de AC raiz, o órgão é responsável por credenciar e

descredenciar os demais participantes da cadeia, supervisionar e auditar os processos.

Além de ACs e ARs, a ICP-Brasil é formada, também, por Autoridades Certificadoras do Tempo - ACTs, Prestadores de Serviço Biométrico - PSBios, Prestadores de Serviço de Suporte - PSS e por uma autoridade gestora de políticas, ou seja, o Comitê Gestor da ICP-Brasil (ITI, 2020b).

Uma Autoridade Certificadora do Tempo (ACT) é uma entidade confiável, responsável por emitir carimbos do tempo. Com o carimbo do tempo associado à uma assinatura digital, é possível provar a existência da assinatura em um determinado período.

Prestadores de Serviço Biométricos (PSBios) são entidades com capacidade de realizar a identificação biométrica de solicitantes, tornando um solicitante único nos bancos/sistemas de dados biométricos para toda ICP-Brasil. Possibilita a verificação biométrica do solicitante e a comparação biométrica, de acordo com os padrões internacionais de uso.

Um Prestador de Serviço de Suporte (PSS) pode exercer atividades de outras entidades da ICP como PSBios e ACTs. Ele se classifica conforme a atividade prestada dentre três categorias: disponibilização de infraestrutura física e lógica; disponibilização de recursos humanos especializados; ou disponibilização de infraestrutura física e lógica e de recursos humanos especializados.

O Comitê Gestor da ICP-Brasil, de acordo com o Decreto 6.605 de 14 de outubro de 2008, exerce a função de autoridade gestora de políticas de certificação digital. A entidade é vinculada à Casa Civil da Presidência da República e é composta por sete representantes governamentais e cinco representantes da sociedade civil, sendo todos estes designados pelo Presidente da República.

2.4. MERCADO DE CERTIFICAÇÃO DIGITAL

O mercado de certificação digital brasileiro está em crescimento. Segundo o Instituto Nacional de Tecnologia da Informação (ITI), mais de 690 mil certificados pertencentes à hierarquia da ICP-Brasil foram emitidos somente no mês de junho do ano de 2021, cerca de 178 mil certificados a mais quando comparado ao mesmo período de 2020. O ano de 2021 terminou com mais de 7 milhões de certificados emitidos e a previsão para 2022 é de mais de 8 milhões e 600 mil certificados. (Infor

Channel, 2021; ITI, 2022b; Terra, 2022). Por causa de números como esses, cada vez mais empresas surgem ou se adaptam para atuar neste mercado, seja para operar diretamente na emissão e gestão de certificados, ou para prover soluções para entidades que já atuam na área.

2.5. PROTOCOLO CMP

O *Certificate Management Protocol* (CMP) é um protocolo para criação e gerenciamento de certificados X.509v3. Ele proporciona a interação online entre entidades da ICP, incluindo a comunicação entre uma autoridade certificadora e um sistema cliente (ADAMS et al., 2005).

O protocolo CMP implementa 27 tipos de mensagens utilizadas para comunicação das entidades e gestão de certificados digitais. Existem mensagens para gerar, renovar, atualizar e revogar certificados digitais.

Todas as mensagem CMP, especificadas na RFC 4210 (ADAMS et al., 2005), possuem a estrutura representada na figura 6:

Figura 6 – Estrutura PKIMessage

```
PKIMessage ::= SEQUENCE {
    header           PKIHeader,
    body             PKIBody,
    protection   [0] PKIProtection OPTIONAL,
    extraCerts   [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                     OPTIONAL
}
```

Fonte: (ADAMS et al., 2005).

O *header* contém dados comuns entre as mensagens do protocolo. A figura 7 descreve a sua estrutura.

Figura 7 – Estrutura PKIHeader

```
PKIHeader ::= SEQUENCE {
    pvno                INTEGER     { cmp1999(1), cmp2000(2) },
    sender              GeneralName,
    recipient           GeneralName,
    messageTime     [0] GeneralizedTime         OPTIONAL,
    protectionAlg   [1] AlgorithmIdentifier     OPTIONAL,
    senderKID       [2] KeyIdentifier           OPTIONAL,
    recipKID        [3] KeyIdentifier           OPTIONAL,
    transactionID   [4] OCTET STRING            OPTIONAL,
    senderNonce     [5] OCTET STRING            OPTIONAL,
    recipNonce      [6] OCTET STRING            OPTIONAL,
    freeText        [7] PKIFreeText             OPTIONAL,
    generalInfo     [8] SEQUENCE SIZE (1..MAX) OF
                        InfoTypeAndValue     OPTIONAL
}
```

Fonte: (ADAMS et al., 2005)

As informações contidas nesta estrutura são utilizadas para indicar a chave do remetente e destinatário, identificar o algoritmo utilizado para proteger a mensagem (caso exista uma proteção), identificar a transação, prevenir ataques de repetição e enviar informações que podem ser úteis ao destinatário. A obrigatoriedade de preenchimento de alguns campos depende do contexto da mensagem e da presença ou ausência de outros campos.

O *body* especifica qual o tipo da mensagem. O conteúdo da mensagem varia de acordo com o tipo escolhido. A figura 8 mostra as opções existentes.

Figura 8 – Estrutura PKIBody

```
PKIBody ::= CHOICE {
        ir       [0]  CertReqMessages,       --Initialization Req
        ip       [1]  CertRepMessage,        --Initialization Resp
        cr       [2]  CertReqMessages,       --Certification Req
        cp       [3]  CertRepMessage,        --Certification Resp
        p10cr    [4]  CertificationRequest,  --PKCS #10 Cert.  Req.
        popdecc  [5]  POPODecKeyChallContent --pop Challenge
        popdecr  [6]  POPODecKeyRespContent, --pop Response
        kur      [7]  CertReqMessages,       --Key Update Request
        kup      [8]  CertRepMessage,        --Key Update Response
        krr      [9]  CertReqMessages,       --Key Recovery Req
        krp      [10] KeyRecRepContent,      --Key Recovery Resp
        rr       [11] RevReqContent,         --Revocation Request
        rp       [12] RevRepContent,         --Revocation Response
        ccr      [13] CertReqMessages,       --Cross-Cert.  Request
        ccp      [14] CertRepMessage,        --Cross-Cert.  Resp
        ckuann   [15] CAKeyUpdAnnContent,    --CA Key Update Ann.
        cann     [16] CertAnnContent,        --Certificate Ann.
        rann     [17] RevAnnContent,         --Revocation Ann.
        crlann   [18] CRLAnnContent,         --CRL Announcement
        pkiconf  [19] PKIConfirmContent,     --Confirmation
        nested   [20] NestedMessageContent,  --Nested Message
        genm     [21] GenMsgContent,         --General Message
        genp     [22] GenRepContent,         --General Response
        error    [23] ErrorMsgContent,       --Error Message
        certConf [24] CertConfirmContent,    --Certificate confirm
        pollReq  [25] PollReqContent,        --Polling request
        pollRep  [26] PollRepContent         --Polling response
}
```

Fonte: (ADAMS et al., 2005).

2.5.1. ASN.1

As mensagens do protocolo CMP são, basicamente, estruturas ASN.1.

Segundo o *ITU Telecommunication Standardization Sector* (ITU-T, s.d.), ASN.1 se trata de uma notação utilizada para descrever dados transmitidos por protocolos de telecomunicações, independentemente da implementação de linguagem e da representação física desses dados, independente da complexidade da aplicação. A notação fornece alguns tipos básicos pré-definidos, por exemplo: inteiros, booleanos, *strings* de bits, *strings* de caracteres, entre outros. Também possibilita a definição de tipos construídos, como: estruturas, listas, escolha entre tipos, entre outros.

Além disso, a notação ASN.1 oferece extensibilidade que aborda o problema e fornece suporte para a interoperabilidade entre sistemas previamente implantados e versões mais recentes, projetadas anos depois.

2.5.2. OPERAÇÕES DE GERENCIAMENTO DE CERTIFICADO

2.5.2.1. EMISSÃO DE CERTIFICADO

Para emitir um certificado, a AR deve enviar para a AC uma mensagem para solicitação de certificação (*Certification Req*), o tipo 2 na Figura 8. A *Certification Req* tem a estrutura exibida na figura 9.

Figura 9 – Estrutura CertReqMessages

```
CertReqMessages ::= SEQUENCE SIZE (1..MAX) OF CertReqMsg

CertReqMsg ::= SEQUENCE {
   certReq   CertRequest,
   popo       ProofOfPossession  OPTIONAL,
   -- content depends upon key type
   regInfo   SEQUENCE SIZE(1..MAX) of AttributeTypeAndValue OPTIONAL
}
```

Fonte: (SCHAAD, 2005).

A mensagem consiste, basicamente, em uma lista de *CertReqMsg*.

Uma *CertReqMsg* é formada por (SCHAAD, 2005):

- **certReq**: Contém o modelo do certificado solicitado, preenchido com os dados enviados pela entidade final.

- **popo**: Se refere a prova de posse. Ele contém o valor utilizado para demonstrar que a entidade final, o sujeito do certificado solicitado, realmente possui a chave privada correspondente, permitindo a AC ou AR validar o vínculo entre o sujeito e o par de chaves. Este campo existe para prevenir alguns tipos de fraude, como a emissão de certificados para pessoas diferentes contendo a mesma chave pública.
- **regInfo**: Deve conter apenas informações complementares relativas ao contexto da solicitação de certificado, onde essas informações são necessárias para atender a solicitação. Pode incluir informações de contato, cobrança ou outras informações auxiliares que são úteis para completar a requisição.

A figura 10 mostra a estrutura do CertRequest.

Figura 10 – Estrutura CertRequest

```
CertRequest ::= SEQUENCE {
   certReqId     INTEGER,        -- ID for matching request and reply
   certTemplate  CertTemplate, --Selected fields of cert to be issued
   controls      Controls OPTIONAL } -- Attributes affecting issuance

CertTemplate ::= SEQUENCE {
   version      [0] Version               OPTIONAL,
   serialNumber [1] INTEGER               OPTIONAL,
   signingAlg   [2] AlgorithmIdentifier   OPTIONAL,
   issuer       [3] Name                  OPTIONAL,
   validity     [4] OptionalValidity      OPTIONAL,
   subject      [5] Name                  OPTIONAL,
   publicKey    [6] SubjectPublicKeyInfo  OPTIONAL,
   issuerUID    [7] UniqueIdentifier      OPTIONAL,
   subjectUID   [8] UniqueIdentifier      OPTIONAL,
   extensions   [9] Extensions            OPTIONAL }

OptionalValidity ::= SEQUENCE {
   notBefore  [0] Time OPTIONAL,
   notAfter   [1] Time OPTIONAL } --at least one must be present

Time ::= CHOICE {
   utcTime        UTCTime,
   generalTime    GeneralizedTime }
```

Fonte: (SCHAAD, 2005).

O CertTemplate contido na estrutura é um modelo de um certificado X.509v3, como foi descrito na seção 1.2. Estes são os dados que estarão presentes no certificado emitido.

A comunicação, basicamente, ocorre da seguinte forma: A entidade final gera um par de chaves e solicita um certificado, enviando as informações necessárias para a AR (informações da entidade e chave pública).

A AR realiza a verificação e, se necessário, adiciona ou modifica algumas informações. Após verificar, a AR envia um desafio para ser assinado pela chave privada, par da chave pública que foi enviada pela entidade final no momento da solicitação.

Então, a entidade final assina o desafio e envia a assinatura para a AR. A AR verifica se a assinatura é válida. Esta etapa é a prova de posse.

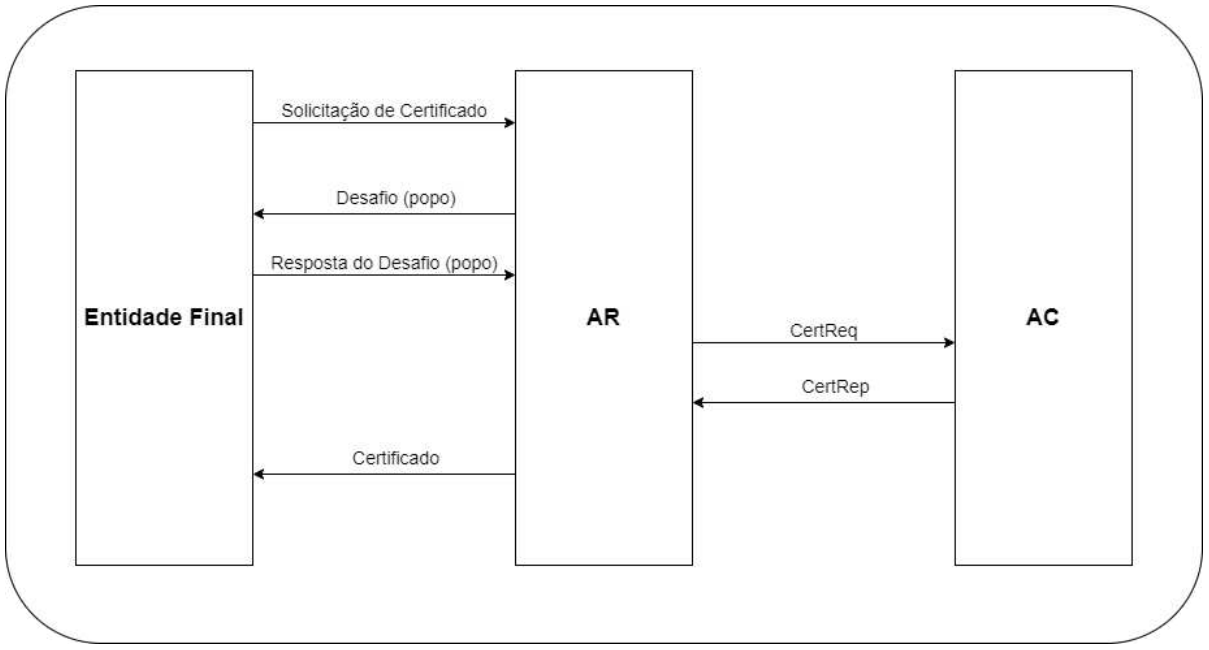
A AR envia uma mensagem CMP do tipo *Certification Req* para a AC.

A AC emite o certificado e retorna uma mensagem CMP do tipo *Certification Rep* para a AR.

Por fim, a AR retorna o certificado emitido para a entidade final.

A figura 11 ilustra esta comunicação.

Figura 11 – Comunicação durante emissão de certificado



Fonte: O autor (2022).

2.5.2.2. REVOGAÇÃO DE CERTIFICADO

Para revogar um certificado, a AR deve enviar para a AC uma mensagem para solicitação de revogação (*Revocation Request*), o tipo 11 na Figura 8. A *Revocation Request* tem a seguinte estrutura (ADAMS et al., 2005):

Figura 12 – Estrutura RevReqContent

```
RevReqContent ::= SEQUENCE OF RevDetails

RevDetails ::= SEQUENCE {
    certDetails         CertTemplate,
    crlEntryDetails     Extensions       OPTIONAL
}
```

Fonte: (ADAMS et al., 2005).

A mensagem consiste, basicamente, em uma lista de *RevDetails*.

Um *RevDetails* é formado por (ADAMS et al., 2005):

- **certDetails**: Permite que o solicitante especifique o quanto puder sobre o certificado que deseja revogar.
- **crlEntryDetails**: Extensões opcionais com detalhes sobre a entrada do certificado na LCR

A AC deve responder a AR com uma *Revocation Response*, o tipo 12 na figura 8. A *Revocation Response* tem a seguinte estrutura (ADAMS et al., 2005):

Figura 13 – Estrutura RevRepContent

```
RevRepContent ::= SEQUENCE {
    status          SEQUENCE SIZE (1..MAX) OF PKIStatusInfo,
    revCerts    [0] SEQUENCE SIZE (1..MAX) OF CertId OPTIONAL,
    crls        [1] SEQUENCE SIZE (1..MAX) OF CertificateList
                    OPTIONAL
}
```

Fonte: (ADAMS et al., 2005).

O *RevRepContent* é formado por:

- **status**: Status de cada revogação solicitada, na ordem em que foi enviado no RevReqContent.
- **revCerts**: Identificador dos certificados que tiveram a revogação solicitada no pedido, na mesma ordem do status. Campo opcional.
- **crls**: As LCRs que podem ter sido geradas após revogação, pode haver mais de uma. Campo opcional.

2.5.2.3. OUTRAS MENSAGENS

Além das mensagens citadas nas seções 1.5.2 e 1.5.3, o protocolo CMP fornece mensagens para as seguintes finalidades (HOUSLEY; POLK, 2001):

- **Mensagens de requisição e resposta para certificação cruzada**: Estas mensagens devem ser utilizadas para solicitar e entregar certificados para ACs.
- **Mensagens de requisição e resposta para recuperação de chaves**: Estas mensagens devem ser utilizadas para solicitar e entregar chaves privadas usadas para o gerenciamento de chaves caso tenham respaldo da ICP.
- **Mensagens para desafio e resposta de prova de posse**: Estas mensagens podem ser utilizadas para realizar a prova de posse de chaves privadas utilizadas para concordância de chaves ou transporte de chaves.
- **Mensagens para distribuição de certificado e LCR**: Mensagens que podem ser utilizadas para anunciar a emissão de um certificado ou LCR.
- **Mensagens diversas**: Estas mensagens incluem uma mensagem de confirmação, uma mensagem de erro, uma mensagem de solicitação geral e mensagens aninhadas para suportar múltiplos assinantes.

2.5.3. TRANSPORTE DAS MENSAGENS

Como especificado no documento RFC 6712 (KAUSE; PEYLO, 2012), as mensagens do protocolo CMP são transportadas via HTTP.

A utilização deste protocolo para transportar as mensagens CMP usa exclusivamente o método POST. Uma PKIMessage codificada como DER (*Distinguished Encoding Rules*), uma codificação binária para estruturas utilizadas na certificação, deve ser enviada no corpo da requisição HTTP.

Em caso de sucesso da solicitação, o servidor deve enviar uma mensagem de resposta CMP no corpo de uma resposta HTTP com o *status code* 200. Respostas HTTP para mensagens CMP de anúncio utilizam o status code 201 ou 202, para identificar se informação recebida foi processada. Os status codes de erro 4xx e 5xx podem ser utilizados para informar erros ao cliente.

Deve-se utilizar o *header* HTTP “*Content-Type*” com o valor “application/pkixcmp” ao transportar uma mensagem CMP.

A utilização do HTTP possui alguns riscos:

- Existe o risco de sofrer ataques de negação de serviço, devido ao consumo de recursos causados por múltiplas conexões ao servidor HTTP. Por isso, conexões ociosas devem ser terminadas depois de um *timeout* apropriado. Ao enviar uma mensagem CMP de erro, o servidor deve fechar a conexão, ainda que a transação não tenha sido completada.
- Ao utilizar o HTTP sem encapsula-lo em protocolos de segurança, como o TLS (*Transport Layer Security*), não há proteção de integridade. Portanto, informações do protocolo HTTP não devem ser usadas para alterar o estado da transação.

3. TRABALHOS RELACIONADOS

Esta seção apresenta alguns trabalhos que possuem correlação com esta monografia. Os trabalhos foram escolhidos pois utilizam e/ou analisam o protocolo CMP.

3.1. AN ECC BASED PUBLIC KEY INFRASTRUCTURE USABLE FOR MOBILE APPLICATIONS

Em Ray et al. (2012), é apresentado um modelo de Infraestrutura de Chaves Públicas (ICP) baseado em criptografia de curvas elípticas voltado para o universo de dispositivos móveis.

Na época, o número e complexidade dos aplicativos para dispositivos móveis estavam aumentando e se tornou necessário projetar ou melhorar a infraestrutura de chaves públicas para dispositivos móveis.

Então, os autores propõem uma ICP baseada em criptografia de curvas elípticas que contempla as limitações dos dispositivos móveis, como: tela pequena, menor poder computacional e menor capacidade de armazenamento.

Neste modelo, utiliza-se uma entidade denominada MHA (Mobile Home Agent) que se responsabiliza por fazer todas as operações criptográficas e comunicação com entidades da ICP, tirando essas tarefas do dispositivo móvel.

Para realizar a comunicação com as entidades da ICP, os autores utilizam o CMP (Certificate Management Protocol), o mesmo protocolo que é abordado no Trabalho de Conclusão de Curso em desenvolvimento. Porém, devido ao contexto, ele utiliza a versão WCMP, uma implementação do protocolo voltada para ICP wireless, proposto por Y. Lee et al (2008).

3.2. USING LDAP DIRECTORIES FOR MANAGEMENT OF PKI PROCESSES

Em Karatsiolis et al. (2004), os autores apresentam um *framework* para estender a funcionalidade de servidores LDAP (*Lightweight Directory Access Protocol*) no contexto de Infraestrutura de Chaves Públicas (ICP), onde são utilizados como repositórios de certificados e CRLs (*Certificate Revocation List*), criando suporte para gestão de processos da ICP implementando, inclusive, um método de realizar a prova de posse de chaves criptográficas reduzindo a quantidade de mensagens trocadas, uma vez que não precisa enviar confirmações.

No texto ele descreve as operações de solicitação de certificado, conexão segura e prova de posse, mostrando que é viável prover a prova de posse via LDAP criando, junto como registro no ldap, uma senha e um certificado, cifrados pela chave pública do solicitante, são armazenados no registro, para ter acesso a senha o solicitante deve descriptografar a senha e utilizar a mesma para se autenticar no repositório e adicionar o certificado, completando a prova de posse. Uma outra opção seria criptografar apenas uma parte da senha e ter a outra parte fornecida ao usuário durante o cadastro criando um segredo compartilhado.

Os autores fazem comparativos com outros métodos, incluindo o CMP e citam como ponto negativo do CMP a necessidade de enviar mensagens de confirmação. Por fim, concluem que os modelos são válidos e divulgam que pretendem implementar a ICP da Universidade de Darmstadt.

3.3. ANÁLISE E IMPLEMENTAÇÃO DE UM PROTOCOLO DE GERENCIAMENTO DE CERTIFICADOS

Em Silvério (2011), o autor se propôs a realizar a análise e implementação de um protocolo de gerenciamento de certificados para ser utilizado pelo Sistema de Gerenciamento de Certificados Digitais da Infraestrutura de Chaves Públicas para Ensino e Pesquisa (SGCI) que, quando o trabalho foi desenvolvido, possuía um protocolo próprio que não garantia a integridade e autenticidade das mensagens trocadas entre AC e AR.

Para isso, o autor realizou a implementação das mudanças necessárias no SGCI e implementou uma biblioteca para gerar e manipular mensagens do protocolo a partir de estruturas XML, definindo como limitações do trabalho a utilização do modelo X.509, o foco apenas na comunicação entre Autoridade de Registro e Autoridade Certificadora e a abordagem apenas das mensagens utilizadas no processo de emissão e revogação de certificados .

Foi feita uma análise sobre o protocolo que seria utilizado, após comparar o protocolo CMP e o protocolo CMC (Certificate Management over CMS) concluiu-se que o CMP é o protocolo mais adequado.

No texto, o autor explica fundamentos sobre criptografia, Infraestruturas de Chaves Públicas e sobre a estrutura das mensagens implementadas. Ele obteve êxito em implementar o protocolo e adaptar o sistema dentro do que foi proposto.

Como trabalhos futuros, o autor propõe a implementação de um módulo público para comunicação entre o usuário final e a Autoridade de Registro, a implementação das outras operações do protocolo CMP, formalização das estruturas adicionadas ao protocolo e do uso do XML para codificação das mensagens e, por fim, um estudo sobre como aumentar a segurança das chaves privadas de entidades que operam *online*.

3.4. CONSIDERAÇÕES SOBRE OS TRABALHOS

De maneira geral, os trabalhos apresentam o uso do protocolo CMP em diferentes contextos. Os contextos apresentados foram, também, motivadores para o desenvolvimento deste trabalho, pois apresentam cenários onde o conversor poderia ser aplicado para facilitar a criação e o processamento das mensagens, principalmente o trabalho citado na seção 3.1, que é voltado para aplicações *mobile*.

4. DESENVOLVIMENTO DE UM CONVERSOR DE MENSAGENS CMP

Esta seção descreve as tecnologias utilizadas e o processo de desenvolvimento do sistema.

4.1. TECNOLOGIAS DE DESENVOLVIMENTO

O sistema sugerido é uma API desenvolvida na linguagem Java com o uso do framework Spring Boot. A biblioteca *Bouncy Castle* foi utilizada para gerar as mensagens CMP. Para testar as mensagens, o *software* EJBCA foi utilizado para emitir e revogar certificados digitais.

As subseções a seguir apresentam e justificam as tecnologias utilizadas.

4.1.1. JAVA

A linguagem Java foi escolhida para o desenvolvimento do trabalho devido a familiaridade do autor com a utilização da mesma e a disponibilidade da biblioteca *Bouncy Castle*. Além disso, existe uma grande quantidade de suporte disponível para a linguagem.

4.1.2. BOUNCY CASTLE

De acordo com *The Legion of Bouncy Castle* (s.d.), a *Bouncy Castle* consiste em um conjunto de bibliotecas de código aberto disponível para as linguagens de programação Java e C#. A biblioteca disponibiliza diversas APIs e ferramentas para criptografia, incluindo ferramentas que possibilitam a geração e processamento das mensagens do protocolo CMP.

4.1.3. FRAMEWORK SPRING BOOT

O Spring Boot é um *framework* para a linguagem Java. Ele disponibiliza ferramentas como injeção de dependências, filtro e validação de requisições, tratamento de erros e ambiente de testes. Além disso, ele cuida automaticamente de grande parte da configuração da aplicação web.

4.1.4. EJBCA

EJBCA é um *software* de ICP de código aberto que suporta mensagens CMP. Com ele, foi possível criar uma ICP de teste para realizar operações como emitir e revogar certificados digitais.

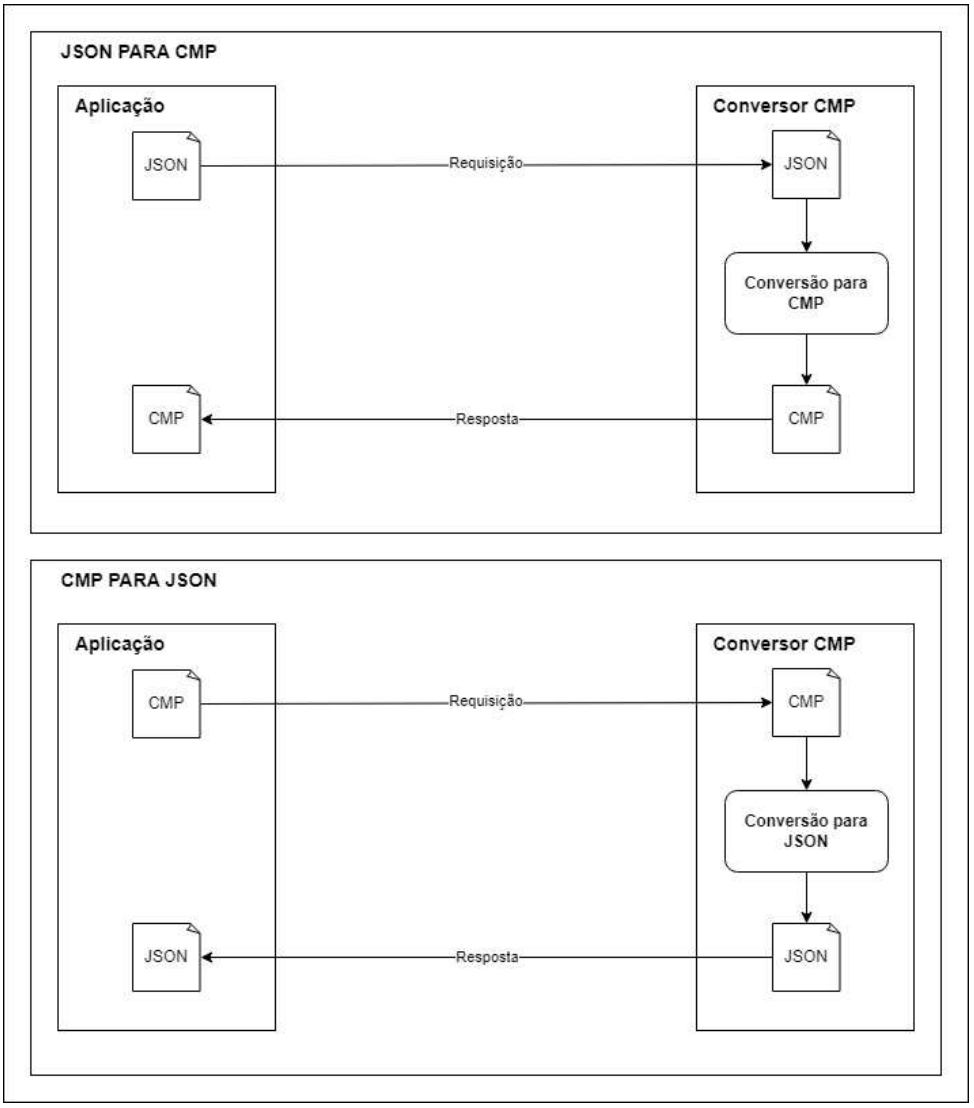
4.2. PROTÓTIPO INICIAL

Com as tecnologias definidas, um protótipo inicial da API foi desenvolvido. Este protótipo definia arquitetura do código, os *endpoints* de conversão de mensagens e a conversão do header e extra certs da PKIMessage (Figura 6), que são estruturas comuns entre as mensagens do protocolo.

Para realizar a conversão das estruturas para JSON, o sistema recebe, *no endpoint de conversão*, uma mensagem CMP em binário e cria instâncias das estruturas contidas por meio das ferramentas da biblioteca *Bouncy Castle*. Então, estas instâncias são utilizadas para construir objetos de classes java correspondentes às mesmas. Por fim, estes objetos são serializados em estruturas JSON e retornados como resposta das solicitações.

Para a conversão de JSON para CMP, o sistema recebe, no *endpoint* de conversão, um JSON como entrada. Então, os dados recebidos são codificados, por meio das ferramentas da biblioteca *Bouncy Castle*, em estruturas ASN.1 correspondentes. Por fim, as estruturas geradas são unidas em uma estrutura do tipo PKIMessage e esta estrutura é retornada como binário. A figura 14 ilustra os processos de conversão.

Figura 14 – Conversão das mensagens



Fonte: O autor (2022).

4.3. VERSÃO FINAL

A versão final foi desenvolvida a partir do protótipo descrito na seção anterior. Nesta versão, foi adicionada a conversão da estrutura body da PKIMessage (Figura 6). Para utilizar apenas uma rota de conversão para todas as mensagens, o sistema utiliza polimorfismo da estrutura body e ferramentas do *framework* Spring Boot para fazer a injeção dinâmica de dependências, usando como base a propriedade “type” que está contida no conteúdo do JSON e na mensagem CMP.

Além da conversão, foram adicionadas rotas para adicionar proteção na mensagem, realizar a assinatura de mensagens, realizar assinatura da requisição (para fazer prova de posse) e verificar a validade da assinatura da mensagem CMP.

O sistema permite gerar mensagens dos tipos *Certificate Request* e *Revocation Request*, e é capaz de converter mensagens dos tipos *Certificate Request*, *Certificate Response*, *Revocation Request* e *Revocation Response*. Assim, viabilizando a emissão e revogação de certificados digitais via protocolo CMP.

As mensagens foram testadas manualmente, utilizando uma AC gerida por uma instância do software EJBCA, que realizou a emissão e revogação de certificados.

4.4. CONFIGURAÇÃO DA EJBCA

O *software* EJBCA *Community Edition* foi utilizado para testar o funcionamento das mensagens geradas. O software funciona como uma AC e aceita mensagens CMP.

Para finalidade desejada, uma versão efêmera de instalação simples foi disponibilizada em ambiente local. Uma entidade final identificada pelo nome comum “Entidade Final Teste” foi cadastrada e recebeu privilégios para poder solicitar emissões e revogações de certificados digitais. Um perfil de certificação foi criado para definir os campos que devem ser codificados no certificado, incluindo a extensão “subjectAlternativeNames” como especificado pelas regras da ICP-Brasil.

Por fim, o módulo de CMP foi ativado, utilizando o método “EndEntityCertificate” para verificar a autenticidade das mensagens por meio do certificado da entidade final e a assinatura da mensagem.

5. RESULTADOS

Esta seção apresenta o detalhamento dos *endpoints* disponíveis e o resultado do uso das mensagens geradas para realizar operações de emissão e revogação de certificados digitais.

5.1. ENDPOINTS

Esta seção descreve a funcionalidade e o conteúdo esperado no corpo da requisição de cada *endpoint*. A Tabela 1 lista todos os endpoints disponíveis.

Tabela 1 – *Endpoints* disponíveis

Método HTTP	***Endpoint***
POST	/converter
POST	/pki-message/sign
POST	/pki-message/attach-signature
POST	/pki-message/cert-request
POST	/cert-request/sign

Fonte: O autor (2023).

5.1.1. POST /converter

Este é o *endpoint* responsável por converter estruturas JSON em mensagens CMP e vice-versa. O que define o tipo de conteúdo a ser gerado é o *header* HTTP “Content-Type”. A rota aceita, como valor do header HTTP, os valores “application/json” e “application/pkixcmp”.

Para gerar uma mensagem a partir de um JSON, será utilizado “application/json” como valor do *header* “Content-Type”. O JSON contido no corpo da requisição é análogo aos tipos definidos na RFC 4210 (ADAMS et al., 2005). Ele deve possuir a estrutura descrita na tabela 2.

Tabela 2 – Estrutura JSON /convert

Nome do campo	Tipo	Descrição
type	String	Tipo da mensagem que será gerada. Ex: “cr”, “rr”
header	Object	Conteúdo para ser codificado no header da mensagem.
body	Object	Conteúdo para ser codificado no body da mensagem
extra_certs	Array<Object>	Conteúdo para ser codificado no extraCerts da mensagem.

Fonte: O autor (2023).

O campo “header” deve possuir a estrutura descrita na tabela 3.

Tabela 3 – Estrutura JSON header

Nome do campo	Tipo	Opcional
sender	Object	não
recipient	Object	não
message_time	String	sim
protection_alg	String	sim
sender_kid	String	sim
recipient_kid	String	sim
transaction_id	String	sim
sender_nonce	String	sim
recip_nonce	String	sim

Fonte: O autor (2023).

O campo “message_time” deve conter uma data no formato “dd/MM/yyyy HH:mm:ss”.

Os campos “sender” e “recipient” devem possuir a estrutura descrita na tabela 4.

Tabela 4 – Estrutura JSON sender e recipient

Nome do campo	Tipo	Opcional
organization	String	sim
locality	String	sim
country	String	sim
common_name	String	não
organizational_units	Array	sim
state_or_province	String	sim

Fonte: O autor (2022).

Os objetos contidos no campo “extra_certs” devem possuir a estrutura descrita na tabela 5.

Tabela 5 – Estrutura JSON extra_certs

Nome do campo	Tipo	Opcional
type	String	sim
content	String	sim

O autor (2023)

O campo “type” pode ser utilizado para informar a quem pertence o certificado para o conversor. Assim, caso presente, os dados do “sender” e “recipient” contidos no header serão codificados com base no conteúdo dos certificados. Os valores possíveis são “issuer_cert” e “recipient_cert”.

As estruturas dos possíveis conteúdos do campo “body” estão descritos no apêndice A.

Em caso de sucesso, o retorno possui *status code* 200 e um binário no corpo da resposta. Este binário é a mensagem CMP.

Para gerar um JSON a partir de uma mensagem CMP, será utilizado “application/pkixcmp” como valor do *header* “Content-Type”. O conteúdo da

requisição deve ser uma mensagem CMP em binário. O retorno é um JSON que segue a mesma estrutura definida para criação das mensagens.
Em caso de erro, o retorno será um *status code* entre 400 e 500. A estrutura do erro está descrita no apêndice A.

5.1.2. POST /pki-message/sign

Este endpoint gera uma assinatura da mensagem CMP no padrão SHA256 com RSA. O corpo da requisição deve ser um JSON com a estrutura descrita na tabela 6.

Tabela 6 – Estrutura JSON /pki-message/sign

Nome do campo	Tipo	Opcional
pki_message	String	não
priv_key	String	não

O autor (2023)

O campo “pki_message” deve conter o conteúdo da mensagem CMP codificado em Base64.

O campo “priv_key” deve conter a chave privada do remetente da mensagem, descriptografada e codificada em Base64.

Em caso de sucesso, o retorno possui *status code* 200 e como conteúdo um JSON contendo a estrutura descrita na tabela 7.

Tabela 7 – Estrutura JSON retorno /pki-message/sign

Nome do campo	Tipo
pki_message_signature	String

O autor (2023)

O campo “pki_message_signature” possui a assinatura codificada em Base64.

Em caso de erro, o retorno será um *status code* entre 400 e 500. A estrutura do erro está descrita no apêndice A.

5.1.3. POST /pki-message/attach-signature

Este *endpoint* adiciona uma assinatura a uma mensagem CMP. Tornando a mensagem autenticavel. O corpo da requisição deve ser um JSON com a estrutura descrita na tabela 8.

Tabela 8 – Estrutura JSON /pki-message/attatch-signature

Nome do campo	Tipo	Opcional
pki_message	String	não
signature	String	não

O autor (2023)

O campo “pki_message” deve conter o conteúdo da mensagem CMP codificado em Base64.

O campo “signature” deve conter uma assinatura codificada em Base64.

A assinatura pode ter sido realizada pelo próprio sistema ou por alguma ferramenta externa.

Em caso de sucesso, o retorno possui *status code* 200 e como conteúdo um binário contendo a mensagem assinada.

Em caso de erro, o retorno será um *status code* entre 400 e 500. A estrutura do erro está descrita no apêndice A.

5.1.4. POST /pki-message/cert-request

Este endpoint extrai o conteúdo do campo certReq contido no body de uma CertReqMessage e retorna o mesmo. Esta funcionalidade é necessária pois é preciso assinar este conteúdo para realizar a prova de posse. O corpo da requisição deve ser um JSON com a estrutura descrita na tabela 9.

Tabela 9 – Estrutura JSON retorno /pki-message/attatch-signature

Nome do campo	Tipo	Opcional
pki_message	String	não

O autor (2023)

O campo “pki_message” deve conter o conteúdo de uma mensagem CMP de tipo “cr”, codificado em Base64.

Em caso de sucesso, o retorno possui *status code* 200 e como conteúdo um binário contendo a certReq.

Em caso de erro, o retorno será um *status code* entre 400 e 500. A estrutura do erro está descrita no apêndice A.

5.1.5. POST /cert-request/sign

Este endpoint gera uma assinatura do conteúdo do certReq no padrão SHA256 com RSA. O corpo da requisição deve ser um JSON com a estrutura descrita na tabela 10.

Tabela 10 – Estrutura JSON /cert-request/sign

Nome do campo	Tipo	Opcional
cert_request	String	não
priv_key	String	não

O autor (2023)

O campo “cert_request” deve conter o conteúdo da mensagem CMP codificado em Base64.

O campo “priv_key” deve conter a chave privada correspondente à chave pública do certificado solicitado, descriptografada e codificada em Base64.

Em caso de sucesso, o retorno possui *status code* 200 e como conteúdo um JSON contendo a estrutura descrita na tabela 11.

Tabela 11 – Estrutura JSON retorno /cert-request/sign

Nome do campo	Tipo
cert_request_signature	String

O autor (2023)

O campo “cert_request_signature” possui a assinatura codificada em Base64.

Em caso de erro, o retorno será um *status code* entre 400 e 500. A estrutura do erro está descrita no apêndice A.

5.2. EMISSÃO DE CERTIFICADO

Para realizar a emissão do certificado, um par de chaves rsa de 2048 bits foi gerado por linha de comando. Em seguida, uma mensagem foi gerada pelo conversor, utilizando o conteúdo JSON exibido na figura 15.

Figura 15 – JSON para gerar Certification Request

```
{
    "type": "cr",
    "header": {
        "sender": {
            "common_name": "Entidade Final Teste"
        },
        "recipient": {
            "common_name": "ManagementCA"
        },
        "message_time": "01/01/2023 03:00:00",
        "protection_alg": "1.2.840.113549.1.1.11"
    },
    "body": {
        "cert_req_messages": [
            {
                "cert_req": {
                    "cert_template": {
                        "subject": {
                            "common_name": "Henrique Gomes",
                            "organization": "UFSC",
                            "organizational_units": [
                                "ICP-Brasil",
                                "OU Teste"
                            ],
                            "locality": "Florianopolis",
                            "state_or_province": "SC",
                            "country": "BR"
                        },
                        "public_key": "<Base64>",
                        "extensions": {
                            "subject_alt_names": [
                                {
                                    "oid": "2.16.76.1.3.1",
                                    "value":
"2906202100000000191000000000000000000000000",
                                    "general_name": "otherName"
```

```
                                },
                                {
                                    "oid": "2.16.76.1.3.5",
                                    "value": "000000000000",
                                    "general_name": "otherName"
                                },
                                {
                                    "oid": "2.16.76.1.3.6",
                                    "value": "000000000000",
                                    "general_name": "otherName"
                                },
                                {
                                    "value": "henrique@mail.com",
                                    "general_name": "rfc822name"
                                }
                            ]
                        }
                    }
                }
            ]
    },
    "extra_certs": [
        {
            "type": "issuer_cert",
            "content": "<Base64>"
        },
        {
            "type": "recipient_cert",
            "content": "<Base64>"
        }
    ]
}
```

O autor (2023)

Devido ao tamanho do texto, os campos que contém dados codificados em Base64 foram substituídos na imagem por “<Base64>”.

O campo “public_key” possui a chave pública gerada anteriormente, que será codificada no certificado emitido. “issuer_cert” e “recipient_cert” possuem respectivamente, o certificado do remetente e do destinatário.

O conteúdo retornado foi enviado para o *endpoint* “/pki-message/cert-request” para obter o conteúdo do certReq, necessário para realizar a prova de posse da chave.

A assinatura do certReq foi realizada, utilizando a chave privada par da chave pública que estará contida no certificado. A mensagem cmp foi gerada novamente, agora com a assinatura e tipo da prova de posse adicionados no campo “popo” da estrutura “certReq”, como mostra o JSON exibido na figura 16.

Figura 16 – JSON para gerar Certification Request com prova de posse

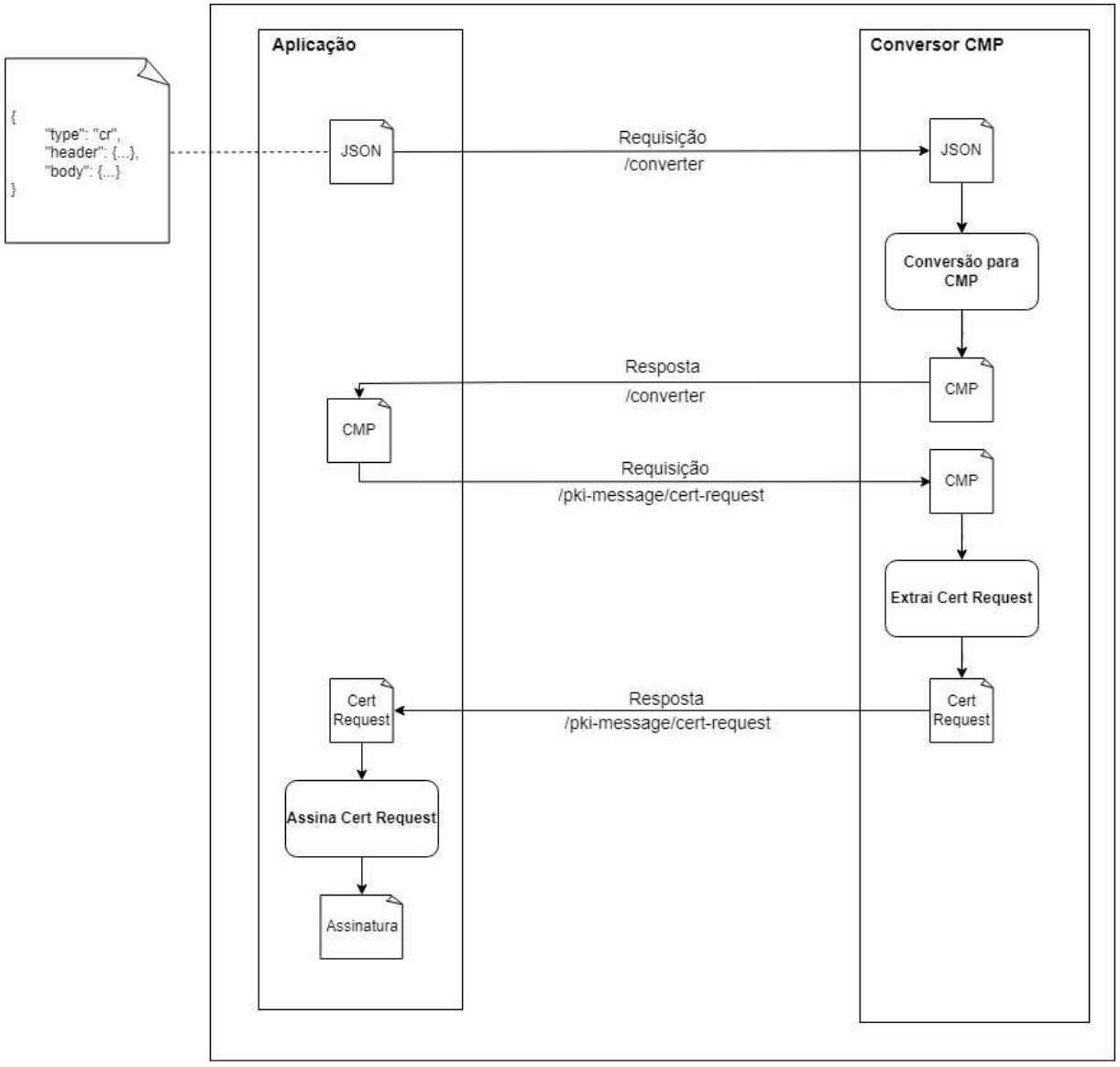
```
{
    "type": "cr",
    "header": {
        "sender": {
            "common_name": "Entidade Final Teste"
        },
        "recipient": {
            "common_name": "ManagementCA"
        },
        "message_time": "01/01/2023 03:00:00",
        "protection_alg": "1.2.840.113549.1.1.11"
    },
    "body": {
        "cert_req_messages": [
            {
                "cert_req": {
                    "cert_template": {
                        "subject": {
                            "common_name": "Henrique Gomes",
                            "organization": "UFSC",
                            "organizational_units": [
                                "ICP-Brasil",
                                "OU Teste"
                            ],
                            "locality": "Florianopolis",
                            "state_or_province": "SC",
                            "country": "BR"
                        },
                        "public_key": "<Base64>",
                        "extensions": {
                            "subject_alt_names": [
                                {
```

```
                                    "oid": "2.16.76.1.3.1",
                                    "value":
"2906202100000000191000000000000000000000000000",
                                    "general_name": "otherName"
                                },
                                {
                                    "oid": "2.16.76.1.3.5",
                                    "value": "000000000000",
                                    "general_name": "otherName"
                                },
                                {
                                    "oid": "2.16.76.1.3.6",
                                    "value": "000000000000",
                                    "general_name": "otherName"
                                },
                                {
                                    "value": "henrique@mail.com",
                                    "general_name": "rfc822name"
                                }
                            ]
                        }
                    }
                },
                "popo": {
                    "type": "popo_signing_key",
                    "value": "<Base64>"
                }
            }
        ]
    },
    "extra_certs": [
        {
            "type": "issuer_cert",
            "content": "<Base64>"
        },
        {
            "type": "recipient_cert",
            "content": "<Base64>"
        }
    ]
}
```


O autor (2023)

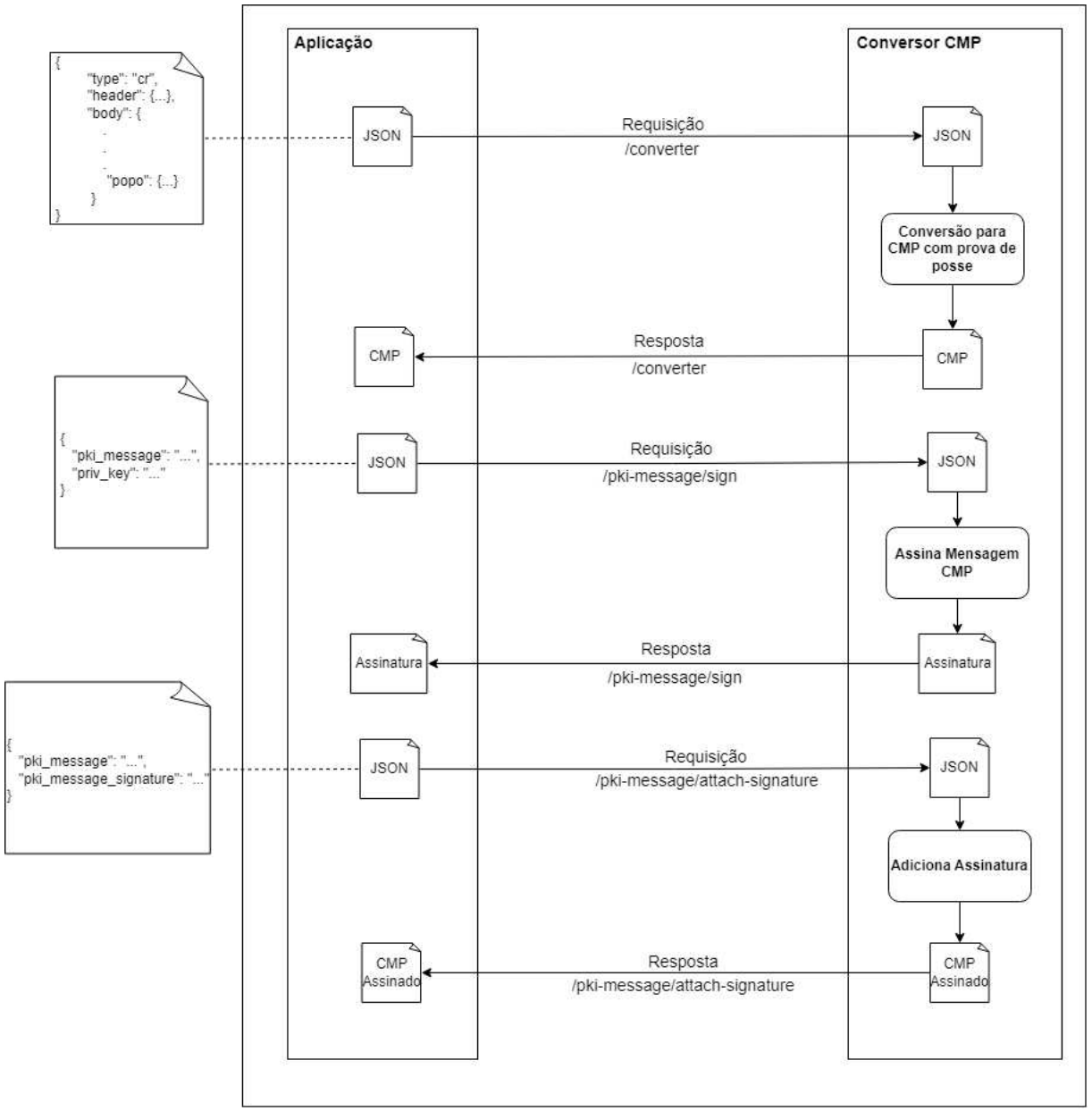
Então, o conteúdo retornado foi assinado por meio do *endpoint* “/pki-message/sign” e a assinatura foi adicionada a mensagem utilizando o *endpoint* “/pki-message/attatch-signature”, gerando a mensagem completa, assinada e com prova de posse. As figuras 17 e 18 ilustram o fluxo para obter um *Certification Request* assinado.

Figura 17 – Parte 1 do fluxo para obter *Certification Request* assinado



O autor (2023)

Figura 18 – Parte 2 do fluxo para obter *Certification Request assinado*



O autor (2023)

A mensagem foi enviada para a EJBCA, que retornou uma outra mensagem, do tipo “cp”. Esta mensagem foi enviada para o conversor, que gerou o JSON representado na figura 19.

Figura 19 – *Certification Response* convertido

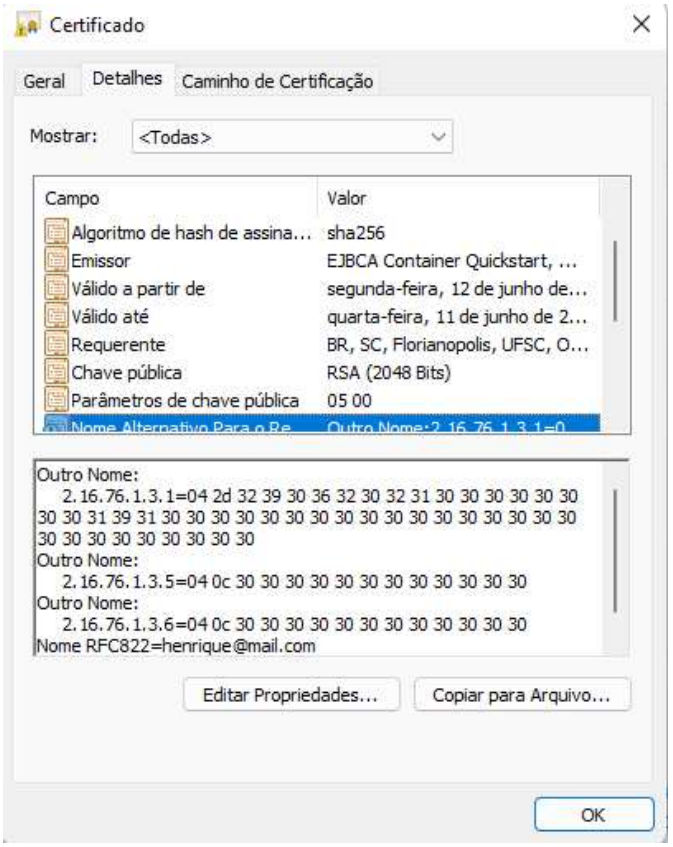

```
{
    "type": "cp",
    "header": {
        "sender": {
            "organization": "EJBCA Container Quickstart",
            "common_name": "ManagementCA"
        },
        "recipient": {
            "organization": "UFSC",
            "locality": "Florianopolis",
            "country": "BR",
            "common_name": "Henrique Gomes",
            "organizational_units": [
                "ICP-Brasil",
                "OU Teste"
            ],
            "state_or_province": "SC"
        },
        "message_time": "12/06/2023 22:59:26",
        "protection_alg": "1.2.840.113549.1.1.11",
        "sender_kid":
"5D:9D:E2:A6:8D:AE:B8:C1:CB:92:26:8B:C3:D4:EA:70:C9:04:F9:6D",
        "sender_nonce":
"91:C5:B0:1F:34:2F:8F:60:09:32:B1:43:0C:51:05:25"
    },
    "body": {
        "caPubs": [
            "<Base64>"
        ],
        "response": [
            {
                "cert_req_id": 3,
                "pki_status_info": {
                    "status": "accepted"
                },
                "certified_key_pair": {
                    "cert_or_enc_cert": {
                        "certificate": "<Base64>"
                    }
                }
            }
        ]
```

```
    },
    "extra_certs": [
        {
            "content": "<Base64>"
        }
    ]
}
```

O autor (2023)

O Certificado emitido está contido no campo “certificate”. A figura 20 mostra alguns detalhes do certificado.

Figura 20 – Visualização do certificado



Fonte: O autor (2023), adaptado do gerenciador de certificados do Windows.

5.3. REVOGAÇÃO DE CERTIFICADO

Para revogar o certificado, uma mensagem do tipo “rr” foi gerada com base no JSON representado na figura 21.

Figura 21 – JSON para gerar *Revocation Request*

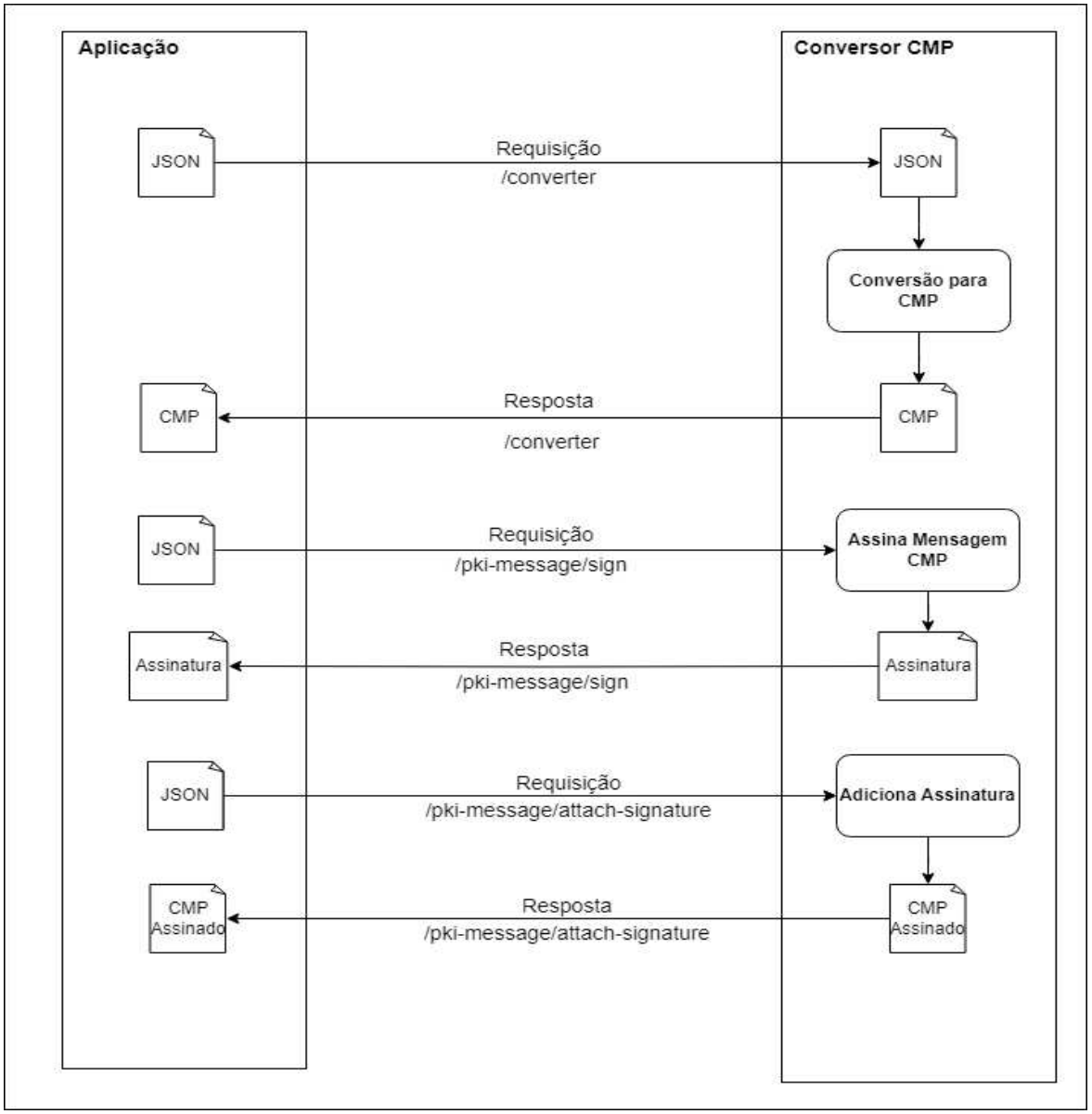
```
{
    "type": "rr",
    "header": {
        "sender": {
            "common_name": "Entidade Final Teste"
        },
        "recipient": {
            "common_name": "ManagementCA",
            "organization": "EJBCA Container Quickstart "
        },
        "protection_alg": "1.2.840.113549.1.1.11"
    },
    "body": {
        "rev_details": [
            {
                "cert_details": {
                    "serial_number":
"#3C:19:ED:90:E4:FA:CF:E0:7B:A0:B4:E5:78:6B:E8:F0:64:E8:86:1D"
                },
                "crl_entry_details": {
                    "reason_code": {
                        "value": "keyCompromise"
                    }
                }
            }
        ]
    },
    "extra_certs": [
        {
            "type": "issuer_cert",
            "content": "<Base64>"
        },
        {
            "type": "recipient_cert",
            "content": "<Base64>"
        }
    ]
}
```

O autor (2023)

A mensagem possui o serial number do certificado a ser revogado e o motivo da revogação, no caso, chaves comprometidas. O processo para assinatura da mensagem foi realizado conforme descrito na seção anterior.

A Figura 22 ilustra a troca de mensagens com o conversor para geração da mensagem.

Figura 22 – Troca de mensagens para obter *Revocation Request*



O autor (2023)

A mensagem foi enviada para a EJBCA, que retornou uma outra mensagem, do tipo “rp”. Esta mensagem foi enviada para o conversor, que gerou o JSON representado na figura 23.

Figura 23 – *Revocation Response* convertido


```
{
    "type": "rp",
    "header": {
        "sender": {
            "organization": "EJBCA Container Quickstart",
            "common_name": "ManagementCA"
        },
        "recipient": {
            "common_name": "Entidade Final Teste"
        },
        "message_time": "13/06/2023 00:06:21",
        "protection_alg": "1.2.840.113549.1.1.11",
    "sender_kid":
"5D:9D:E2:A6:8D:AE:B8:C1:CB:92:26:8B:C3:D4:EA:70:C9:04:F9:6D",
        "recip_kid": null,
        "transaction_id": null,
    "sender_nonce": "90:55:9D:FD:3F:34:C4:51:66:CB:F4:7F:04:2E:3C:53",
        "recip_nonce": null,
        "general_info": null
    },
    "body": {
        "status": [
            {
                "status": "accepted"
            }
        ],
        "rev_certs": []
    },
    "extra_certs": [
        {
            "content": "<Base64>"
        }
    ]
}
```


O autor (2023)

A EJBCA retornou uma mensagem informando que o pedido de revogação foi aceito. A figura 24 mostra a condição do certificado no sistema da EJBCA.

Figura 24 – Dados do certificado na EJBCA

Username	**Henrique Gomes**
Certificate number	1 of 1
Certificate Type/Version	X.509 v.3
Certificate Serial Number	3C19ED90E4FACFE07BA0B4E5786BE8F064E8861D
Issuer DN	UID=c-0s32rcbf63u7z2wel,CN=ManagementCA,O=EJBCA Container Quickstart
Valid from	2023-06-12 22:49:26+00:00
Valid to	2025-06-11 22:49:25+00:00

Subject DN	**CN=Henrique Gomes,OU=ICP-Brasil,OU=OU Teste,O=UFSC,L=Florianopolis,ST=SC,C=BR**
Subject Alternative Name	rfc822name=henrique@mail.com
Subject Directory Attributes	None
Public key	RSA (2048 bits): 98B426267DCFCB88918FAD2EAF8C008B18C0CF8072D4D52...

Basic constraints	**End Entity**
Key usage	Digital Signature,Non-repudiation,Key encipherment
Extended key usage	Client Authentication
Name constraints	No
Authority Information Access	No
Qualified Certificates Statements	No
Certificate Transparency SCTs	No
Signature Algorithm	SHA256WITHRSA
Fingerprint SHA-256	DABADECD7E11FC49CF2E298C1D445785 0677C7E3B2259574190E5E08B02A724D
Fingerprint SHA-1	CBB3178EC513AD454967311386FF13F062DFE430
Revoked	Yes Revocation date : 2023-06-13 00:06:21+00:00 Revocation reasons : Key compromise
Republish	

Fonte: O autor (2023), adaptado do software EJBCA.

6. CONCLUSÕES E TRABALHOS FUTUROS

O trabalho realizou a implementação de uma API capaz de converter uma mensagem CMP para uma estrutura JSON e de criar mensagens CMP através de estruturas JSON. A funcionalidade e validação das mensagens foram testadas através de operações de gerenciamento de certificados digitais realizadas no *software* EJBCA.

Os objetivos do trabalho foram atingidos, uma vez que, com o sistema desenvolvido, é possível converter mensagens e utilizar as mesmas para realizar operações de emissão e revogação de certificados.

A API desenvolvida pode ser incorporada como um serviço interno de uma organização e ser utilizada por aplicações que se comunicam com entidades de ICPs de forma fácil, inclusive, suportando estruturas exigidas pela ICP-Brasil, como os campos definidos para a extensão *subject alternative names*. A utilização da API elimina a necessidade de trabalhar com estruturas ASN.1, as quais normalmente não possuem suporte nativo em linguagens modernas, tornando o conversor responsável por processar as mensagens. Além disso, pode ser utilizado nos trabalhos relacionados apresentados nas seções 3.1 e 3.3, removendo a necessidade de tratar as mensagens CMP dentro dos sistemas apresentados e lidando apenas com estruturas JSON.

O sistema desenvolvido possui pontos que podem ser aprimorados em trabalhos futuros. Os principais pontos são: Suporte a outros algoritmos de proteção, outras opções de prova de posse e a geração de mensagens do tipo “cp” e "rp" e das demais mensagens necessárias para realizar outras operações disponíveis, como a renovação de certificados.

Um outro ponto que pode ser trabalhado é o desenvolvimento de uma documentação da API e um guia para desenvolvedores, para auxiliar no desenvolvimento de novas funcionalidades.

O sistema possui código aberto e pode ser estendido por outros desenvolvedores. O código-fonte pode ser obtido pelo *link* contido no apêndice B.

REFERÊNCIAS

ADAMS, C. et al. **Internet X.509 Public Key Infrastructure Certificate Management Protocols**. IETF, set. 2005. RFC 4210 (Proposed Standard). (Request for Comments, 4210). <http://www.ietf.org/rfc/rfc4210.txt>.

BELLARE, M.; ROGAWAEY, P. **Introduction to modern cryptography**. Davis, United States of America: University of California at Davis, 2005. Disponível em: https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf. Acesso em: 03 jul. 2023.

CHOKHANI, S.; FORD, W. Internet X.509 **Public key infrastructure certificate policy and certification practices framework**. 1999. RFC 2527.

DIFFIE, W.; HELLMAN, M. E. **New Directions in Cryptography**. IEEE Transactions on Information Theory, IT-22, n. 6, p. 644–654, 1976.

FOROUZAN, Behrouz A. **Introduction to Cryptography and Network Security**. Nova Iorque: McGraw-Hill, 2008.

HOUSLEY, Russ; POLK, Tim. **Planning for PKI**: Best practices guide for deploying public key infrastructure. Nova Iorque: Wiley, 2001.

IBM. **Digital signatures**. 30 de Junho de 2022. Disponível em: https://www.ibm.com/docs/en/ztpf/2022?topic=concepts-digital-signatures. Acesso em: 23 nov. 2022.

IBM Brasil. [s.d.]. **O que é criptografia? Definição de criptografia de dados**. Disponível em: https://www.ibm.com/br-pt/topics/encryption#:~:text=A%20criptografia%20%C3%A9%20uma%20forma,com%20uma%20chave%20de%20descriptografia.. Acesso em: 03 jul. 2023.

Infor Channel. **Mercado de certificação digital tem crescimento recorde no País**. 2021. Disponível em: <https://inforchannel.com.br/2021/07/28/mercado-de-certificado-digital-tem-crescimento-recorde-no-pais/>. Acesso em: 15 jul. 2022.

ITU-T. **Itu-t recommendation x.509 — iso/iec 9594-8: ”information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks”**. 2008. Relatório técnico.

ITU-T. [s.d.]. **Introduction to ASN.1**. Disponível em: https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx. Acesso em: 03 jul. 2023.

ITI. **Benefícios**. 2020a. Disponível em: <https://www.gov.br/iti/pt-br/assuntos/certificado-digital/beneficios>. Acesso em: 21 nov. 2022.

ITI. **Entes da ICP-Brasil**. 2020b. Disponível em: <https://www.gov.br/iti/pt-br/assuntos/icp-brasil/entes-da-icp-brasil>. Acesso em: 07 dez. 2022.

ITI. **ITI em Números**. 2022b. Disponível em: <https://numeros.iti.gov.br>. Acesso em: 15 jul. 2022.

ITI. **ICP-Brasil**. 2017. Disponível em: <https://www.gov.br/iti/pt-br/assuntos/icp-brasil>. Acesso em: 07 dez. 2022.

ITI. **Obter Certificado Digital**. 2022a. Disponível em: <https://www.gov.br/pt-br/servicos/obter-certificacao-digital>. Acesso em: 21 nov. 2022.

KAUSE, T; PEYLO, M. **Internet X.509 Public Key Infrastructure -- HTTP Transfer for the Certificate Management Protocol (CMP)**. IETF, set. 2012. RFC 6712 (Proposed Standard). (Request for Comments, 6712). <http://www.ietf.org/rfc/rfc6712.txt>

KARATSIOLIS, Vangelis; LIPPERT, Marcus; WIESMAIER, Alexander. **Using LDAP Directories for Management of PKI Processes**. European Public Key Infrastructure Workshop**.** Springer, Berlin, Heidelberg. p. 126-134, 2004.

LEE, Yong; LEE, Jaeil; LEE, GooYeon. **Wireless certificate management protocol supporting mobile phones**. 2008 IEEE Congress on Services-Part I. IEEE. p. 353-359, 2008.

RAY, Sangram; BISWAS, G. P. **An ECC based public key infrastructure usable for mobile applications**. Proceedings of the second international conference on computational science, engineering and information technology**.** p. 562-568, 2012.

SCHAAD, J. **Internet X.509 Public Key Infrastructure Certificate Request Message Format (CRMF)**. IETF, set. 2005. RFC 4211 (Proposed Standard). (Request for Comments, 4211). <http://www.ietf.org/rfc/rfc4211.txt>.

STALLINGS, William. **Cryptography and Network Security Principles and Practice**. 6. ed. Nova Jersey: Pearson, 2013.

SILVÈRIO, Anderson Luiz. **ANÁLISE E IMPLEMENTAÇÃO DE UM PROTOCOLO DE GERENCIAMENTO DE CERTIFICADOS**. [S.l.: s.n.], 2011. Monografia (Bacharel em Ciências da Computação), UFSC (Universidade de Santa Catarina, Brasil).

The Legion of the Bouncy Castle. [s.d.]. **The Bouncy Castle Cryptography Library**. Disponível em: https://www.bouncycastle.org/. Acesso em: 03 jul. 2023.

Terra. **Panorama do mercado da certificação digital**. 2022. Disponível em: <https://www.terra.com.br/noticias/panorama-do-mercado-da-certificacao-digital,edf742ce990777d25bd6e5cca69188f7ecwtt54o.html>. Acesso em: 15 jul. 2022.

TOORANI, M.; SHIRAZI, A.A.B. **LPKI - A Lightweight Public Key Infrastructure for the Mobile Environments**. 11th IEEE International Conference on Communication Systems (IEEE ICCS'08), p.162-166, Guangzhou, China, Nov. 2008

APÊNDICE A - ESTRUTURAS SUPORTADAS COMO BODY NA CONVERSÃO JSON PARA CMP

Este apêndice descreve as estruturas suportadas no campo “body” pertencente ao JSON enviado no *endpoint* “/converter”, utilizado para gerar as mensagens CMP.

As tabelas que descrevem o conteúdo tem a seguinte estrutura:

NomeDaEstrutura	
Nome do campo	**Tipo**
valor	valor

“NomeDaEstrutura” é utilizado como identificador do objeto descrito. “Nome do campo” e “Tipo” possuem, respectivamente, o valor da chave do campo no JSON e o tipo de valor contido no mesmo.

As estruturas deste campo são mais complexas pois possuem objetos aninhados, por tanto, os valores contidos na coluna “tipo” seguem as seguintes regras:

- Campos com tipos básicos como *strings* e *numbers* terão seus próprios tipos registrados como valor da coluna.
- Campos que possuem como tipo um outro objeto terão como valor da coluna o título da tabela que descreve os campos do objeto contido.
- Caso o campo seja um array, o valor da coluna será Array<O>, onde O é um dos tipos básicos ou o título de uma tabela que descreve os campos dos objetos contidos.

Estruturas comuns em diferentes mensagens:

Tabela 12 – Estrutura JSON CertTemplate

CertTemplate	
Nome do campo	**Tipo**
version	number
issuer	X500Name
validity	Validity
subject	X500Name
extensions	Extensions<subject_alt_names>
serial_number	string
signing_alg	string
public_key	string
issuer_uid	string
subject_uid	string

Fonte: O autor (2023).

Tabela 13 – Estrutura JSON X500Name

X500Name	
Nome do campo	**Tipo**
organization	string
locality	string
country	string
common_name	string
state_or_province	string
organizational_units	Array<string>

Fonte: O autor (2023).

Tabela 14 – Estrutura JSON Validity

Validity	
Nome do campo	**Tipo**
not_before	string
not_after	string

Fonte: O autor (2023).

Tabela 15 – Estrutura JSON Extensions

Extensions	
Nome do campo	**Tipo**
subject_alt_names	Array<SubjectAltName>
reason_code	ReasonCode

Fonte: O autor (2023).

Estrutura do campo “body” para geração de mensagens do tipo “cr”:

Tabela 16 – Estrutura JSON Body cr

Body	
Nome do campo	**Tipo**
cert_req_messages	Array<CertReqMsg>

Fonte: O autor (2023).

Tabela 17 – Estrutura JSON CertReqMsg

CertReqMsg	
Nome do campo	**Tipo**
cert_req	CertReq
popo	ProofOfPossession
reg_info	RegInfo

Fonte: O autor (2023).

Tabela 18 – Estrutura JSON CertReq

CertReq	
Nome do campo	**Tipo**
controls	Controls
cert_req_id	number
cert_template	CertTemplate

Fonte: O autor (2023).

Tabela 19 – Estrutura JSON Controls

Controls	
Nome do campo	**Tipo**
authenticator_control	AuthenticatorControl
reg_token_control	RegTokenControl

Fonte: O autor (2023).

Tabela 20 – Estrutura JSON AuthenticatorControl

AuthenticatorControl	
Nome do campo	**Tipo**
value	string

Fonte: O autor (2023).

Tabela 21 – Estrutura JSON RegTokenControl

RegTokenControl	
Nome do campo	**Tipo**
value	string

Fonte: O autor (2023).

Tabela 22 – Estrutura JSON SubjectAltName

SubjectAltName	
Nome do campo	**Tipo**
oid	string
value	string
general_name	string

Fonte: O autor (2023).

Tabela 23 – Estrutura JSON ProofOfPossession

ProofOfPossession	
Nome do campo	**Tipo**
type	string
value	string

Fonte: O autor (2023).

Tabela 24 – Estrutura JSON RegInfo

RegInfo	
Nome do campo	**Tipo**
utf8_pairs	Array<Utf8Pair>
cert_req	CertReq

Fonte: O autor (2023).

Tabela 25 – Estrutura JSON Utf8Pair

Utf8Pair	
Nome do campo	**Tipo**
value	string

Fonte: O autor (2023).

Estrutura do campo “body” para geração de mensagens do tipo “rr”:

Tabela 26 – Estrutura JSON Body rr

Body	
Nome do campo	**Tipo**
rev_details	Array<RevDetail>

Fonte: O autor (2023).

Tabela 27 – Estrutura JSON RevDetail

RevDetail	
Nome do campo	**Tipo**
cert_details	CertTemplate
crl_entry_details	Extensions<reason_code>

Fonte: O autor (2023).

Tabela 28 – Estrutura JSON ReasonCode

ReasonCode	
Nome do campo	**Tipo**
value	string

Fonte: O autor (2023).

APÊNDICE B - CÓDIGO FONTE DO PROJETO

O código fonte deste projeto pode ser obtido em:
https://github.com/henriquemgomes/cmp-converter

APÊNDICE C - ARTIGO SBC

Desenvolvimento de um Conversor de Mensagens do Protocolo CMP à Luz da ICP-Brasil

Henrique Meireles Costa Gomes[1], Carla Merkle Westphall[1],

[1] Departamento de Informática e Estatística
Universidade Federal de Santa Catarina (UFSC)
88.040-900 – Florianopolis – SC

henrique.gomes@grad.ufsc.br, carla.merkle.westphall@ufsc.br

Abstract. *The Certificate Management Protocol (CMP) is a messaging protocol designed for certificate management operations in a public key infrastructure. Its messages consist of a signed ASN.1 structure that contains data about the sender and recipient, as well as relevant information for the operation at hand. Implementing this protocol in modern applications poses a challenge. Systems developed in languages like JavaScript, PHP, and Python often need to use modules or services in other languages to encode or decode the messages. This work aims to facilitate the creation and consumption of CMP messages within the requirements of ICP-Brasil by developing a web application programming interface (API). The API is capable of converting the content of a CMP message to a JSON-formatted structure and vice versa. This conversion is beneficial since the JSON structure is simpler and natively supported in most modern programming languages.*

Resumo. *Certificate Management Protocol (CMP) é um protocolo de mensagens para realizar operações de gerenciamento de certificados em uma infraestrutura de chaves publicas, suas mensagens consistem basicamente em: Uma estrutura ASN.1 assinada, que contém dados sobre o remetente e o destinatário, além de informações relevantes para a operação em questão. A implementação deste protocolo em aplicações modernas é um desafio. Sistemas desenvolvidos em linguagens como JavaScript, PHP e Python precisam utilizar módulos ou serviços em outras linguagens, a fim de codificar ou decodificar as mensagens. Este trabalho tem como objetivo facilitar a criação e o consumo de mensagens CMP, dentro das exigências da ICP-Brasil, por meio do desenvolvimento de uma interface de programação de aplicação (API) web, capaz de converter o conteúdo de uma mensagem CMP para uma estrutura no formato JSON e vice-versa, uma vez que esta estrutura é mais simples e possui suporte nativo na maioria das linguagens de programação modernas.*

1. Introdução

O certificado digital é um documento eletrônico que possibilita a identificação de pessoas físicas e jurídicas, a troca segura de informações e a integridade e confidencialidade de mensagens. Além disso, com um certificado dentro dos padrões da Infraestrutura de Chaves Públicas Brasileira, podemos realizar assinaturas digitais com validade jurídica e acessar serviços digitais, como os fornecidos pelo governo federal, de forma prática e segura [ITI 2022].

O mercado de certificação digital brasileiro está em crescimento. Por isso, cada vez mais empresas surgem ou se adaptam para atuar neste mercado, seja para operar diretamente na emissão e gestão de certificados, ou para prover soluções para entidades que já atuam na área.

O processo para operar como Autoridade Certificadora ou Autoridade de Registro, pertencente à cadeia da ICP-Brasil, é rigoroso. Existe um processo extenso e burocrático, onde é necessário ser aprovado por auditorias que verificam o cumprimento das exigências definidas pelo ITI.

A comunicação entre Autoridades Certificadoras e Autoridades de Registro é um dos pontos auditados, a troca de informações deve ser feita de forma segura. Uma forma de garantir a segurança na comunicação é utilizar o protocolo CMP (Certificate Management Protocol), que define um padrão de mensagens para criação e gestão de certificados. Estas mensagens são assinadas e possuem informações sobre remetente e destinatário, além de dados referentes à transação.

A proposta deste trabalho consiste em implementar uma API capaz de converter estruturas JSON em mensagens CMP e vice-versa, ou seja, criar mensagens CMP a partir de entradas no formato JSON e transformar entradas no formato CMP em estruturas JSON, possibilitando a fácil utilização do protocolo em aplicações modernas baseadas em linguagens que não possuem suporte nativo e bibliotecas que consigam interpretar o protocolo.

2. Objetivos

2.1. Geral

O objetivo geral deste trabalho é desenvolver uma API web para conversão de mensagens CMP. A aplicação deverá ser capaz de receber requisições HTTP e, baseado no conteúdo recebido, retornar uma estrutura JSON com os dados da mensagem CMP recebida ou criar uma nova mensagem CMP a partir dos dados contidos em uma estrutura JSON recebida. As mensagens CMP geradas deverão suportar algumas estruturas estabelecidas pelo ITI, para atender exigências da ICP-Brasil.

Além disso, para demonstração, as mensagens CMP geradas pelo sistema devem ser interpretadas pelo software de Autoridade Certificadora “EJBCA Community Edition”, desenvolvido pela empresa PrimeKey, realizando operações de emissão e revogação de certificados digitais.

2.2. Específicos

- Explicar o protocolo CMP e a estrutura das mensagens, como especificado no documento RFC 4210 [Adams et al. 2005], dentro do domínio do trabalho.
- Conversão do conteúdo de mensagens CMP para estruturas JSON.
- Criação de mensagens CMP a partir de estruturas JSON.
- Utilização das mensagens geradas para realizar operações de emissão e gerenciamento de certificados em um software de Autoridade Certificadora.

3. Conceitos Básicos

3.1. Criptografia

Historicamente, a criptografia surgiu como um meio de permitir que as partes envolvidas mantenham a privacidade das informações que enviam umas às outras, mesmo na presença de um sujeito malicioso com acesso ao canal de comunicação. Embora fornecer privacidade continue sendo um objetivo central, o campo se expandiu para englobar outros, como garantir a integridade e autenticidade das comunicações [Bellare et al. 2005]. Os dois principais tipos de criptografia são assimétrica e simétrica [IBM Brasil s.d.].

3.1.1. Criptografia Simétrica

A criptografia simétrica utiliza uma única chave (conjunto de valores numéricos utilizados por algoritmos de criptografía) para criptografar e descriptografar dados. Esse sistema necessita de um canal seguro entre as partes envolvidas para transmitir a chave secreta compartilhada, utilizada na operação. As mensagens criptografadas podem ser transmitidas por um canal sem proteção, pois um terceiro malicioso escutando o canal não possui a chave necessária para descriptografar a mensagem e acessar o conteúdo original [Forouzan 2008].

Neste sistema, os indivíduos A e B recebem a chave por meio de um canal seguro. O indivíduo A cifra a mensagem utilizando a chave e envia a mensagem cifrada por um outro canal. O indivíduo B recebe a mensagem cifrada e decifra essa mensagem utilizando a chave, obtendo o conteúdo original cifrado pelo indivíduo A, como mostra a figura 1.

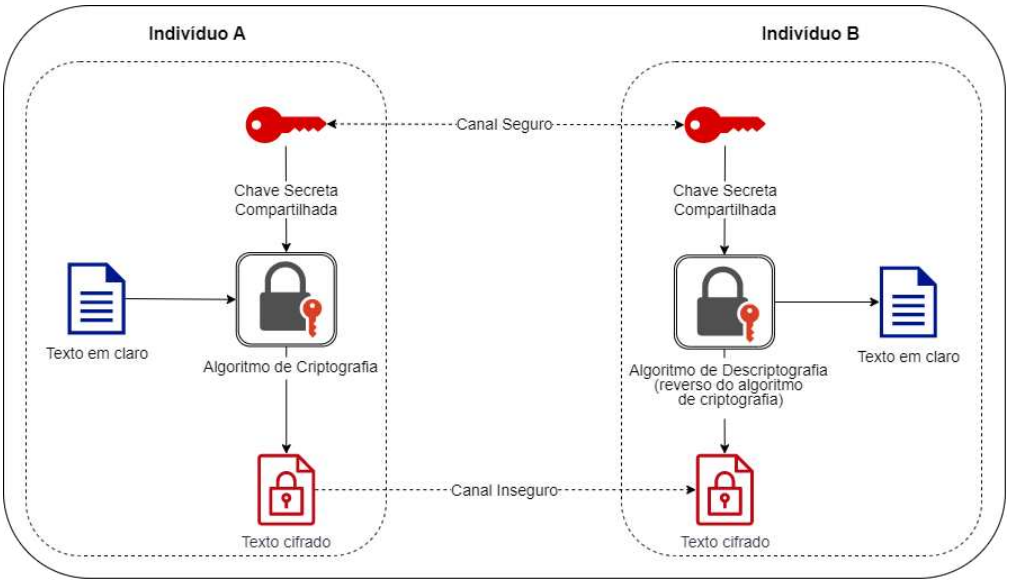


Figura 1. Criptografia Simétrica

3.1.2. Criptografia Assimétrica

Proposto por Whitfield Diffie e Martin E. Hellman no artigo New Directions in Cryptography [Diffie and Hellman 1976], se trata de um sistema de criptografia que supera alguns limites existentes na criptografia simétrica, na qual é necessário estabelecer um canal seguro para troca de chaves, o que limita a troca de mensagens criptografadas para a comunicação apenas entre partes que se prepararam previamente para realizá-la.

A criptografia assimétrica, também conhecida como criptografia de chave pública, é baseada em pares de chaves. Um par de chaves é composto por uma chave pública e uma chave privada, sendo a chave privada secreta, ou seja, conhecida apenas pelo seu dono, e a chave pública aberta, que pode ser acessada por outros.

Por meio de funções matemáticas utilizadas na geração do par de chaves, é possível garantir que não é possível descobrir a chave privada a partir da chave pública e que, uma mensagem cifrada por uma chave pública só pode ser decifrada pela chave privada correspondente e vice-versa.

Assim, caso o indivíduo A deseje enviar uma mensagem privada que pode ser lida apenas pelo indivíduo B, ele deve cifrar a mensagem com a chave pública do indivíduo B. Para ler a mensagem, o indivíduo B deve decifrá-la utilizando sua própria chave privada, como mostra a figura 2.

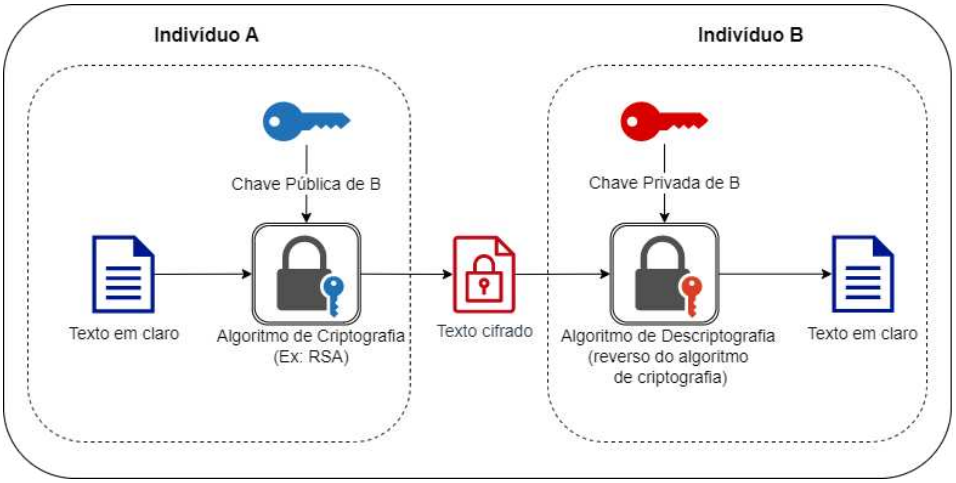


Figura 2. Criptografia Assimétrica

3.1.3. Resumo Criptográfico

Resumo criptográfico, ou hash, se trata de um conteúdo gerado por uma função hash. Este tipo de função recebe como entrada um bloco de dados de tamanho variável e produz uma saída de tamanho fixo. O objetivo do resumo criptográfico é garantir a integridade dos dados. Uma mudança em qualquer bit do conteúdo da entrada resulta, com grandes chances, em uma mudança no resumo criptográfico [Stallings 2013].

Um algoritmo de hash utilizado para criptografia é conhecido como função hash criptográfica. Algoritmos deste tipo também devem garantir que é computacionalmente

inviável recriar o conteúdo original a partir do seu hash e gerar uma colisão (produzir duas entradas que resultam no mesmo hash).

A figura 3 mostra o resultado da execução da função hash SHA256 recebendo como entrada dois textos parecidos

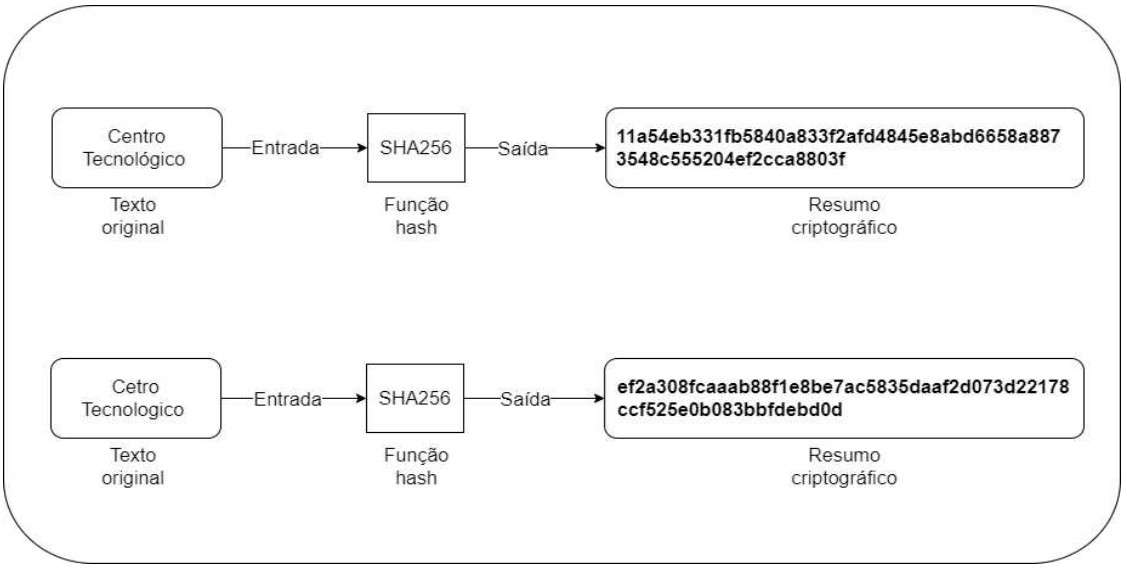


Figura 3. Exemplo SHA256

3.1.4. Assinatura Digital

A criptografia simétrica resolve problemas relacionados à interceptação de mensagens trocadas em canais inseguros, porém não resolve problemas relacionados à falsificação e adulteração do conteúdo original [IBM, 2022].

A assinatura digital é uma solução para situações como essas, onde não há plena confiança na outra parte. A assinatura digital deve possuir as seguintes propriedades: Deve checar o autor e a data e hora da assinatura, deve autenticar o conteúdo no momento da assinatura e deve ser verificável por terceiros.

Uma função segura de resumo criptográfico é uma base para satisfazer esses requisitos. A assinatura é, basicamente, o ato de usar a chave privada para cifrar o resumo criptográfico do conteúdo original, ela consiste no hash criptografado e informações como os algoritmos utilizados para criptografia e hash [IBM 2022].

A verificação da assinatura consiste, basicamente, em decifrar o resumo criptográfico com a chave pública do assinante e comparar o resumo criptográfico assinado com o resumo criptográfico do conteúdo original.

A figura 4 demonstra esses processos.

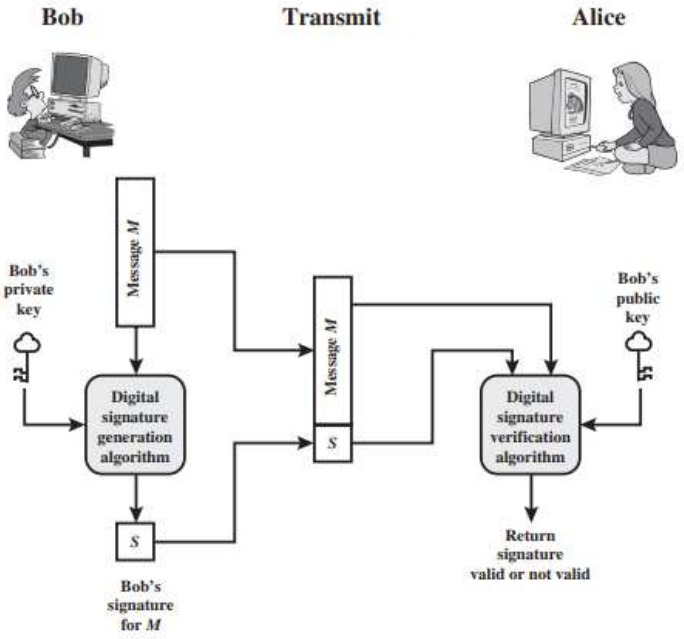


Figura 4. Modelo genérico do processo de assinatura digital (Adaptado de [STALLINGS 2013])

3.2. Certificado X.509

Também conhecido como certificado digital ou certificado de chave pública, é um arquivo eletrônico assinado por uma Autoridade Certificadora (AC) que contém, além da assinatura, uma chave pública, informações sobre a identidade do dono do par de chaves e informações sobre o emissor do certificado [Stallings 2013].

X.509 é um padrão internacional que define uma estrutura para os certificados de chave pública e listas de certificados revogados [ITU-T 2008], utilizadas para avisar a revogação dos certificados.

A Figura 5 mostra a estrutura de um certificado digital e de uma lista de revogação. A Infraestrutura de Chaves Públicas Brasileira utiliza a versão 3 do certificado X.509.

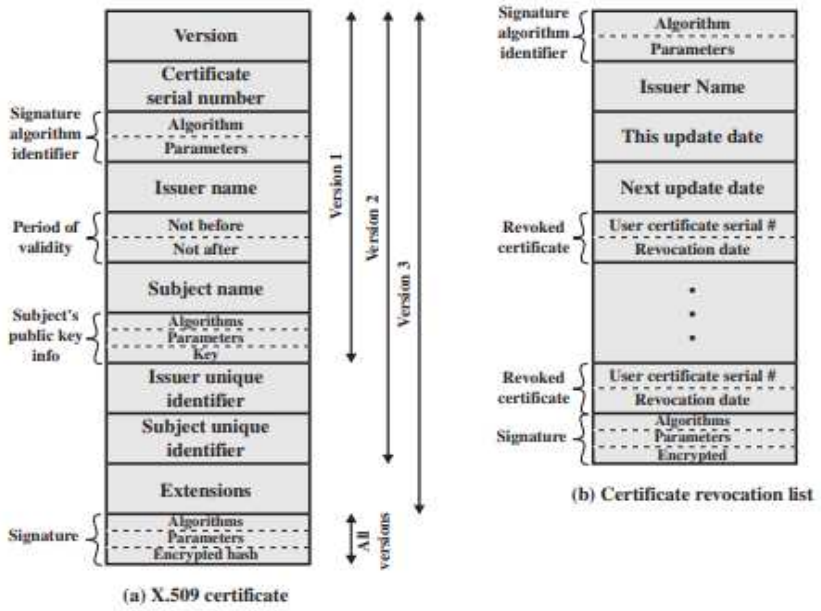


Figura 5. Estrutura Certificado X.509 e LCR (Adaptado de [STALLINGS 2013])

3.3. Infraestruturas de Chaves Públicas

Uma Infraestrutura de Chave Pública (ICP) é composta por hardwares, softwares, pessoas, políticas e procedimentos necessários para criar, gerenciar, distribuir, usar, armazenar e revogar certificados digitais [Toorani and Shirazi 2008]. Dentre as entidades que compõem uma ICP, podemos destacar Autoridades Certificadoras e Autoridades de Registro.

Estas entidades são descritas pelos autores Chokhani, Ford, Sabett, Merrill, e Wu (2003) da seguinte forma:

- Autoridade Certificadora: Responsável por emitir os certificados digitais. Ela é a entidade emissora em relação aos certificados que emite e é a entidade final em relação a AC que emitiu seu certificado. As ACs estão organizadas hierarquicamente, onde ACs de um nível maior emitem certificados para outras ACs que emitem certificados para outras Entidades Finais.
- Autoridade de Registro: Entidades que estabelecem os procedimentos para solicitação de certificado, identificam e autenticam os solicitantes, iniciam ou enviam solicitações de revogação e aprovam ou reprovam solicitações de emissão e renovação de certificados.

As entidades que solicitam e recebem certificados são conhecidas como Entidades Finais ou como Solicitantes. No contexto da ICP-Brasil os solicitantes mais comuns são pessoas físicas e pessoas jurídicas, porém também é possível emitir certificados para outros fins como por exemplo, certificados digitais para objetos metrológicos, como bombas de gasolina.

3.4. Protocolo CMP

O Certificate Management Protocol (CMP) é um protocolo para criação e gerenciamento de certificados X.509v3. Ele proporciona a interação online entre entidades da ICP, incluindo a comunicação entre uma autoridade certificadora e um sistema cliente [Adams et al. 2005].

O protocolo CMP implementa 27 tipos de mensagens utilizadas para comunicação das entidades e gestão de certificados digitais. Existem mensagens para gerar, renovar, atualizar e revogar certificados digitais.

Todas as mensagem CMP, especificadas na RFC 4210 [Adams et al. 2005], possuem a estrutura representada na figura 6:

```
PKIMessage ::= SEQUENCE {
   header           PKIHeader,
   body             PKIBody,
   protection   [0] PKIProtection OPTIONAL,
   extraCerts   [1] SEQUENCE SIZE (1..MAX) OF CMPCertificate
                    OPTIONAL
}
```

Figura 6. Estrutura PKIMessage (Adaptado de [ADAMS et al. 2005])

O header contém dados comuns entre as mensagens do protocolo. A figura 7 descreve a sua estrutura.

```
PKIHeader ::= SEQUENCE {
    pvno                INTEGER     { cmp1999(1), cmp2000(2) },
    sender              GeneralName,
    recipient           GeneralName,
    messageTime     [0] GeneralizedTime         OPTIONAL,
    protectionAlg   [1] AlgorithmIdentifier     OPTIONAL,
    senderKID       [2] KeyIdentifier           OPTIONAL,
    recipKID        [3] KeyIdentifier           OPTIONAL,
    transactionID   [4] OCTET STRING            OPTIONAL,
    senderNonce     [5] OCTET STRING            OPTIONAL,
    recipNonce      [6] OCTET STRING            OPTIONAL,
    freeText        [7] PKIFreeText             OPTIONAL,
    generalInfo     [8] SEQUENCE SIZE (1..MAX) OF
                        InfoTypeAndValue     OPTIONAL
}
```

Figura 7. Estrutura PKIHeader (Adaptado de [ADAMS et al. 2005])

As informações contidas nesta estrutura são utilizadas para indicar a chave do remetente e destinatário, identificar o algoritmo utilizado para proteger a mensagem (caso exista uma proteção), identificar a transação, prevenir ataques de repetição e enviar informações que podem ser úteis ao destinatário. A obrigatoriedade de preenchimento de alguns campos depende do contexto da mensagem e da presença ou ausência de outros campos.

O body especifica qual o tipo da mensagem. O conteúdo da mensagem varia de acordo com o tipo escolhido. A figura 8 mostra as opções existentes.

```
PKIBody ::= CHOICE {
        ir       [0]  CertReqMessages,       --Initialization Req
        ip       [1]  CertRepMessage,        --Initialization Resp
        cr       [2]  CertReqMessages,       --Certification Req
        cp       [3]  CertRepMessage,        --Certification Resp
        p10cr    [4]  CertificationRequest,  --PKCS #10 Cert.  Req.
        popdecc  [5]  POPODecKeyChallContent --pop Challenge
        popdecr  [6]  POPODecKeyRespContent, --pop Response
        kur      [7]  CertReqMessages,       --Key Update Request
        kup      [8]  CertRepMessage,        --Key Update Response
        krr      [9]  CertReqMessages,       --Key Recovery Req
        krp      [10] KeyRecRepContent,      --Key Recovery Resp
        rr       [11] RevReqContent,         --Revocation Request
        rp       [12] RevRepContent,         --Revocation Response
        ccr      [13] CertReqMessages,       --Cross-Cert.  Request
        ccp      [14] CertRepMessage,        --Cross-Cert.  Resp
        ckuann   [15] CAKeyUpdAnnContent,    --CA Key Update Ann.
        cann     [16] CertAnnContent,        --Certificate Ann.
        rann     [17] RevAnnContent,         --Revocation Ann.
        crlann   [18] CRLAnnContent,         --CRL Announcement
        pkiconf  [19] PKIConfirmContent,     --Confirmation
        nested   [20] NestedMessageContent,  --Nested Message
        genm     [21] GenMsgContent,         --General Message
        genp     [22] GenRepContent,         --General Response
        error    [23] ErrorMsgContent,       --Error Message
        certConf [24] CertConfirmContent,    --Certificate confirm
        pollReq  [25] PollReqContent,        --Polling request
        pollRep  [26] PollRepContent         --Polling response
}
```

Figura 8. Estrutura PKIBody (Adaptado de [ADAMS et al. 2005])

3.4.1. ASN.1

As mensagens do protocolo CMP são, basicamente, estruturas ASN.1. Segundo o ITU Telecommunication Standardization Sector [ITU-T s.d.], ASN.1 se trata de uma notação utilizada para descrever dados transmitidos por protocolos de telecomunicações, independentemente da implementação de linguagem e da representação física desses dados, independente da complexidade da aplicação. A notação fornece alguns tipos básicos pré-definidos, por exemplo: inteiros, booleanos, strings de bits, strings de caracteres, entre outros. Também possibilita a definição de tipos construídos, como: estruturas, listas, escolha entre tipos, entre outros.

Além disso, a notação ASN.1 oferece extensibilidade que aborda o problema e fornece suporte para a interoperabilidade entre sistemas previamente implantados e versões mais recentes, projetadas anos depois.

3.4.2. Transporte das mensagens

Como especificado no documento RFC 6712 [Kause and Peylo 2012], as mensagens do protocolo CMP são transportadas via HTTP.

A utilização deste protocolo para transportar as mensagens CMP usa exclusivamente o método POST. Uma PKIMessage codificada como DER (Distinguished Encoding Rules), uma codificação binária para estruturas utilizadas na certificação, deve ser enviada no corpo da requisição HTTP. Em caso de sucesso da solicitação, o servidor deve enviar uma mensagem de resposta CMP no corpo de uma resposta HTTP com o status code 200. Respostas HTTP para mensagens CMP de anúncio utilizam o status code 201 ou 202, para identificar se informação recebida foi

processada. Os status codes de erro 4xx e 5xx podem ser utilizados para informar erros ao cliente. Deve-se utilizar o header HTTP “Content-Type” com o valor “application/pkixcmp” ao transportar uma mensagem CMP.

Ao enviar uma mensagem CMP de erro, o servidor deve fechar a conexão, ainda que a transação não tenha sido completada. Ao utilizar o HTTP sem encapsula-lo em protocolos de segurança, como o TLS (Transport Layer Security), não há proteção de integridade. Portanto, informações do protocolo HTTP não devem ser usadas para alterar o estado da transação.

4. Desenvolvimento de Um Conversor de Mensagens CMP

Esta seção descreve as tecnologias utilizadas e o processo de desenvolvimento do sistema.

4.1. Tecnologias de Desenvolvimento

O sistema sugerido é uma API desenvolvida na linguagem Java com o uso do framework Spring Boot. A biblioteca Bouncy Castle foi utilizada para gerar as mensagens CMP. Para testar as mensagens, o software EJBCA foi utilizado para emitir e revogar certificados digitais. As subseções a seguir apresentam e justificam as tecnologias utilizadas.

4.1.1. Java

A linguagem Java foi escolhida para o desenvolvimento do trabalho devido a familiaridade do autor com a utilização da mesma e a disponibilidade da biblioteca Bouncy Castle. Além disso, existe uma grande quantidade de suporte disponível para a linguagem.

4.1.2. Bouncy Castle

De acordo com The Legion of Bouncy Castle (s.d.), a Bouncy Castle consiste em um conjunto de bibliotecas de código aberto disponível para as linguagens de programação Java e C#. A biblioteca disponibiliza diversas APIs e ferramentas para criptografia, incluindo ferramentas que possibilitam a geração e processamento das mensagens do protocolo CMP.

4.1.3. Framework Spring Boot

O Spring Boot é um framework para a linguagem Java. Ele disponibiliza ferramentas como injeção de dependências, filtro e validação de requisições, tratamento de erros e ambiente de testes. Além disso, ele cuida automaticamente de grande parte da configuração da aplicação web.

4.1.4. EJBCA

EJBCA é um software de ICP de código aberto que suporta mensagens CMP. Com ele, foi possível criar uma ICP de teste para realizar operações como emitir e revogar certificados digitais.

4.2. Protótipo Inicial

Com as tecnologias definidas, um protótipo inicial da API foi desenvolvido. Este protótipo definia arquitetura do código, os endpoints de conversão de mensagens e a conversão do header e extra certs da PKIMessage (figura 6), que são estruturas comuns entre as mensagens do protocolo.

Para realizar a conversão das estruturas para JSON, o sistema recebe, no endpoint de conversão, uma mensagem CMP em binário e cria instâncias das estruturas contidas por meio das ferramentas da biblioteca Bouncy Castle. Então, estas instâncias são utilizadas para construir objetos de classes java correspondentes às mesmas. Por fim, estes objetos são serializados em estruturas JSON e retornados como resposta das solicitações. Para a conversão de JSON para CMP, o sistema recebe, no endpoint de conversão, um JSON como entrada. Então, os dados recebidos são codificados, por meio das ferramentas da biblioteca Bouncy Castle, em estruturas ASN.1 correspondentes. Por fim, as estruturas geradas são unidas em uma estrutura do tipo PKIMessage e esta estrutura é retornada como binário. A figura 9 ilustra os processos de conversão.

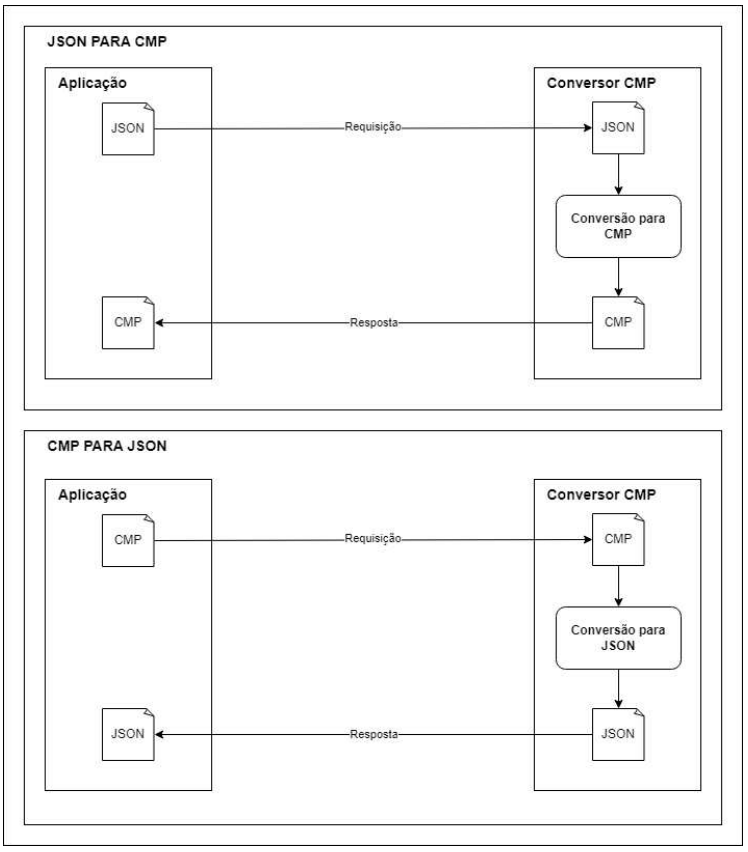


Figura 9. Conversão das mensagens

4.3. Versão Final

A versão final foi desenvolvida a partir do protótipo descrito na seção anterior. Nesta versão, foi adicionada a conversão da estrutura body da PKIMessage (Figura 6). Para utilizar apenas uma rota de conversão para todas as mensagens, o sistema utiliza

polimorfismo da estrutura body e ferramentas do framework Spring Boot para fazer a injeção dinâmica de dependências, usando como base a propriedade “type” que está contida no conteúdo do JSON e na mensagem CMP.

Além da conversão, foram adicionadas rotas para adicionar proteção na mensagem, realizar a assinatura de mensagens, realizar assinatura da requisição (para fazer prova de posse) e verificar a validade da assinatura da mensagem CMP. O sistema permite gerar mensagens dos tipos Certificate Request e Revocation Request, e é capaz de converter mensagens dos tipos Certificate Request, Certificate Response, Revocation Request e Revocation Response. Assim, viabilizando a emissão e revogação de certificados digitais via protocolo CMP. As mensagens foram testadas manualmente, utilizando uma AC gerida por uma instância do software EJBCA, que realizou a emissão e revogação de certificados.

5. Resultados

A tabela 1 descreve os endpoints disponíveis.

Tabela 1. Endpoints

Método	Endpoint	Descrição
POST	/converter	Realiza conversão entre JSON e CMP
POST	/pki-message/sign	Assina mensagem CMP
POST	/pki-message/attach-signature	Adiciona assinatura na mensagem CMP
POST	/pki-message/cert-request	Extrai conteúdo do cert request
POST	/cert-request/sign	Assina cert request

5.1. Emissão de Certificado

Para realizar a emissão do certificado, um par de chaves rsa de 2048 bits foi gerado por linha de comando. Em seguida, uma mensagem do tipo “cr”, contendo a chave pública, foi gerada pelo conversor por meio do endpoint “/converter”. O conteúdo retornado foi enviado para o endpoint “/pki-message/cert-request” para obter o conteúdo do certReq, necessário para realizar a prova de posse da chave. A assinatura do certReq foi realizada, utilizando a chave privada, par da chave pública citada anteriormente. A mensagem cmp foi gerada novamente, agora com a assinatura e tipo da prova de posse adicionados no campo “popo” da estrutura “certReq”. Então, o conteúdo retornado foi assinado por meio do endpoint “/pki-message/sign” e a assinatura foi adicionada a mensagem utilizando o endpoint “/pki-message/attatch-signature”, gerando a mensagem completa, assinada e com prova de posse.

A mensagem foi enviada para a EJBCA, que retornou uma outra mensagem, do tipo “cp”. O Certificado emitido está contido no campo “certificate” da resposta.

5.2. Revogação de Certificado

Para revogar o certificado, uma mensagem do tipo “rr” foi gerada. A mensagem possui o serial number do certificado a ser revogado e o motivo da revogação, no caso, chaves comprometidas. O processo para assinatura da mensagem foi realizado conforme descrito na seção anterior. A mensagem foi enviada para a EJBCA, que retornou uma outra mensagem, do tipo “rp”. Esta mensagem foi enviada para o conversor, que gerou um JSON.

A mensagem “rp”, retornada pela EJBCA, confirma que o pedido de revogação foi aceito e o certificado foi revogado.

6. Conclusões

O trabalho realizou a implementação de uma API capaz de converter uma mensagem CMP para uma estrutura JSON e de criar mensagens CMP através de estruturas JSON. A funcionalidade e validação das mensagens foram testadas através de operações de gerenciamento de certificados digitais realizadas no software EJBCA.

Os objetivos do trabalho foram atingidos, uma vez que, com o sistema desenvolvido, é possível converter mensagens e utilizar as mesmas para realizar operações de emissão e revogação de certificados.

A API desenvolvida pode ser incorporada como um serviço interno de uma organização e ser utilizada por aplicações que se comunicam com entidades de ICPs de forma fácil, inclusive, suportando estruturas exigidas pela ICP-Brasil, como os campos definidos para a extensão subject alternative names. A utilização da API elimina a necessidade de trabalhar com estruturas ASN.1, as quais normalmente não possuem suporte nativo em linguagens modernas, tornando o conversor responsável por processar as mensagens. Remove a necessidade de tratar as mensagens CMP dentro dos sistemas apresentados e lidando apenas com estruturas JSON.

O sistema desenvolvido possui pontos que podem ser aprimorados em trabalhos futuros. Os principais pontos são: Suporte a outros algoritmos de proteção, outras opções de prova de posse e a geração de mensagens do tipo “cp” e "rp" e das demais mensagens necessárias para realizar outras operações disponíveis, como a renovação de certificados.

Um outro ponto que pode ser trabalhado é o desenvolvimento de uma documentação da API e um guia para desenvolvedores, para auxiliar no desenvolvimento de novas funcionalidades.

Referências

Adams, C. et al. Internet X.509 Public Key Infrastructure Certificate Management Protocols. IETF, set. 2005. RFC 4210 (Proposed Standard). (Request for Comments, 4210). <http://www.ietf.org/rfc/rfc4210.txt>.

Bellare, M.; Rogawaey, P. Introduction to modern cryptography. Davis, United States of America: University of California at Davis, 2005. Disponível em:

https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf. Acesso em: 03 jul. 2023.

Chokhani, S.; Ford, W. Internet X.509 Public key infrastructure certificate policy and certification practices framework. 1999. RFC 2527.

Diffie, W.; Hellman, M. E. New Directions in Cryptography. IEEE Transactions on Information Theory, IT-22, n. 6, p. 644–654, 1976.

Forouzan, Behrouz A. Introduction to Cryptography and Network Security. Nova Iorque: McGraw-Hill, 2008.

IBM. Digital signatures. 30 de Junho de 2022. Disponível em: https://www.ibm.com/docs/en/ztpf/2022?topic=concepts-digital-signatures. Acesso em: 23 nov. 2022.

IBM Brasil. [s.d.]. O que é criptografia? Definição de criptografia de dados. Disponível em: https://www.ibm.com/br-pt/topics/encryption#:~:text=A%20criptografia%20%C3%A9%20uma%20forma,com%20uma%20chave%20de%20descriptografia.. Acesso em: 03 jul. 2023.

ITU-T. Itu-t recommendation x.509 — iso/iec 9594-8: ”information technology - open systems interconnection - the directory: Public-key and attribute certificate frameworks”. 2008. Relatório técnico.

ITU-T. [s.d.]. Introduction to ASN.1. Disponível em: https://www.itu.int/en/ITU-T/asn1/Pages/introduction.aspx. Acesso em: 03 jul. 2023.

ITI. Obter Certificado Digital. 2022. Disponível em: <https://www.gov.br/pt-br/servicos/obter-certificacao-digital>. Acesso em: 21 nov. 2022.

Kause, T; Peylo, M. Internet X.509 Public Key Infrastructure -- HTTP Transfer for the Certificate Management Protocol (CMP). IETF, set. 2012. RFC 6712 (Proposed Standard). (Request for Comments, 6712). <http://www.ietf.org/rfc/rfc6712.txt>

Stallings, William. Cryptography and Network Security Principles and Practice. 6. ed. Nova Jersey: Pearson, 2013.

The Legion of the Bouncy Castle. [s.d.]. The Bouncy Castle Cryptography Library. Disponível em: https://www.bouncycastle.org/. Acesso em: 03 jul. 2023

Toorani, M.; Shirazi, A.A.B. LPKI - A Lightweight Public Key Infrastructure for the Mobile Environments. 11th IEEE International Conference on Communication Systems (IEEE ICCS'08), p.162-166, Guangzhou, China, Nov. 2008