UNIVERSIDADE FEDERAL DO RIO DE JANEIRO

INSTITUTO DE COMPUTAÇÃO

CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

BRUNO NEVES SARAIVA

GARANTINDO A INTEGRIDADE DE DOCUMENTOS ELETRÔNICOS POR ASSINATURA DIGITAL EM UM SISTEMA ACADÊMICO UNIVERSITÁRIO

RIO DE JANEIRO

2023

BRUNO NEVES SARAIVA

GARANTINDO A INTEGRIDADE DE DOCUMENTOS ELETRÔNICOS POR ASSINATURA DIGITAL EM UM SISTEMA ACADÊMICO UNIVERSITÁRIO

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Aloísio Carlos de Pina

RIO DE JANEIRO

2023

CIP - Catalogação na Publicação

S243g

Saraiva, Bruno Neves
Garantindo a integridade de documentos eletrônicos por assinatura digital em um sistema acadêmico universitário / Bruno Neves Saraiva. -- Rio de Janeiro, 2023.
50 f.

Orientador: Aloísio Carlos de Pina.
Trabalho de conclusão de curso (graduação) - Universidade Federal do Rio de Janeiro, Instituto de Computação, Bacharel em Ciência da Computação, 2023.

1. autenticação. 2. segurança. 3. criptografia. 4. sistemas de informação. 5. documentos digitais. I. Pina, Aloísio Carlos de, orient. II. Título.

Elaborado pelo Sistema de Geração Automática da UFRJ com os dados fornecidos pelo(a) autor(a), sob a responsabilidade de Miguel Romeu Amorim Neto - CRB-7/6283.

BRUNO NEVES SARAIVA

GARANTINDO A INTEGRIDADE DE DOCUMENTOS ELETRÔNICOS POR ASSINATURA DIGITAL EM UM SISTEMA ACADÊMICO UNIVERSITÁRIO

Trabalho de conclusão de curso de graduação apresentado ao Instituto de Computação da Universidade Federal do Rio de Janeiro como parte dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Aprovado em 28 de SETEMBRO de 2023.

BANCA EXAMINADORA:

Aloísio Carlos de Pina, D.Sc
Instituto de Computação - UFRJ

Juliana Baptista dos Santos França, D.Sc
Instituto de Computação - UFRJ

Daniel Chicayban Bastos, D.Sc
Instituto de Computação - UFRJ

AGRADECIMENTOS

Agradeço à Universidade Federal do Rio de Janeiro, em especial ao corpo docente do Instituto de Computação, pela excelência no ensino e democratização do conhecimento científico. Sou grato por fazer parte dessa comunidade extraordinária que, além das contribuições incalculáveis no meu desenvolvimento acadêmico, afirma incessantemente seu compromisso em fazer do mundo um lugar melhor.

Expresso, também, meus mais sinceros agradecimentos à Equipe SIGA UFRJ por seu papel fundamental na minha formação pessoal e técnica. A excelência no desenvolvimento de software e o carinho pelo produto e seus usuários inspiram cada passo de minha jornada profissional.

RESUMO

Desde os períodos embrionários da sociedade moderna, declarações escritas são utilizadas para comprovar estados, condições, fatos e acontecimentos. Por isso, a humanidade sempre buscou formas de garantir segurança e confiabilidade à informação presente nesses documentos de texto, de selos de cera a assinaturas ortográficas. Com o desenvolvimento tecnológico e a crescente popularização do uso de documentos digitais, foi necessário buscar por soluções que levassem para o mundo digital as mesmas garantias do mundo real. O sistema digital de gestão acadêmica da UFRJ, uma das principais instituições de ensino e pesquisa da América Latina, lida com uma quantidade significativa de documentos digitais emitidos todos os dias e, por isso, é fundamental garantir a integridade e autenticidade dessa informação de maneira transparente e segura. Considerando essa necessidade, este trabalho apresenta um serviço de autenticação para o histórico acadêmico, um dos documentos mais robustos emitidos pelo sistema de gestão da universidade. Primeiro, são revisados conceitos matemáticos e algoritmos fundamentais para o pleno entendimento do fluxo proposto. Em seguida, a solução é destrinchada a partir de três pontos de análise que se complementam: a arquitetura de software do serviço, as especificações técnicas da aplicação e, finalmente, a usabilidade da interface do sistema na visão do usuário final.

Palavras-chave: autenticação; segurança; criptografia; sistemas de informação; documentos digitais.

ABSTRACT

Since the embryonic periods of modern society, written statements have been used to prove states, conditions, facts and events. Therefore, humanity has always looked for ways to guarantee security and reliability of the information present in these documents, from wax seals to orthographic signatures. With the technological development and the growing popularization of digital documents, it was necessary to look for solutions that would bring the same guarantees from the real world to the digital world. The digital academic management system used by UFRJ, one of the main teaching and research institutions in Latin America, handles a significant amount of digital documents issued every day and, therefore, it is essential to guarantee the integrity and authenticity of this information in a transparent and safe manner. Considering this need, this work presents an authentication service for student's academic records, one of the most robust documents issued by the university's management system. First, fundamental mathematical concepts and algorithms are reviewed for a full understanding of the proposed flow. Then, the solution is unraveled from three points of analysis that complement each other: the service software architecture, the application technical specifications and, finally, the usability of the system interface from the end user's point of view.

Keywords: authentication; security; cryptography; informational systems; digital documents.

LISTA DE ILUSTRAÇÕES

LISTA DE CÓDIGOS

LISTA DE TABELAS

LISTA DE SIGLAS

3DES	– Triple Data Encryption Standard
AES	– Advanced Encryption Standard
EEA	– Extended Euclidean Algorithm
JDK	– Java Development Kit
MD4	– Message Digest Algorithm
MIT	– Massachusetts Institute of Technology
NIST	– National Institute of Standards and Technology
RSA	– Rivest-Shamir-Adleman
SHA-1	– Secure Hash Algorithm v1
SIGA	– Sistema Integrado de Gestão Acadêmica
UFRJ	– Universidade Federal do Rio de Janeiro
US DOC	– United States Department of Commerce

SUMÁRIO

1 INTRODUÇÃO

Autenticação é um processo de segurança cujo objetivo é garantir validade e autenticidade de um objeto ou indivíduo. Sua função é impedir que dados sejam forjados, informações confidenciais sejam violadas, que um indivíduo tente se passar por outra pessoa e inúmeras outras aplicações nos mais diversos contextos. A palavra tem origem no grego "*authentikos*" – que significa real ou genuíno – e está aplicada, mesmo que de maneira imperceptível, a eventos simples do cotidiano, como reconhecer um amigo de longa data ao encontrá-lo na rua ou, de maneira mais evidente, ao desbloquear o acesso a um dispositivo móvel utilizando biometria facial.

Apesar do grande número de técnicas de autenticação, é fácil perceber que a abordagem geral da maioria dessas técnicas envolve mecanismos de comparação entre dois elementos: aquele cuja procedência não se conhece e que se deseja autenticar e aquele cujas propriedades são conhecidas e reproduzidas de maneira genuína. Portanto, o desafio é garantir confiabilidade a esses processos de comparação, assegurando-se de que sejam assertivos, ou seja, não apresentem falhas de julgamento, e que sejam imunes a tentativas de fraude praticadas por entidades maliciosas.

1.1 CONTEXTO HISTÓRICO

Quando restringimos nossos olhares de maneira mais específica à autenticação de documentos de texto, é necessário levar em conta o processo evolutivo de um elemento que surgiu como um recurso físico, manuscrito, confeccionado a partir de ferramentas primitivas de escrita; evoluiu para versões mais modernas, modeladas por meio de máquinas e equipamentos mecânicos; até por fim se tornar uma sequência eletrônica de símbolos disponível apenas no plano digital.

Durante a idade média, em um dos primeiros processos de autenticação de documentos sobre os quais existem registros históricos, brasões em alto relevo esculpidos em sinetes de ferro eram pressionados contra cera quente formando selos (ou lacres), que eram usados para assinar cartas reais, manuscritos oficiais e documentos de posse (UoN, 2011). Dessa forma, a partir do desenho do brasão no material já solidificado, era possível reconhecer a procedência do documento e garantir confiabilidade ao seu conteúdo. Mesmo que bastante limitada e altamente forjável quando em posse das ferramentas necessárias, essa prática se provou

bastante eficiente e foi adotada como um padrão de segurança por séculos, sendo tradicionalmente utilizada ainda nos dias de hoje por monarcas e famílias reais.

Sob uma perspectiva contemporânea, uma prática antiga de autenticação de documentos que se mantém bastante relevante é o uso de assinaturas ortográficas. Assim como é possível identificar um indivíduo por seu tom de voz ou por suas características físicas, podemos também dizer que a escrita produzida por ele carrega sinais de sua individualidade. Os detalhes e pormenores gráficos da escrita podem ser verificados através de perícias grafotécnicas, que de maneira bastante assertiva podem identificar o punho escritor de origem da assinatura e aferir a autenticidade do traço (TIROTTI, 2021).

No entanto, essas e outras técnicas clássicas de autenticação vem perdendo força numa velocidade exponencial. Com o avanço da tecnologia e a popularização do uso de documentos digitais, o processo de autenticação de documentos tem se tornado cada vez mais complexo e desafiador. Hoje, encontramos com certa facilidade ferramentas extremamente robustas e inteligentes, capazes de reproduzir assinaturas físicas com perfeição e de editar imagens e textos digitais de maneira imperceptível a um olhar não-treinado.

1.2 OBJETIVO

A partir dessas observações acerca da evolução e dos limites da autenticação de documentos nos dias atuais, este trabalho visa a explorar uma alternativa de autenticação fundamentada em técnicas computacionais modernas, e propõe uma solução capaz de garantir proteção, segurança e estabilidade a um sistema de emissão de documentos digitais.

Como inspiração para arquitetar esta solução de software, nos embasamos no estudo de Sharif, Ginting e Dias (2021), que descreve uma aplicação capaz de autenticar documentos eletrônicos por meio de algoritmos de segurança e interfaces *web*. Além disso, muitos conceitos e fundamentações teóricas acerca dos algoritmos utilizados e suas implementações foram obtidos, principalmente, a partir da obra de Paar e Pelzl (2010).

Assim, o texto está organizado da seguinte forma: o segundo capítulo, que vem a logo a seguir, introduz o contexto por trás da solução proposta. São expostos conceitos básicos acerca da autenticação de documentos digitais, bem como o cenário atual de autenticação acadêmica na Universidade Federal do Rio de Janeiro (UFRJ) e suas limitações que motivaram a necessidade de uma nova abordagem de autenticação de documentos eletrônicos. O terceiro capítulo constrói a fundamentação teórica necessária para basear o fluxo de autenticação proposto. Nele são apresentados todos os conceitos matemáticos, softwares e

algoritmos utilizados ao longo da solução de modo a solidificar e embasar o conhecimento aplicado ao capítulo seguinte. Finalmente, o quarto capítulo apresenta a solução arquitetada e sua implementação no sistema acadêmico da universidade, especificando versões de código, dependências externas e o fluxo de execução dos serviços de emissão de documentos e de autenticação. O quinto e último capítulo conclui o texto e adiciona algumas observações finais.

2 DOCUMENTOS DIGITAIS

Nos últimos tempos, a humanidade testemunhou a popularização massiva de documentos no formato digital em detrimento de documentos físicos. A evolução tecnológica e a busca por cada vez mais praticidade e eficiência encontraram no formato digital a solução ideal: documentos digitais podem ser criados e editados a partir de qualquer smartphone ou computador pessoal. Além disso, podem ser armazenados em nuvem – o que garante segurança, redundância e acessibilidade – e transportados de maneira instantânea para qualquer lugar do globo com acesso à Internet. Para além dessa crescente e orgânica adoção, o episódio pandêmico de COVID-19 acelerou ainda mais a necessidade de interações remotas e a adoção de processos digitais, evidenciando que a utilização desses recursos no formato eletrônico parece ser uma tendência global para os próximos anos.

No entanto, junto à popularização dos documentos digitais, surgiu simultaneamente a necessidade de autenticação dessas entidades. Primeiro, para garantir a integridade da informação em jogo, uma vez que esses objetos digitais podem ser forjados e alterados sem deixar resquícios; também porque dados que trafegam publicamente na rede podem ser capturados por entidades maliciosas, gerando fraudes, roubo de identidade e acesso a informações confidenciais. Por isso, é necessário voltar nossos olhares a ferramentas e algoritmos desenvolvidos com o intuito de garantir validade, integridade e confiabilidade a todo e qualquer documento exposto à rede mundial de computadores.

2.1 CRIPTOGRAFIA

Para iniciar nossa análise, precisamos nos debruçar sobre uma área de estudos fundamental, que serve de arcabouço teórico a toda discussão relacionada à segurança digital nos dias atuais: a criptografia. Apesar de abordar temas bastante complexos, o conceito de criptografia está pautado em uma premissa extremamente simples: garantir um canal de comunicação segura entre duas entidades na presença de um ou mais adversários. No entanto, apesar de seu objetivo primeiro e mais notável ser a garantia de privacidade, diversas aplicações em segurança da informação, como autenticação e integridade, são centrais à criptografia moderna.

Historicamente, os primeiros protocolos criptográficos foram desenvolvidos com o intuito de proteger sobretudo informações confidenciais de guerra. A Cifra de César, uma

técnica de substituição de caracteres utilizada pelo imperador romano Júlio César ao transmitir comandos textuais a seus subordinados, talvez seja um dos exemplos mais clássicos dessa arte exata (LUCIANO, PRICHETT, 1987). Contudo, foi com a invenção do rádio e a possibilidade de propagação de mensagens a distâncias globais que se tornou relevante a busca por mecanismos criptográficos de segurança que impedissem, por exemplo, um soldado alemão de interceptar ondas eletromagnéticas e ter acesso a instruções secretas de um general russo. Ao olharmos para trás, podemos afirmar que o curso da Segunda Guerra Mundial foi em algum nível definido pelo uso de sistemas criptográficos em ondas de rádio e, principalmente, pelas primeiras máquinas computacionais desenvolvidas pela inteligência britânica para acessar e descriptografar informações inimigas (LUCIANO, PRICHETT, 1987).

Atualmente, mesmo com o poder computacional exponencialmente superior à época das grandes guerras, fomos capazes de desenvolver diversos algoritmos criptográficos suficientemente seguros, ou seja, que não podem ser violados ou decifrados em tempo hábil. Hoje, nem mesmo o mais potente dos supercomputadores é capaz de encontrar os fatores primos de um número de centenas de dígitos em um período inferior a alguns milhares de anos, o que garante que grande parte dos algoritmos modernos de criptografia digital, que se baseiam nessa premissa, possam ser utilizados com confiança (COUTINHO, 2014).

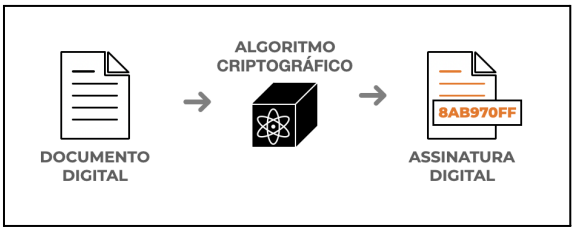
Dado todo esse poder e confiabilidade, é difícil pensar em situações e ações que podemos tomar enquanto usuários sensatos da Internet que não envolvam ao menos um protocolo de segurança. A troca de mensagens por vias seguras passou a ser uma preocupação constante entre nós e definiu padrões criptográficos implementados em inúmeros contextos digitais. Afinal, ao navegarmos por páginas web, estamos nos comunicando com máquinas servidoras a partir do protocolo HTTPS (Hypertext Transfer Protocol Secure), cuja conexão é criptografada e autenticada via certificados digitais. O mesmo vale para compras online utilizando cartões de crédito, uma vez que dados sensíveis como o número do cartão e a data de validade precisam ser protegidos durante a transmissão através da Internet.

2.2 ASSINATURA DIGITAL

No contexto de documentos digitais, ao qual se restringe este trabalho, a solução mais eficaz de autenticação por meio de criptografia se baseia no conceito de assinatura digital. Esse mecanismo foi criado com o objetivo de substituir a assinatura manuscrita, outrora segura, por uma evolução que levasse para o mundo digital as mesmas garantias do mundo

real. No entanto, a simples digitalização de uma assinatura manuscrita não fornece garantia alguma, uma vez que para forjá-la bastaria anexá-la ao documento de interesse a partir de uma ferramenta de edição. Portanto, é necessário atenção ao distinguir conceitualmente uma assinatura digital de uma assinatura física convertida para o formato eletrônico: uma assinatura digital é fruto de uma operação matemática de criptografia que gera, a partir do conjunto de dados contido no documento, uma sequência de caracteres específica e sem valor semântico, de modo que qualquer alteração no documento, por menor que seja, invalida a assinatura original e gera uma nova assinatura drasticamente diferente.

Figura 1: Fluxograma de geração de uma assinatura digital



Ao longo do texto, abordaremos sob um olhar mais técnico e de maneira mais aprofundada alguns mecanismos úteis de criptografia adotados na solução aqui proposta, como as funções unidirecionais (ou hash criptográficas) e também implementações do sistema de criptografia assimétrica por chave pública.

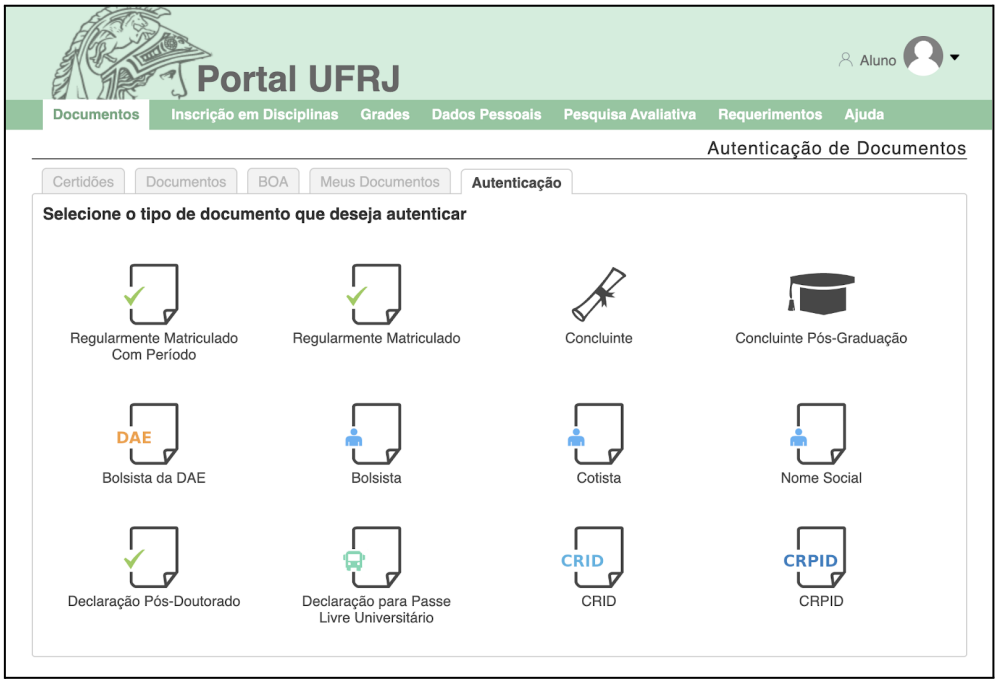
2.3 AUTENTICAÇÃO DE DOCUMENTOS NA UFRJ

A Universidade Federal do Rio de Janeiro, comumente referenciada pela sigla UFRJ, conta com um sistema de informação *online* complexo e robusto o suficiente para suportar cerca de 70 mil alunos matriculados nos mais diversos níveis de ensino e mais de 10 mil funcionários distribuídos entre cargos de docência, saúde e serviços técnico-administrativos (UFRJ, 2022). É a partir do SIGA, acrônimo para Sistema Integrado de Gestão Acadêmica, que é feito o controle das turmas que serão oferecidas aos estudantes, com seus horários e salas de aula, onde são realizadas as inscrições em disciplinas, a partir do qual é feita a visualização da grade horária dos cursos, dentre outras funcionalidades igualmente relevantes para o bom funcionamento da instituição.

Além de realizar a gestão acadêmica, esse portal também disponibiliza uma série de documentos digitais ao corpo discente para controle e acompanhamento universitário, como

históricos acadêmicos e comprovantes de matrícula ativa. Como é de se imaginar, esses documentos têm papel bastante relevante principalmente em processos burocráticos externos à UFRJ. Alguns deles são imprescindíveis para que o aluno comprove a natureza de seu vínculo com a universidade, por exemplo, ao iniciar um programa de estágio, aderir a uma nova conta bancária, realizar processos de transferência acadêmica, etc. Por isso, foi necessário desenvolver mecanismos de autenticação desses documentos digitais que validem a origem e a integridade da informação neles contida.

Figura 2: Interface de autenticação de documentos e certidões gerados pela UFRJ



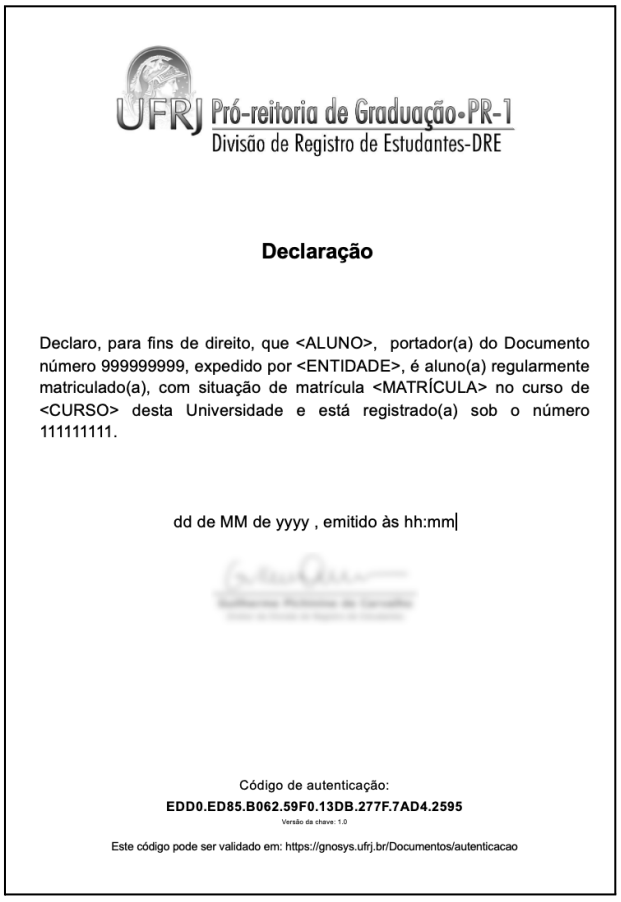
2.3.1 Gerando documentos autenticados

Para gerar um novo documento, o serviço de emissão deve primeiro consultar sua base de dados em busca das informações que irão compor o documento final: o nome completo do aluno, seu número de matrícula, o curso em formação, dentre outros campos que variam conforme a natureza do documento que se deseja emitir. Em posse desses valores, é então executado um algoritmo de *digest* (ou função hash) cujo objetivo é gerar a assinatura digital resultante daquele documento específico.

Como valores de entrada, esse algoritmo recebe 2 parâmetros: as informações necessárias do usuário, consultadas a priori, e uma chave criptográfica confidencial, conhecida apenas pela própria aplicação, a fim de tornar o algoritmo em execução mais

secreto e, por consequência, mais seguro. Por fim, como valor de saída, é retornado ao serviço de emissão a assinatura digital, uma sequência de 32 caracteres hexadecimais, que é anexada ao rodapé do documento e utilizada a posteriori como mecanismo de autenticação do mesmo. Finalmente, é então gerado um documento em formato PDF, de modo que o seu conteúdo está diretamente relacionado à assinatura digital (código de autenticação) que ele carrega.

Figura 3: Exemplo de documento emitido com assinatura digital


UFRJ Pró-reitoria de Graduação•PR-1
Divisão de Registro de Estudantes-DRE

Declaração

Declaro, para fins de direito, que <ALUNO>, portador(a) do Documento número 999999999, expedido por <ENTIDADE>, é aluno(a) regularmente matriculado(a), com situação de matrícula <MATRÍCULA> no curso de <CURSO> desta Universidade e está registrado(a) sob o número 111111111.

dd de MM de yyyy , emitido às hh:mm

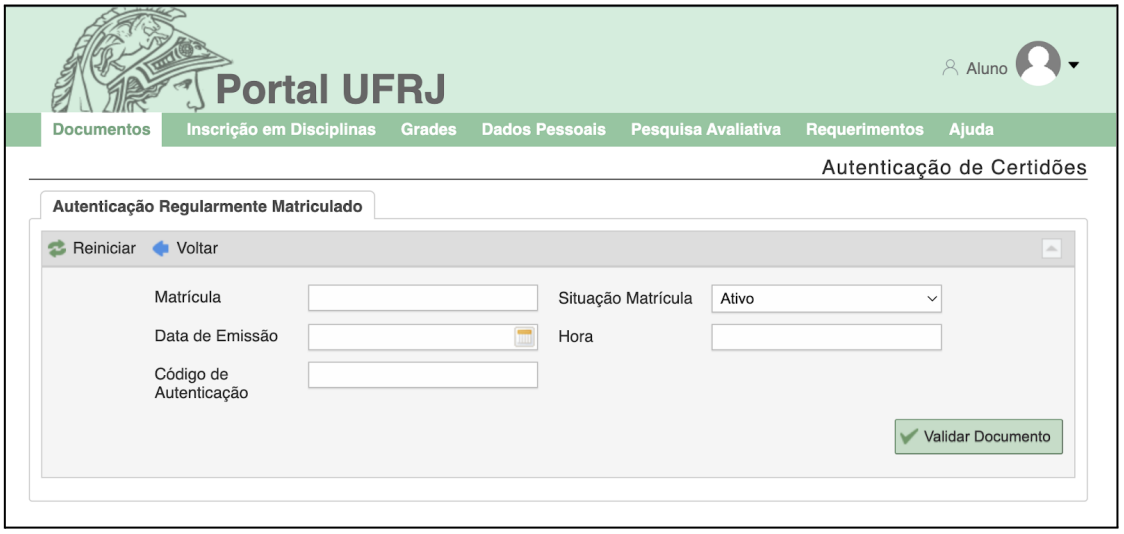
Código de autenticação:
EDD0.ED85.B062.59F0.13DB.277F.7AD4.2595
Versão da chave: 1.0

Este código pode ser validado em: https://gnosys.ufrj.br/Documentos/autenticacao

Uma vez que o documento é gerado, ele pode ser então autenticado a partir do link público indicado no rodapé do documento, que leva ao serviço de autenticação do sistema. Naturalmente, cada tipo de documento possui a sua própria interface de autenticação, uma vez que os dados que os compõem variam em natureza e quantidade. Contudo, a ideia por trás do serviço de autenticação é a mesma entre todos os documentos e acontece da seguinte forma: para garantir que o documento é íntegro e não fora violado, repetimos o mesmo processo de geração da assinatura digital a partir dos campos presentes no documento e então comparamos o resultado dessa nova operação com o resultado obtido originalmente. Dito de outra forma, queremos garantir que aquele conjunto de dados de fato gera aquele código de autenticação ao ser submetido à mesma função hash executada na sua emissão.

Se o resultado obtido for diferente, podemos pressupor dois cenários possíveis: que houve falha humana no preenchimento dos campos do formulário e alguma informação foi lida ou transcrita de maneira equivocada; ou, sendo essa a opção mais alarmante, que o documento foi adulterado ou fraudado. Seja qual for a razão da falha de autenticação, garantimos um mecanismo de validação da informação e confiabilidade aos documentos oficiais emitidos pela universidade.

Figura 4: Exemplo de interface para autenticação de um documento



Atualmente, o Portal UFRJ permite a emissão de 12 tipos diferentes de documentos acadêmicos autenticáveis por meio do fluxo apresentado envolvendo assinaturas digitais.

No entanto, para que essa solução seja satisfatória e amigável a nível do usuário do serviço, é necessário que o documento emitido tenha poucos elementos chave utilizados na geração da assinatura. Não seria interessante emitir um documento com dezenas de campos, uma vez que a interface de autenticação exigiria o preenchimento manual de cada um deles por parte do usuário. Até este momento, os documentos que exigem autenticação são em sua totalidade bastante objetivos, carregando não mais do que meia dúzia de dados autenticáveis, tomando alguns poucos segundos no fluxo de autenticação.

2.3.2 Nova necessidade de autenticação

Recentemente, alguns trâmites administrativos internos à UFRJ, como o processo de mudança de curso, passaram a exigir um novo documento, o histórico acadêmico do candidato pleiteando a vaga, como um dos documentos necessários ao processo de avaliação e classificação. Esse documento passa a ter relevância nesses cenários por possuir dados que

podem ser usados como critério de desempate e até mesmo confirmar que o candidato cumpre certos requisitos acadêmicos exigidos no processo administrativo em que está inscrito.

Sendo um documento emitido de maneira online a partir da conta pessoal do aluno, não existem garantias de que o documento que chega aos avaliadores não tenha sido forjado ou alterado por algum candidato malicioso. Dessa forma, para garantir a integridade dos dados que serão utilizados no processo de seleção, surge a necessidade de autenticar o histórico acadêmico.

Se tentarmos aplicar o fluxo de autenticação já estabelecido no sistema da UFRJ ao histórico do discente, veremos que a condição fundamental para que o serviço funcione bem é absolutamente violada. O histórico é, senão o primeiro, um dos documentos mais robustos e complexos que a universidade emite. Ele armazena, como o próprio nome sugere, todo o histórico acadêmico do aluno: forma de ingresso na universidade; disciplinas cursadas; aprovações, reprovações; coeficientes de presença e de rendimento; monitorias, atividades complementares, etc. Fica evidente a inviabilidade de autenticação desse documento seguindo o fluxo outrora eficaz. A grande quantidade de campos necessários tornaria o preenchimento do formulário de autenticação cansativo e propenso a erros humanos, além de torná-lo extenso o suficiente para que não haja tempo hábil para autenticar um volume grande de documentos de uma vez - tal como em um processo seletivo universitário.

Com isso, surge então uma nova necessidade um pouco mais complexa: autenticar o histórico acadêmico da UFRJ de maneira automatizada, sem a necessidade de preencher todos os seus campos manualmente em um formulário online de autenticação.

Em um primeiro momento, essa tarefa parece ser quase impossível. No cenário problemático em que um candidato decide se inscrever no concurso presencialmente, levando consigo cópias físicas dos documentos necessários ao seu cadastro, como faríamos para autenticar de maneira automatizada um histórico impresso em folhas A4? Seria necessário uma etapa anterior em que iríamos escanear esses históricos físicos e compilá-los de volta para um documento puramente digital a partir de um fluxo de tratamento de imagens e métodos afins. Contudo, esse cenário está fora do escopo deste projeto, uma vez que o processo de cadastro de candidatos nos concursos internos se dá de maneira estritamente digital. Com isso, mesmo que por ora, a necessidade de autenticação abrange apenas documentos digitais, descartando a preocupação com documentos físicos, escaneados, tratamento de imagens, etc. Assim, a solução proposta de autenticação pode atuar estritamente a nível digital, aproveitando-se da estrutura interna de documentos em formato PDF e das vantagens que essa extensão de arquivos oferece.

3 FUNDAMENTAÇÃO

Na nova solução de autenticação proposta neste trabalho, levaremos em conta não só os campos chave do documento que desejamos proteger, mas o arquivo digital como um todo. A cadeia de *bytes* que compõe o PDF do documento será utilizada como entrada em um algoritmo de *digest*, cuja função é reduzir a informação eletrônica a uma sequência randômica de caracteres de tamanho fixo e conhecido. O *output* gerado será utilizado para construção de uma assinatura digital do documento. Utilizaremos o sistema de criptografia RSA como uma camada de segurança ao algoritmo, de modo que a assinatura digital final do documento seja resultado da encriptação do *hash* por meio desse algoritmo criptográfico amplamente conhecido. A seguir estão descritos em detalhes os conhecimentos necessários para arquitetar esta solução de software.

3.1 FORMATO PDF

O PDF (*Portable Document Format*) é sem sombra de dúvidas um dos formatos digitais mais difundidos e utilizados na internet. Sua popularidade se deve, como a sigla sugere, à natureza portátil do formato, que permite que um arquivo possa ser acessado e exibido em qualquer dispositivo, independentemente das especificações de hardware do mesmo, da ferramenta de visualização utilizada e do sistema operacional em execução no dispositivo.

A ideia de desenvolver um software que preservasse o formato original do arquivo em qualquer plataforma surgiu em 1992 na *Adobe Systems Incorporated* (*Adobe Inc*.), uma empresa americana especializada no desenvolvimento de software multimídia. Numa época em que os computadores pessoais eram limitados à exibição de textos e imagens simples, o PDF foi revolucionário no compartilhamento eletrônico de documentos: o formato permitia que usuários incluíssem formatações complexas de textos, fontes personalizadas, layouts e tudo aquilo que tornasse o arquivo mais simples de ser lido em tela. Além disso, tornava possível também o compartilhamento de arquivos entre máquinas diferentes e incompatíveis entre si. Mesmo que muito tenha mudado na internet nas últimas décadas, o formato PDF se estabeleceu amplamente como o padrão de uso para o compartilhamento eletrônico de documentos digitais (CASTIGLIONE, SANTIS, SORIENTE, 2010).

3.1.1 Estrutura de um arquivo PDF

Um arquivo PDF consiste de um conjunto de chaves e objetos que juntos descrevem a aparência e o conteúdo de cada uma de suas páginas. A estrutura de um documento comum possui quatro componentes básicos: um cabeçalho, o corpo do documento, uma tabela de referências e uma seção que localiza a tabela de referências dentro do arquivo (MLADENOV, MAINKA, 2019).

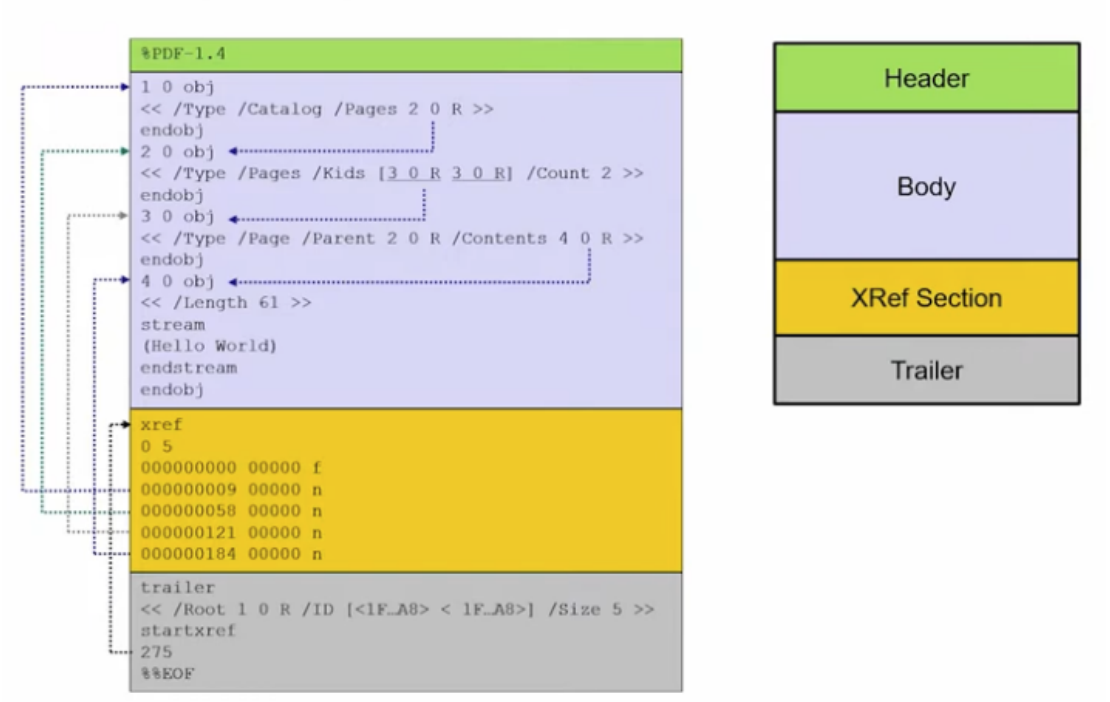
O **cabeçalho** se restringe à primeira linha da estrutura interna do PDF e define a versão do interpretador que deve ser utilizada para ler o documento.

Já o **corpo** do documento especifica seu conteúdo e armazena de maneira estruturada todos os elementos que constituem os dados exibidos em tela: textos, imagens, fontes, vetores gráficos e qualquer outro elemento multimídia. Essa seção do PDF é definida por um conjunto de variáveis internas chamadas de *objetos***.** Cada página do documento é representada por um conjunto de objetos que, por sua vez, são compostos de novos objetos, formando assim a estrutura completa do documento de maneira recursiva. Além disso, essa seção também armazena informações internas do documento que não serão exibidas no modo de visualização, como datas de criação e edição do arquivo, número de páginas, autor do documento, palavras-chave e diversos outros metadados. Apesar de ocultas, essas informações podem ser acessadas posteriormente a partir da tabela de propriedades do arquivo.

A **tabela de referências** contém ponteiros para todos os objetos presentes no documento. O propósito dessa tabela é permitir acesso direto aos objetos, sem a necessidade de percorrer todo o arquivo em busca de algum elemento específico, já que este pode ser utilizado diversas vezes durante a montagem do documento.

Uma vez que um arquivo PDF é lido para a memória, ele deve ser processado de trás para frente, pois a seção que finaliza o documento, chamada de **trailer**, tem como função localizar o início da tabela de referências dentro do arquivo. Essa seção é de suma importância, uma vez que no início da tabela de referências está contido o objeto raiz da seção do corpo, a partir do qual todos os demais objetos serão montados.

Figura 5: Estrutura interna de um arquivo PDF simples (MLADENOV, MAINKA, 2019)



Por conta de seu formato autocontido e universal, que independe de fatores e elementos externos, podemos garantir que arquivos gerados no formato PDF sempre corresponderão à mesma sequência eletrônica de caracteres, sejam quais forem as circunstâncias às quais tiverem sido expostos. Essa garantia é imprescindível se quisermos validar um documento que tenha trafegado na internet ou tenha sido compartilhado entre diferentes contextos de *hardware* e *software*. Se o conteúdo do arquivo for alterado, então ele é considerado violado e invalidado quando em processo de autenticação, o que deve acontecer apenas em casos verdadeiramente maliciosos.

3.2 HASHES CRIPTOGRÁFICAS

Na sua forma mais simples, funções matemáticas definem uma relação entre elementos de um conjunto A e elementos de um conjunto B, formalmente descritas pela expressão $f: A \rightarrow B$, $y = f(x)$. Assim, para qualquer elemento $x \in A$, existe um correspondente $y \in B$ obtido a partir da transformação f. Por esse prisma, funções hash nada mais são do que um tipo particular de função matemática que possui propriedades úteis e aplicáveis a inúmeros ramos da ciência da computação.

3.2.1 Propriedades

Historicamente, o termo **função hash** é utilizado para descrever funções que tem por objetivo comprimir uma mensagem de texto de tamanho variável e imprevisível em um arranjo de *bits* de tamanho reduzido e fixo. O resultado dessa operação, cuja execução tem de ser direta e extremamente eficiente, possui diversas nomeações na literatura: *checksum*, *digest*, *hashcode* ou simplesmente ***hash*** (PAAR, PELZL, 2010).

Figura 6: Exemplos de execução da função hash MD5 para entradas levemente distintas

```
~
$ md5 -s "a simple text"
MD5 ("a simple text") = 6204e84fd12b595bd5384ecda0dd2854

~
$ md5 -s "a simple test"
MD5 ("a simple test") = f0c52da9adc3c4b5f8db6792489b1102
```

Para fazer parte dessa família, uma função deve respeitar algumas propriedades especiais: ser uma função resistente à **inversão**, ser resistente à **segunda inversão** e ser resistente a **colisões**. Apesar de teoricamente ser possível burlar esses comportamentos, por exemplo por força bruta, na prática, algumas funções se provam computacionalmente resistentes a ataques e tornam inviável a quebra dessas condições em tempo hábil.

Uma função $f\colon X \to Y$ é dita inversível quando existe para ela uma função inversa tal que $f^{-1}\colon Y \to X$. Ou seja, o que era domínio na função original vira imagem na inversa, e o que era imagem na original passa a ser domínio na função inversa. Funções hash são conhecidas como funções de "mão única" (ou *one-way functions*) justamente por serem resistentes à inversão: dado um hash R, deve ser inviável encontrar uma mensagem M tal que $R = h(M)$.

Além disso, dado uma mensagem M e seu hash R conhecidos, tal que $R = h(M)$, deve ser computacionalmente inviável encontrar uma segunda mensagem M' tal que $h(M') = R$. Essa propriedade é fundamental para evitar o seguinte cenário: dado um documento eletrônico M_1 que tenha sido aplicado a uma função hash de modo que $R = h(M_1)$, se encontrássemos com facilidade um outro documento $M_2 \neq M_1$ com o mesma

hash $R = (M_2)$, então a assinatura legitimamente gerada para M_1 seria **ilegitimamente válida** para M_2. É fácil imaginar as consequências maléficas desse cenário.

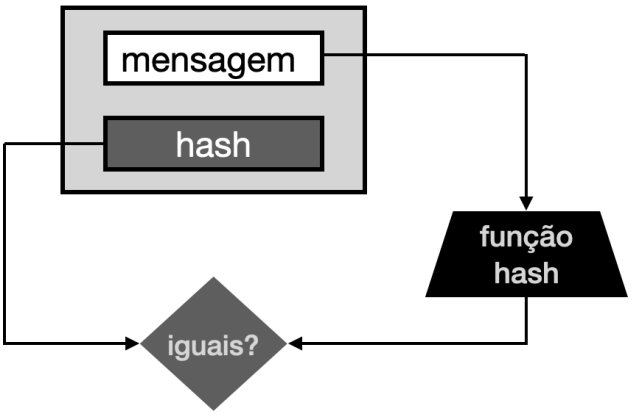
Finalmente, por transformarem um conjunto de entrada de tamanho arbitrário em um conjunto de saída limitado e de tamanho fixo, essas funções estão suscetíveis à colisão de hashes. Ou seja, existem duas mensagens diferentes $M_1 e M_2$ tal que $R = h(M_1) = h(M_2)$. As consequências de uma colisão são imprevisíveis e vão depender do contexto em que a função estiver aplicada. Por isso, é necessário tornar virtualmente impossível que duas mensagens distintas gerem o mesmo *output*.

3.2.2 Utilização

Dentro do contexto de autenticação digital, o objetivo de uma função hash é garantir a integridade do dado por meio de uma **assinatura digital**. O conteúdo do documento que se deseja proteger é utilizado como entrada na função, que retorna em seguida uma sequência de caracteres randômicos e de tamanho fixo. Esse hash é então anexado ao corpo do documento e transportado junto a ele.

Dessa forma, se houver qualquer alteração de dado, ou mesmo que o objeto tenha sido transportado por meio de um canal inseguro, a integridade de seu conteúdo pode ser verificada recalculando a hash resultante e comparando-a com a assinatura presente no documento.

Figura 7: Protocolo de autenticação de uma mensagem por hash



3.2.3 Família MD4

Como mencionado, as funções hash são capazes de processar entradas extremamente extensas e produzir uma saída de tamanho fixo em um curto espaço de tempo. Para isso, o algoritmo segmenta a entrada em uma série de blocos e os computa individualmente em sequência. Cada bloco é processado pela função de compressão em uma execução iterativa, de modo que a hash final é resultante do último laço de iteração.

Ao longo das últimas décadas, diversos algoritmos foram propostos, estudados e até refutados, caso alguma vulnerabilidade fosse identificada. Na prática, as funções que ganharam mais notoriedade e aplicações reais foram as pertencentes à família MD4. O MD4, o algoritmo que dá nome a essa classe de funções e define os princípios adotados por todos os seus algoritmos subsequentes, foi desenvolvido por Ronald Rivest em 1990 e sua popularidade se deve a uma implementação extremamente eficiente, que utiliza apenas variáveis de 32 bits e operações lógicas simples entre bits (*AND, OR, XOR, NOT*). O MD5, a família SHA e diversos outros algoritmos populares são implementações baseadas nesse mesmo princípio.

Um ano mais tarde, em 1991, Rivest propôs talvez a função hash mais popular e difundida na história da criptografia, uma evolução direta e mais segura de sua versão antecessora: o MD5. Apesar do grande número de aplicações em protocolos de segurança de rede, cálculo de *checksums,* e armazenamento de senhas, foram encontrados sinais claros de potenciais falhas, seguidos de diversos ataques de colisão, o que levou ao declínio do uso desse algoritmo alguns anos depois de sua divulgação, levantando a busca por novas versões que contornassem as limitações encontradas.

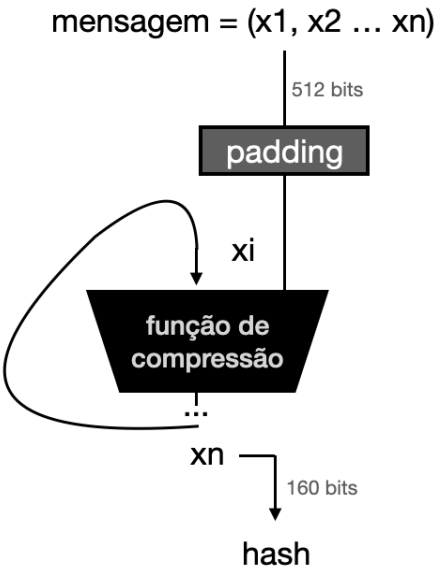
Dessa forma, em 1993, a NIST (National Institute of Standard and Technology), uma agência do Departamento de Comércio Americano (US DOC), publicou um novo padrão que ficou conhecido como SHA (Secure Hash Algorithm). Dois anos mais tarde, em 1995, o algoritmo SHA – ou SHA-0 – deu lugar a seu sucessor, o SHA-1. Desde então, o SHA-1 foi amplamente difundido e é, nos dias de hoje, o algoritmo mais implementado em aplicações e protocolos computacionais (PAAR, PELZL, 2010).

3.2.4 Construção do SHA-1

O algoritmo SHA-1 recebe como entrada uma mensagem de tamanho virtualmente arbitrário, dado seu limite de até 2^{64} bits (2 Exabytes), e produz uma saída de 160 bits na

forma de uma *string* hexadecimal de 40 caracteres. O processo de *digest* de cada um dos blocos que compõem a mensagem se concentra em uma função de compressão executada em 80 rodadas de manipulação divididas em 4 estágios de 20 rodadas cada.

Figura 8: Fluxograma não-exaustivo do algoritmo SHA-1



No entanto, antes de dar início ao processo de compressão, o algoritmo executa um fluxo de pré-processamento para garantir que seja possível dividir a mensagem em blocos iguais de 512 bits. Para isso, é anexado ao final da mensagem uma cadeia de bits de tamanho variável composta de um único bit "1" seguido de n bits desligados e, em seguida, a representação em 64 bits do tamanho da mensagem. O número variável de n bits anexados à mensagem é calculado de modo a tornar a quantidade total de bits um múltiplo de 512. Por exemplo, para uma mensagem de 300 bits, o valor de n deve ser tal que $300 + 1 + n + 64 = 512$, logo $n = 147$. Já para uma mensagem de 513 bits, são utilizados os 512 primeiros para a construção do primeiro bloco, enquanto o último bit é submetido ao processo de *padding*. Seguindo a mesma abordagem, é fácil perceber que $n = 512 - 1 - 1 - 64 = 446$.

Dessa forma, para qualquer mensagem composta por T bits, o número de bits extras pode ser diretamente obtido a partir da expressão

$$n = 512 - ((T + 1 + 64) mod\ 512)$$

Uma vez obtidos os blocos de tamanhos iguais, é então iniciado o processo iterativo de compressão. Esse processo, apesar de bastante robusto e complexo à primeira vista, é

baseado em manipulações lógicas bem definidas. A ideia gira em torno de atualizar exaustivamente 5 variáveis de 32 bits de modo a obter 5 valores finais que, ao serem concatenados, formam a saída de 160 bits do algoritmo. Essas variáveis são inicializadas no começo do programa com valores pré-alocados e são atualizadas a cada iteração. A configuração inicial se dá da seguinte forma:

$$A = 67452301;\ B = efcdab89;\ C = 98badcfe;$$
$$D = 10325476;\ E = c3d2e1f0$$

Cada um dos blocos de 512 bits é então subdividido em 16 elementos de 32 bits tal que $x_i = (x_i^{0}, x_i^{1} \dots x_i^{15})$. Com essa subdivisão, são definidas variáveis de entrada para cada um dos 80 blocos a serem executados, de modo que cada variável W_j é definida por:

$$W_j = \begin{cases} x_i^{(j)} & 0 \le j \le 15 \\ (W_{j-16} \oplus W_{j-14} \oplus W_{j-8} \oplus W_{j-3})_{\ll 1} & 16 \le j \le 79 \end{cases}$$

tal que $\ll n$ representa um deslocamento circular de n bits à esquerda.

Dessa forma, a primeira rodada de compressão para o primeiro bloco será executada a partir da entrada W_0 e dos valores iniciais de $A,\ B,\ C,\ D\ e\ E$. A segunda rodada, por meio de W_1 e dos valores atualizados na primeira iteração de $A,\ B,\ C,\ D\ e\ E$ – e assim por diante. A atualização iterativa dessas variáveis, a parte principal do algoritmo, é definida pela seguinte transformação:

$$A, B, C, D, E = (E + f_t(B, C, D) + (A)_{\ll 5} + W_j + K_t),\ A,\ (B)_{\ll 30},\ C,\ D$$

Os novos elementos $f_t\ e\ K_t$ que aparecem na fórmula de transformação representam, respectivamente, uma função e uma constante relativas ao estágio em execução, variando conforme t. Sabendo que o algoritmo é composto de 80 rodadas divididas igualmente em 4 estágios, é fácil perceber que $1 \le t \le 4$. Os possíveis valores atribuídos a esses dois elementos são descritos na Tabela 1.

Ao final da octagésima rodada de cada bloco de 512 bits, o resultado obtido para as 5 variáveis de 32 bits A, B, C, D e E servirá de entrada para a execução seguinte a partir de uma operação de soma em módulo 2^{32} com os valores inicialmente armazenados por elas no início do algoritmo.

Tabela 1: Constantes e funções utilizadas em cada rodada do SHA-1

t	Rodada j	Constante	Função f_t
1	0…19	$K_1 = 5A827999$	$f_1(B,C,D) = (B \wedge C) \vee (\neg B \wedge D)$
2	20…39	$K_2 = 6ED9EBA1$	$f_2(B,C,D) = B \oplus C \oplus D$
3	40…59	$K_3 = 8F1BBCDC$	$f_3(B,C,D) = (B \wedge C) \vee (B \wedge D) \vee (C \wedge D)$
4	60…79	$K_4 = CA62C1D6$	$f_4(B,C,D) = B \oplus C \oplus D$

Dessa forma, ao final da octagésima iteração do último bloco de 512 bits, os valores alocados nos registradores A, B, C, D e E são concatenados em sequência, formando o hash resultante do SHA-1.

3.2.5 Insuficiência

Funções hash, como apresentado, oferecem mecanismos computacionalmente eficientes de gerar assinaturas para mensagens de tamanho arbitrário. No entanto, são operações que não envolvem chaves secretas ou qualquer tipo de resistência. Em outras palavras, trata-se de um algoritmo genérico que pode ser implementado e reproduzido por qualquer indivíduo que o conheça. Dessa forma, é necessário perceber que, apesar de gerar uma assinatura digital para o documento, essa ferramenta por si só não é suficientemente segura.

Assim, surge a necessidade de utilizar uma camada a mais de proteção à assinatura digital, garantindo que apenas aqueles realmente autorizados possam gerar um exemplar válido. Nesse contexto, um mecanismo de criptografia de chave pública, que irá encriptar a assinatura digital gerada pela função hash, surge na figura do algoritmo RSA.

3.3 CRIPTOGRAFIA RSA

O RSA é sem sombra de dúvidas o algoritmo de criptografia assimétrica mais reconhecido e difundido no meio computacional. Suas aplicações comerciais são incalculáveis, assim como sua relevância para o desenvolvimento da segurança virtual, da

comunicação privada e da popularização do comércio eletrônico. Foi em 1977, no Instituto de Tecnologia de Massachusetts (MIT), que R. L. Rivest, A. Shamir e L. Adleman, três matemáticos criptólogos, desenvolveram esse algoritmo de relevância ímpar para a história recente da humanidade, cujo nome se justifica pelas iniciais do sobrenome de seus criadores: **R**ivest-**S**hamir-**A**dleman (COUTINHO, 2014).

Apesar de matematicamente revolucionário, o RSA é um algoritmo computacionalmente exaustivo e bastante custoso se comparado a opções populares de criptografia simétrica. Por isso, sua implementação se restringe a encriptar dados simples e enxutos como chaves de acesso e sequências alfanuméricas de poucos bytes. Não seria viável, por exemplo, utilizá-lo para proteger um documento extenso de texto como uma ação judicial ou um arquivo multimídia como um filme de *Hollywood* em cartaz.

Dessa forma, no contexto de autenticação de documentos, a limitação do RSA justifica a necessidade de se realizar a compressão dos dados por meio de funções hash como um passo anterior. Com uma sequência curta de caracteres de tamanho conhecido, é possível utilizar o poder e a segurança dessa ferramenta para proteger e garantir a integridade de dados por meio de uma assinatura digital.

3.3.1 Criptografia assimétrica

Assim como diversas outras técnicas e avanços na área de computação e tecnologia, a criptografia assimétrica, também conhecida como criptografia de chave pública, foi introduzida em meados da década de 70. Os algoritmos nessa categoria se baseiam fortemente em princípios relacionados à **Teoria de Números**, além de levar em conta alguns obstáculos de natureza tecnológica que garantem uma aplicabilidade inabalável.

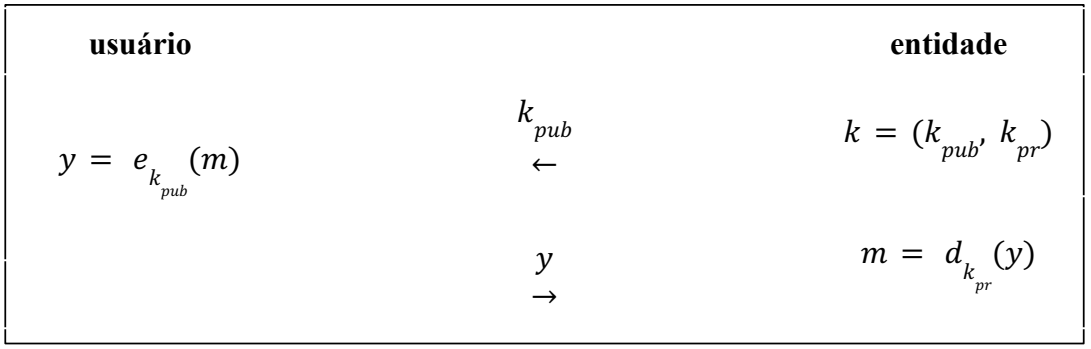
Quando voltamos nossos olhares à criptografia simétrica e a suas limitações, é fácil entender as motivações que levaram à criação de sua recente alternativa. Apesar de seguros e eficientes, algoritmos de criptografia simétrica, como o AES e o 3DES, possuem certos aspectos que os tornam impraticáveis em um mundo em que a comunicação por canais inseguros atingiu proporções globais com o advento da internet.

Primeiro, algoritmos de criptografia simétrica utilizam uma única chave secreta para encriptar e decriptar a mensagem. Portanto, é necessário que essa chave seja estabelecida e compartilhada pelas duas entidades de maneira segura, sem que outros agentes tenham acesso a ela. Evidentemente, uma tarefa que não pode ser realizada de maneira direta quando se trata de comunicação por meio da rede pública de computadores.

Além disso, mesmo que de alguma forma engenhosa tornássemos possível o compartilhamento seguro dessas chaves entre atores, seria necessário criar uma quantidade suficiente de chaves para que cada par de usuários da rede pudessem se comunicar entre si, gerando uma quantidade de chaves da ordem de $n * (n - 1)$, de modo que cada usuário precisaria armazenar $n - 1$ chaves internamente. Em uma corporação privada de médio porte, com alguns milhares de funcionários, a quantidade de chaves necessárias chegaria à casa dos milhões. Quando extrapolamos esse universo internet afora, chegamos a valores astronomicamente impraticáveis.

Para contornar essas deficiências, a ideia central da criptografia assimétrica consiste em particionar a responsabilidade de encriptar/decriptar a mensagem em duas chaves distintas: uma chave **pública** e uma chave **privada**. Nesse formato, uma chave pública é disponibilizada e pode ser acessada e utilizada por qualquer usuário que deseja enviar uma mensagem para a entidade emissora da chave. No entanto, é apenas essa entidade, em posse de uma chave privada conhecida somente por ela (ou por outras entidades autorizadas), que é capaz de decriptar a mensagem e acessar diretamente seu conteúdo. Uma analogia simples que descreve esse formato de maneira bastante intuitiva, é pensar que a entidade que receberá mensagens disponibiliza cópias de um cadeado cuja chave só ela possui. Dessa forma, quando existe a necessidade de comunicação, um usuário escreve sua mensagem e tranca em um cofre utilizando um dos cadeados que a entidade disponibilizou. Assim, quando em um canal inseguro em que mensagens são interceptadas a todo tempo, mesmo que esse compartimento trancado caia nas mãos de entidades maliciosas, o seu conteúdo jamais seria violado, uma vez que a chave necessária para abrir o cofre só pode ser encontrada junto da entidade dona do cadeado.

Figura 9: Protocolo básico de comunicação via criptografia assimétrica



Assim, o RSA implementa uma versão concreta desse fluxograma teórico realizando diversas operações matemáticas que serão descritas a seguir. O conceito de chaves públicas e privadas é traduzido em um algoritmo elegante e virtualmente seguro.

3.3.2 Conceitos

Antes de apresentar o algoritmo RSA propriamente dito, é necessário definir alguns conceitos matemáticos indispensáveis que serão utilizados ao longo de sua execução. Primeiro, é preciso entender que qualquer algoritmo baseado em criptografia por chave pública está fundamentado em alguma técnica matemática fácil de ser executada quando em posse das chaves necessárias, mas muito difícil de ser quebrada por força bruta. No caso do RSA, para que seja possível quebrar a proteção criptográfica, um dos caminhos possíveis e conhecidos é encontrar os fatores primos que compõem um número de centenas de algarismos. Hoje, sabemos que ainda não é conhecido qualquer algoritmo capaz de realizar essa tarefa em tempo hábil.

Ademais, algumas técnicas relacionadas à teoria dos números inteiros são essenciais para o entendimento pleno da criptografia por chave pública – especialmente para o RSA – como o algoritmo Euclidiano, a função ϕ (phi) e o teorema de Euler, assim como o pequeno teorema de Fermat. Esses conceitos serão apresentados de maneira breve ao longo dos próximos parágrafos e servirão de base para a seção que virá logo em seguida.

O **algoritmo Euclidiano** descreve um processo mais eficiente do que a fatoração simples para encontrar o maior divisor comum (*greatest common divisor*) entre dois números inteiros positivos. Para números muito grandes, como aqueles utilizados em esquemas de criptografia por chave pública, a fatorização se torna um processo extremamente custoso e inviável por ser um algoritmo de complexidade exponencial. Como alternativa, o algoritmo de Euclides parte da brilhante observação de que $gcd(r_0, r_1) = gcd(r_o - r_1, r_1)$ para $r_0 > r_1$, cuja prova se dá de maneira simples:

Seja $gcd(r_0, r_1) = g$, como g divide ambos os termos, então podemos reescrever a expressão como $gcd(g.x, g.y) = g$ tal que x e y são coprimos, ou seja, não possuem fatores primos em comum. O ponto chave dessa observação é que x e y se tornam irrelevantes para o resultado final da computação, que se restringe a g, e podem ser manipulados com certa liberdade. Uma maneira prática e eficiente de realizar uma simplificação de variáveis é subtrair um termo do outro, uma vez que $x - y$ e y se mantêm primos entre si. É evidente

que se $x - y$ e y não fossem coprimos, então existiria p que divide ambos. Dividindo os dois fatores por p e somando-os teríamos $(x - y + y)/p$ que simplifica-se a x/p. No entanto, definimos que x e y não podem ser ambos divisíveis por p; o que prova por contradição que $x - y$ e y são primos entre si. Logo, $gcd(g.x,\ g.y) = gcd(g.(x - y),\ g.y) = g$.

Quando aplicamos essa lógica de maneira iterativa, é fácil perceber que $gcd(r_0,\ r_1) = \ldots = gcd(r_0 - mr_1,\ r_1)$ enquanto $r_0 - mr_1$ for maior que zero. Logo, quando escolhemos o valor máximo para m, teremos uma igualdade na forma $gcd(r_0,\ r_1) = gcd(r_0\ mod\ r_1,\ r_1)$ ajustada para $gcd(r_1,\ r_0\ mod\ r_1)$ uma vez que $r_1 > r_0\ mod\ r_1$. Dessa maneira, o processo de encontrar o maior divisor comum se reduz a um par de números cada vez menores, convergindo sempre à $gcd(r_i,\ 0) = r_i$. Logo, $gcd(r_0,\ r_1) = \ldots = gcd(r_i,\ 0) = r_i$.

É possível, ainda, ir um pouco além e perceber que r_i é resultado de uma sequência de operações algébricas entre r_0 e r_1, de modo que pode ser escrito como $r_i = s.r_0 + t.r_1$ tal que s e t são coeficientes inteiros. Portanto, podemos alterar levemente o algoritmo Euclidiano, visto há pouco, a fim de calcular esses novos coeficientes de maneira eficiente. Essa nova versão do algoritmo conhecida como **algoritmo Euclidiano estendido** (EEA) tem papel fundamental na criptografia por chave pública porque, além de encontrar máximos divisores comuns, nos permite computar inversos modulares eficientemente, alicerce para o desenvolvimento do RSA.

O inverso modular de $j\ mod\ M$, denotado por j^{-1}, é tal que $j\ .\ j^{-1} \equiv 1\ mod\ M$ e só pode ser obtido para inteiros coprimos de M, ou seja, com o qual não compartilham fatores primos. Dessa forma, sendo r_0 e r_1 co-primos tal que $gcd(r_0,\ r_1) = 1$, se quiséssemos calcular o inverso de $r_1\ mod\ r_0$ bastaria computar o EEA, cujo resultado é da forma $s.r_0 + t.r_1 = 1$. Aplicando $mod\ r_0$ a essa equação, teremos $s.0 + t.r_1 = 1\ mod\ r_0$ que se reduz à $t.r_1 = 1\ mod\ r_0$, por definição, indicando que t é o inverso modular de $r_1\ mod\ r_0$. Finalmente temos que, sempre que necessário encontrar o inverso modular $j^{-1}\ mod\ M$, basta aplicar o EEA se os inteiros j e M forem coprimos.

A **funçao ϕ de Euler**, ou **função Totiente** é outra ferramenta extremamente útil ao desenvolvimento do RSA. O objetivo dessa função é, dado um inteiro m, avaliar quantos inteiros não-negativos menores que m são primos em relação a ele. Uma abordagem ingênua

para esse problema seria calcular o maior divisor comum entre m e cada um dos naturais de 0 a $m - 1$, computando as iterações em que o resultado dessa operação for igual a 1. Apesar de correto, esse processo se torna claramente custoso e inviável a medida que m assume valores muito grandes.

No entanto, se conhecidos os fatores primos que compõem m, então o resultado da função Totiente pode ser calculado de maneira direta a partir de uma operação computacionalmente leve. Seja m disposto na sua forma canônica de fatoração, isto é, $m = p_1^{e1}.p_2^{e2}.\ldots p_n^{en}$, então

$$\phi(m) = \prod_{i=1}^{n} (p_i^{ei} - p_i^{ei-1}).$$

Pondo de forma prática, dado $m = 240 = 16.15 = 2^4.3.5$, então $\phi(m) = (2^4 - 2^3).(3^1 - 3^0).(5^1 - 5^0) = 64$. Sendo assim, no intervalo fechado de 0 a 239, existem 64 números que não compartilharam fatores primos com $m = 240$.

Por fim, o **pequeno teorema de Fermat** se faz presente principalmente por ser possível realizar testes eficientes de primalidade a partir de sua fórmula.

Dado um inteiro a e um primo p, então por Fermat, temos que $a^p \equiv a \; mod \; p$. Ou, de maneira mais útil para contextos criptográficos, que $a^{p-1} \equiv 1 \; mod \; p$. Se reescrevermos essa equação na forma $a.a^{p-2} \equiv 1 \; mod \; p$, temos por definição o inverso modular de a, ou seja, $a^{-1} \equiv a^{p-2} \; mod \; p$.

Uma generalização do pequeno teorema de Fermat aplicada a módulos primos e não-primos é o **teorema de Euler**: dados dois inteiros a e m primos entre si, então $a\hat{}\phi(m) \equiv 1 \; mod \; m$. É fácil perceber que o pequeno teorema de Fermat é um caso especial do teorema de Euler. Sendo p um primo, então $\phi(p) = (p^1 - p^0) = p - 1$. Aplicando esse resultado ao teorema de Euler, temos que $a\hat{}\phi(p) \equiv a^{p-1} \equiv 1 \; mod \; p$, que é exatamente o pequeno teorema de Fermat. $$a^{\varphi(m)} \equiv 1 \pmod{m}$$

3.3.3 Algoritmo

Antes de dar início ao fluxo de encriptação e decriptação de mensagens, é necessário passar por uma fase inicial de geração das chaves pública e privada que serão utilizadas. Primeiro, deve-se escolher um par de números primos que chamaremos p e q. Já está evidente a necessidade de que p e q sejam números extremamente grandes, mas isso não é condição suficiente: se o módulo $|p - q|$ for pequeno, então a fatoração do produto entre eles se torna

mais simples e a segurança do algoritmo cai por terra. Portanto, é necessário que p e q estejam distantes um do outro. Além disso, outras garantias são necessárias para reforçar ainda mais a robustez da escolha como ter certeza de que $p-1, q-1, p+1$ e $q+1$ não possuem fatores primos pequenos.

Em cenários do mundo real, é recomendado que se utilize bibliotecas criptográficas e frameworks bem estabelecidos para ter acesso a números primos seguros. Essas ferramentas lidam com a geração de números primos extensos de maneira segura e eficiente, mitigando vulnerabilidades de segurança já conhecidas e garantindo que os pontos de atenção necessários para uma implementação adequada do RSA sejam satisfeitos. Como um exemplo de biblioteca com essas características, naturalmente vem à tona a popular ferramenta ***OpenSSL***, um software de código aberto que comporta uma extensa prateleira de algoritmos de segurança e que suporta inúmeras linguagens de programação populares como C, C++, Python e Java. Além disso, o OpenSSL é implementado em aplicações conhecidas da comunidade como servidores *Apache,* NGINX e diversos sistemas operacionais baseados em Linux, como o Ubuntu e o Debian.

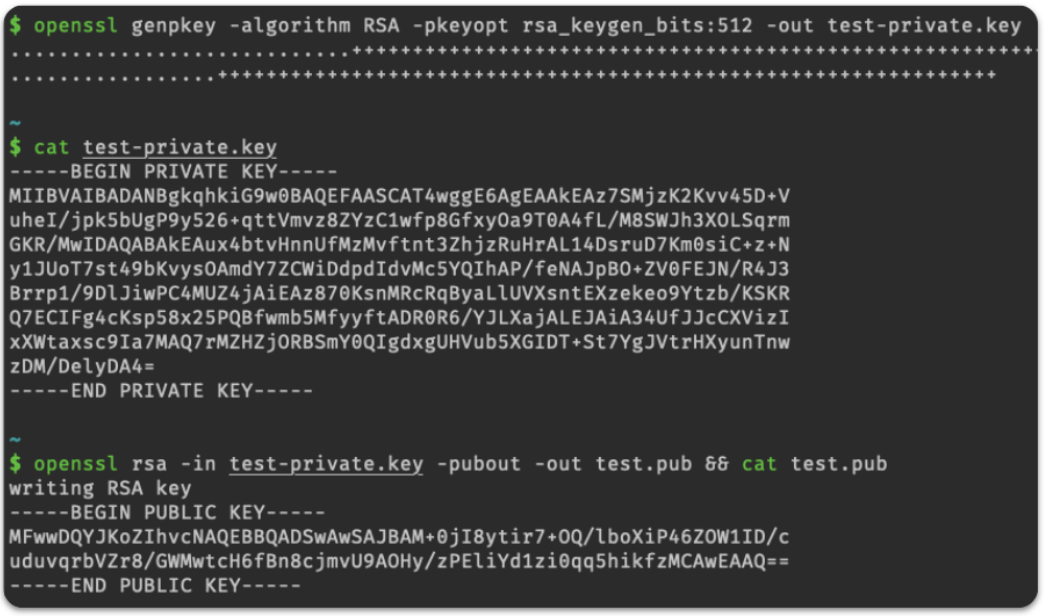
Com os números p e q definidos, deve-se realizar o produto entre eles a fim de se obter $n = p.q$, cujo valor será utilizado como módulo para obtenção tanto da chave pública quanto da chave privada. Em seguida, deve ser computado o total de números coprimos com relação a n no intervalo $[0, n-1]$. Por ser composto pelos fatores primos p e q, então a função *phi* de Euler se dá de maneira direta e $\phi(n) = (p-1).(q-1)$. O resultado de $\phi(n)$ é essencial para os próximos passos.

Finalmente, é preciso selecionar o expoente público $e \in \{1, 2,..., \phi(n)-1\}$ tal que $gcd(e, \phi(n)) = 1$ e em seguida calcular a chave privada d de modo que $d.e \, mod \, \phi(n) = 1$. Venturosamente, ambas incógnitas podem ser calculadas de uma só vez a partir do algoritmo Euclidiano estendido (EEA). Primeiro, é preciso selecionar um candidato para o parâmetro e no intervalo $0 < e < \phi(n)$ que satisfaça a condição inicial $gcd(e, \phi(n)) = 1$. Assim, aplicando o EEA para e e $\phi(n)$ tem-se a relação 1.

$$gcd(\phi(n), e) = s.\phi(n) + t.e \quad (1)$$

E portanto, se $gcd(e, \phi(n)) = 1$, temos a garantia de que e é uma chave pública válida. Mais do que isso, sendo e e $\phi(n)$ coprimos, então t representa o inverso modular de $e \, mod \, \phi(n)$, de modo que $t.e \, mod \, \phi(n) = 1$. Portanto, t satisfaz justamente a condição imposta para o valor atribuído ao expoente privado d. Logo, $d = t \, mod \, \phi(n)$.

Figura 10: Gerando chaves pública e privada para o RSA a partir do OpenSSL



Com n, e e d calculados, temos todas as ferramentas necessárias para dar início ao fluxo criptográfico do RSA. A chave pública a ser utilizada, composta pelo par (n, e) deve ser utilizada para encriptar a mensagem M a partir da função

$$f_{enc} = M^{e} \, mod \, n = y \tag{2}$$

De maneira análoga, a chave privada, resultante da combinação matemática entre n e d, cujo objetivo é reverter a encriptação descrita anteriormente, é descrita pela função

$$f_{dec} = y^{d} \, mod \, n = M \tag{3}$$

Dessa forma, temos finalmente as chaves pública e privada definidas e prontas para serem utilizadas para o tráfego seguro de informação.

3.3.4 Assinatura digital

Além de ser utilizado para criptografar e proteger mensagens ao transmiti-las por canais inseguros, o RSA também pode ser utilizado como uma ferramenta capaz de assinar documentos digitais. Ao serem utilizados nesse contexto, algoritmos criptográficos de chave pública, como o RSA, são capazes de garantir diversos serviços de segurança extremamente importantes para a autenticação de documentos como integridade, identificação da origem de emissão, não-repúdio e confidencialidade.

Ao garantir a integridade da informação assinada, o RSA protege mensagens que trafegam por ambientes inseguros de serem modificadas ou forjadas. Mesmo que um elemento malicioso tenha acesso ao pacote em trânsito e altere seu conteúdo, o RSA é capaz de identificar que a informação presente no documento falsificado foi alterada e portanto que a assinatura digital que ele carrega é inválida naquele contexto.

Além disso, a assinatura digital da mensagem é gerada a partir da chave privada estabelecida, de modo que apenas a entidade emissora é capaz de assinar documentos em seu nome. Portanto, como cada chave secreta identifica unicamente a origem da emissão do documento, não há como repudiar sua originalidade. Ao decriptar a mensagem por meio da chave pública, é fácil validar que a assinatura foi de fato gerada pela chave privada da origem, uma vez que apenas a chave privada correta geraria uma mensagem válida quando decriptada pela chave pública.

Para garantir confidencialidade, é necessário um passo anterior à assinatura, uma vez que o protocolo básico descrito pelo RSA para assinatura digital não provê qualquer tipo de sigilo ao conteúdo trafegado. Por meio de algoritmos de *digest* ou criptografia simétrica, é possível obter proteções dessa natureza, de maneira que a assinatura digital passa a ser calculada a partir do *output* dessa etapa inicial de proteção da informação.

Figura 11: Protocolo básico de assinatura digital por RSA

usuário		**entidade**
	$(n,\ e)$ $\leftarrow$	$k_{pub} = (n,\ e),\ k_{pr} = (n,\ d)$
	$(m,\ s)$ $\leftarrow$	assina a mensagem: $s = sig_{k_{pr}}(m) \equiv m^{\mathrm{d}}\ mod\ n$
verifica mensagem: $x = s^{\mathrm{e}}\ mod\ n$ $x = m\ mod\ n$? válida $x\ != m\ mod\ n$? inválida		

O protocolo básico para autenticação de assinaturas digitais por RSA, descrito na Figura 11, tem como passo inicial a escolha das chaves pública e privada. Esses parâmetros podem ser definidos seguindo as mesmas metodologias e boas práticas apresentadas na execução clássica do algoritmo.

Para gerar a assinatura digital para uma mensagem m, a entidade geradora deve encripta-la a partir de sua chave privada k_{pr}. Como a entidade é a detentora da chave privada e, portanto, a única capaz de gerar uma assinatura digital a partir dela, este passo garante o não-repúdio da assinatura e a autenticação do órgão autor. A assinatura é então anexada à mensagem e ambas são enviadas (ou disponibilizadas) para um usuário interessado em obtê-las. Por sua vez, o usuário em posse da mensagem assinada decripta sua assinatura utilizando a chave pública da entidade assinante, obtendo por fim uma versão x da mensagem. Sendo x equivalente à mensagem original, o usuário consegue afirmar duas coisas: que a mensagem original fora autenticada utilizando a chave privada da entidade, atestando sua autoria, e que a mensagem não fora forjada ou alterada, atestando sua integridade.

4 SOLUÇÃO

Finalmente, todos os conceitos e ferramentas destrinchados até aqui foram condensados em um sistema de segurança que soluciona a necessidade inicialmente estabelecida: ***garantir a integridade de documentos acadêmicos digitais***.

A seguir, a solução proposta será analisada a partir de três abordagens distintas: arquitetura, implementação e usabilidade. A arquitetura da solução é composta de um fluxograma não-exaustivo contendo as principais etapas envolvidas no sistema de emissão da assinatura digital e de autenticação. Já a seção de implementação se atém ao código fonte da aplicação e a comentários relacionados a especificações técnicas, algoritmos e programas utilizados. Por fim, será apresentada a interface do serviço no Portal UFRJ, a partir da qual usuários do sistema poderão realizar a emissão e a verificação do histórico acadêmico de maneira transparente e amigável.

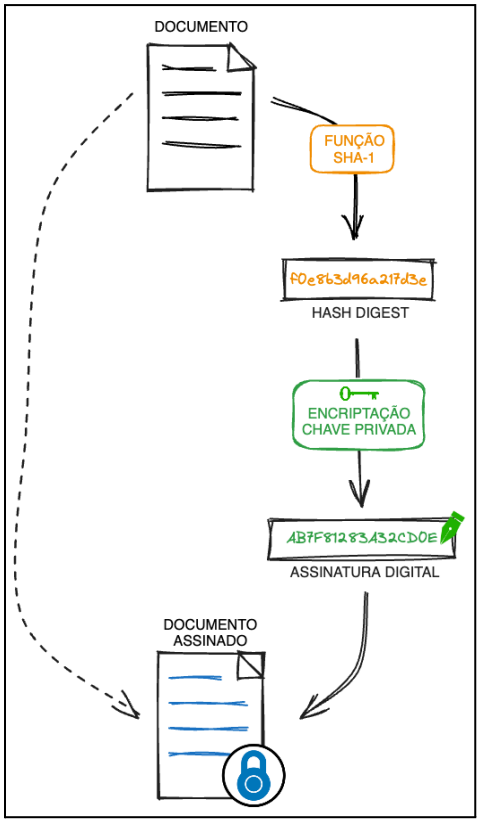
4.1 ARQUITETURA

A adição da assinatura digital no documento acontece somente após a etapa de montagem do mesmo. Nessa etapa preliminar, são coletados os dados referentes ao histórico acadêmico do discente, como forma de ingresso na Universidade e disciplinas cursadas em cada período letivo. A partir desses valores e de um template pré definido, o documento PDF é então montado, dando início ao fluxo de escrita da assinatura digital.

4.1.1 Emissão

Para emitir um documento assinado, é necessário realizar o processo de *digest* do arquivo, reduzindo seu conteúdo a uma sequência de bytes de tamanho fixo e conhecido. Para tal, é preciso computar sua hash equivalente por meio de uma execução do algoritmo SHA-1. Em seguida, é gerada uma assinatura digital a partir do *hash* obtido utilizando a variação do RSA para assinaturas digitais e a chave privada da aplicação previamente calculada e armazenada. Finalmente, a assinatura digital é anexada ao documento como um metadado interno, tornando-o pronto para emissão. Esse fluxograma pode ser observado integralmente na Figura 12.

Figura 12: Fluxograma de emissão do boletim acadêmico assinado



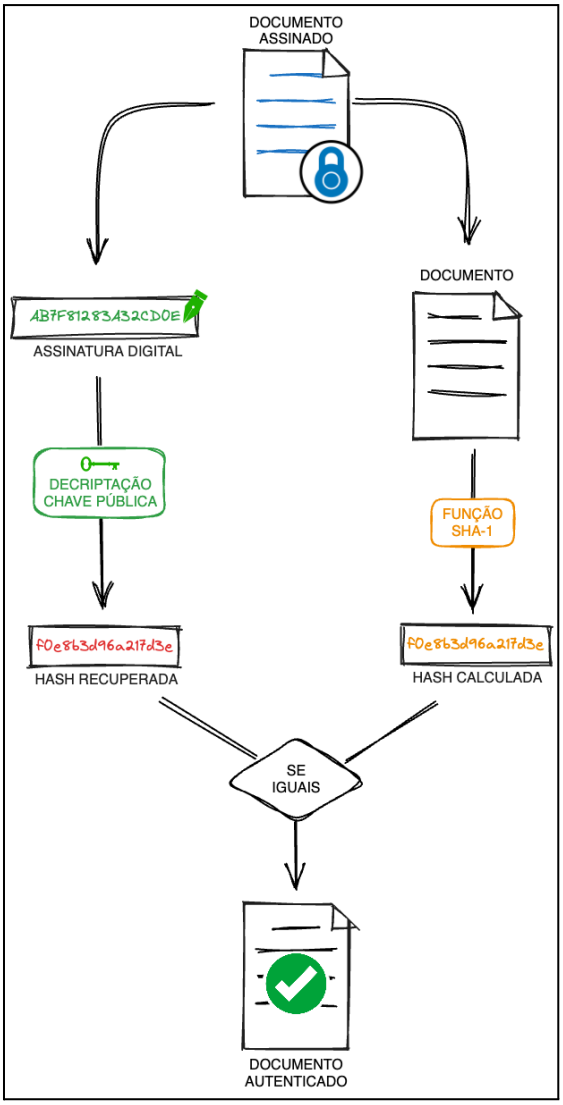
4.1.2 Autenticação

Já no processo de autenticação do documento assinado, o algoritmo de checagem se divide em dois fluxos executados a princípio em paralelo, de modo que ao resultarem na mesma resposta, atestam a integridade do documento em análise. A Figura 13 ilustra a execução desse fluxograma.

Para dar início a esses fluxos, é necessário primeiro extrair a assinatura digital do documento, separando por completo as duas entidades. Assim, a partir da assinatura digital e da chave criptográfica pública da aplicação, é possível realizar o processo de decriptação da assinatura, que retorna como saída a hash interna à assinatura. Enquanto isso, o documento bruto deve ser submetido ao mesmo processo de *digest* apresentado no fluxo de emissão, de modo a obter uma nova hash resultante desta segunda execução do SHA-1.

Por fim, é realizado um processo de comparação entre a nova hash recém calculada e a hash extraída da assinatura digital. Caso ambas sejam iguais, então é possível dizer que o documento não foi violado e que seu conteúdo é autêntico e genuíno. O caso contrário indica que o documento foi violado e que seu conteúdo não é equivalente à assinatura que ele carrega.

Figura 13: Fluxograma de autenticação da assinatura digital no boletim acadêmico



4.2 IMPLEMENTAÇÃO

O sistema de emissão de documentos do Portal UFRJ foi desenvolvido a partir de um web framework interno construído em Java 7. Para a criação dos relatórios que geram os arquivos acadêmicos foi utilizada a biblioteca *open source* Jasper Reports na versão 6.5.1. Já para a manipulação dos PDFs e adição da assinatura digital como um metadado customizado, importamos a dependência Apache PDFBox na versão 2.0.6.

A linguagem Java oferece no seu kit de desenvolvimento (JDK) um utilitário de linha de comando denominado *keytool*, utilizado para gerar e gerenciar chaves de criptografia e certificados digitais. A ferramenta permite que usuários administrem internamente pares de

chaves público-privada usados em serviços de auto-autenticação e de checagem de integridade de dados por assinatura digital.

O Código 1 contém o comando executado a partir da ferramenta *keytool* para gerar as chaves criptográficas da aplicação. Foram atribuídos alguns parâmetros extras ao comando, como senhas de acesso ao registro e uma especificação do algoritmo que fará uso das chaves. Por fim, o arquivo gerado contendo as informações de segurança do serviço será armazenado no ambiente de execução da aplicação para acesso às chaves do RSA.

Código 1: Comando para gerar um Java Keystore contendo chaves criptográficas

```
$ /usr/lib/jvm/j2sdk1.7-oracle/bin/keytool -genkeypair
-alias siga-rsa-key
-storepass <STORE_PASSWORD>
-keypass <KEY_PASSWORD>
-keyalg RSA
-keystore rsa-keystore.jks
```

Em complemento à criação das chaves, é necessário que a aplicação consiga acessá-las em tempo de execução. Para tal, é preciso definir um método estático e global que retorne o par de chaves no contexto do código. O fluxo desse procedimento utiliza a dependência ***java-security***, a biblioteca de segurança padrão do java, para a manipulação do arquivo de segurança e encapsulamento das chaves recuperadas em uma classe utilitária do próprio pacote.

A partir do arquivo de chaves e das senhas configuradas, tanto a chave pública quanto a chave privada podem ser acessadas de maneira direta e transparente, como exemplificado no Código 2. É importante observar que não existe a necessidade de recuperar ambas as chaves de maneira conjunta – na verdade, o ideal é que sejam obtidas em pontos diferentes e independentes do código. Dessa forma, a chave privada deve ser recuperada somente no contexto em que se faz necessária, ou seja, na assinatura do documento, enquanto a chave pública deve ser acessada apenas no fluxo de autenticação. Com isso, fica claro o papel de

cada entidade dentro do contexto do serviço, além de desacoplar os fluxos de assinatura e autenticação.

Código 2: Acessando pela aplicação chaves RSA armazenadas no Java Keystore

```java
public static KeyPair getKeys() {
  InputStream is = RSA.class.getResourceAsStream("rsa-keystore.jks");

  // load key store
  KeyStore keyStore = KeyStore.getInstance("JCEKS");
  keyStore.load(is, "STORE_PASSWORD".toCharArray());

  // private key
  PasswordProtection keypass = new PasswordProtection("KEY_PASSWORD".toCharArray());
  PrivateKeyEntry privateKeyEntry = (PrivateKeyEntry) keyStore.getEntry("siga-rsa-key", keypass);
  PrivateKey privateKey = privateKeyEntry.getPrivateKey();

  // public key
  PublicKey publicKey = keyStore.getCertificate("siga-rsa-key").getPublicKey();

  return new KeyPair(publicKey, privateKey);
}
```

4.2.1 Emissão

Para assinar um novo histórico acadêmico, é necessário primeiro gerar um relatório *jasper* referente ao documento. Esse relatório é montado a partir de um arquivo template acessível pela aplicação e do conjunto de dados acadêmicos do aluno dono do histórico. Após gerado, o relatório é exportado no formato PDF para um *buffer* de *bytes*, que será utilizado para criar uma nova instância da classe ***PDDocument*** (obtida a partir da biblioteca PDFBox do Apache). Essa estrutura possui, entre outras funcionalidades, a capacidade de criar e manipular metadados internos ao PDF. Desse modo, é possível incluir a assinatura digital dentro do documento como um novo metadado customizado.

Evidentemente, antes da adição da assinatura ao documento, é preciso emiti-la a partir de sequência atual de bytes do próprio. Como passo anterior à geração da assinatura, no entanto, é preciso adicionar ao documento o metadado referente a ela armazenando um valor default, tal como um *placeholder*. A necessidade deste passo ficará clara durante a etapa de autenticação. De maneira breve, se tentarmos autenticar um documento com a assinatura referente a ele dentro do mesmo, cairemos em um dilema de causalidade em que o documento em posse de sua assinatura deve gerar a própria assinatura. Por isso é preciso desacoplar a assinatura do documento para autenticá-lo e, para tal, a assinatura é substituída pelo

placeholder. Assim, adicionar o *placeholder* antes de computar a assinatura garante que o documento se encontrará no mesmo estado antes de ser assinado e no momento de autenticação.

Código 3: Emissão de um documento assinado digitalmente

```
public static byte[] buildDocument() {
    // build document from report
    JasperPrint report = JasperFillManager.fillReport(<JASPER_FILE_PATH>, <REPORT_DATA>);
    JRPdfExporter exporter = new JRPdfExporter();
    exporter.setParameter(JRExporterParameter.JASPER_PRINT, report);

    // export document to byte stream
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    exporter.setParameter(JRExporterParameter.OUTPUT_STREAM, baos);
    exporter.exportReport();

    // add placeholder to pdf metadata
    PDDocument doc = PDDocument.load(baos.toByteArray());
    doc.getDocumentInformation().setCustomMetadataValue("sign", <SIGNATURE_PLACEHOLDER>);
    baos.reset();
    doc.save(baos);

    // compute hash from document bytes
    MessageDigest md = MessageDigest.getInstance("SHA-1");
    md.update(baos.toByteArray());
    byte[] hash = md.digest();

    // sign hash
    Signature sig = Signature.getInstance("NONEwithRSA");
    sig.initSign(<PRIVATE_KEY>);
    sig.update(hash);
    byte[] signature = sig.sign();

    // add signature to pdf metadata
    String signatureHexadecimal = DatatypeConverter.printHexBinary(signature);
    doc.getDocumentInformation().setCustomMetadataValue("sign", signatureHexadecimal);
    baos.reset();
    doc.save(baos);

    // export signed document
    return baos.toByteArray();
}
```

O cálculo da assinatura a partir do bytearray equivalente ao documento acontece de maneira direta. Primeiro, é realizado o *digest* do *stream de* bytes utilizando uma implementação da função hash SHA-1 definida no pacote *java.security*. Em seguida, o hash é enviado a uma outra classe de segurança do pacote que, junto à chave privada, computa a assinatura a partir do algoritmo RSA. Por fim, o documento é atualizado com a assinatura e retornado pela função como um novo array de bytes a ser emitido pelo serviço de emissão.

4.2.2 Autenticação

O processo de autenticação de um histórico acadêmico digital se inicia com o upload do arquivo PDF a partir da interface do serviço. Após o upload, o fluxo de verificação se inicia. Primeiro, o arquivo é compilado para uma instância da classe ***PDDocument*** para possibilitar a extração da assinatura que ele armazena. Prontamente, a assinatura é extraída e substituída por seu *placeholder*. Assim, a partir do documento no seu estado pré-assinatura, é feito o cálculo de um nova hash equivalente à sequência atual de bytes carregada.

Finalmente, é realizado o processo de autenticação. Para tal, a hash protegida pela assinatura digital é recuperada a partir da chave pública e, em seguida, comparada à hash calculada no passo anterior. A verificação desses valores é retornada pela função de autenticação e tratada do lado do serviço.

Código 4: Autenticação de documento digitalmente assinado

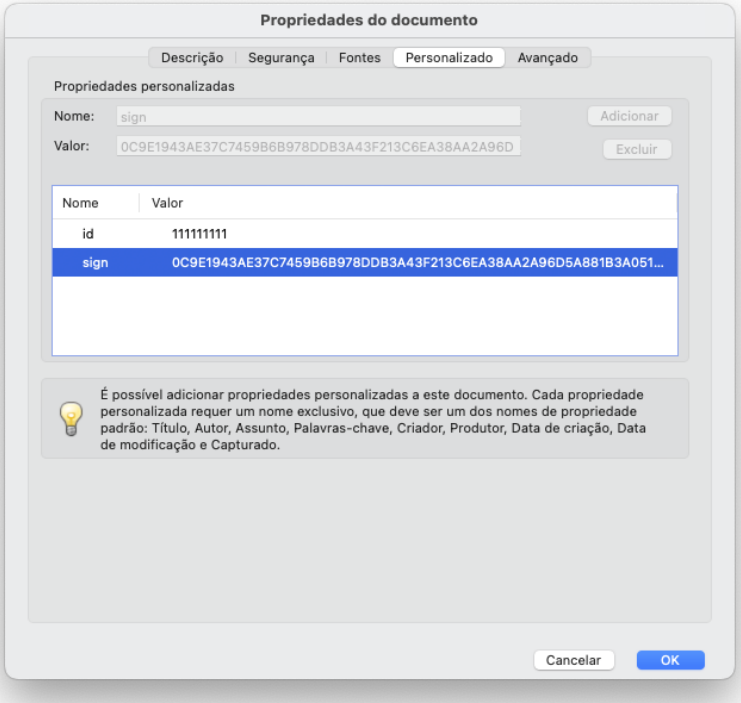
```
public static boolean validateDocument() {
    // initialize document instance
    byte[] bytes = FileUtils.readFileToByteArray(<UPLOADED_DOCUMENT>);
    PDDocument doc = PDDocument.load(bytes);

    // get signature and replace with placeholder
    String signText = doc.getDocumentInformation().getCustomMetadataValue("sign");
    byte[] sign = DatatypeConverter.parseHexBinary(signText);
    doc.getDocumentInformation().setCustomMetadataValue("sign", <SIGNATURE_PLACEHOLDER>);

    // document to byte array
    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    doc.save(baos);

    // compute new hash
    MessageDigest md = MessageDigest.getInstance("SHA-1");
    md.update(baos.toByteArray());
    byte[] hash = md.digest();

    // authenticate
    Signature sig = Signature.getInstance("NONEwithRSA");
    sig.initVerify(<PUBLIC_KEY>);
    sig.update(hash);

    return sig.verify(sign);
}
```

4.3 INTERFACE

A interface dos serviços de emissão e autenticação do histórico acadêmico, bem como de todos os outros documentos e certidões que o Portal UFRJ é capaz de gerar, respeita uma padronização de cores e componentes utilizados por todas as páginas do sistema de gestão acadêmica. Da mesma forma, a interface do documento autenticável segue premissas respeitadas por todos os demais documentos, como a inserção da marca d'água da Universidade.

Para emitir um novo histórico acadêmico, basta que o discente esteja logado em sua conta pessoal e acesse a página de emissão de documentos apresentada na seção 2.3 do texto. Ao ser emitido, o documento carregará a assinatura digital como um metadado customizado interno à estrutura do PDF. Como demonstrado na Figura 14, é possível visualizar essa assinatura acessando as propriedades do arquivo a partir de uma ferramenta de visualização, como o *Adobe Acrobat Reader*.

Figura 14: Visualização das propriedades personalizadas do PDF no *Acrobat Reader*



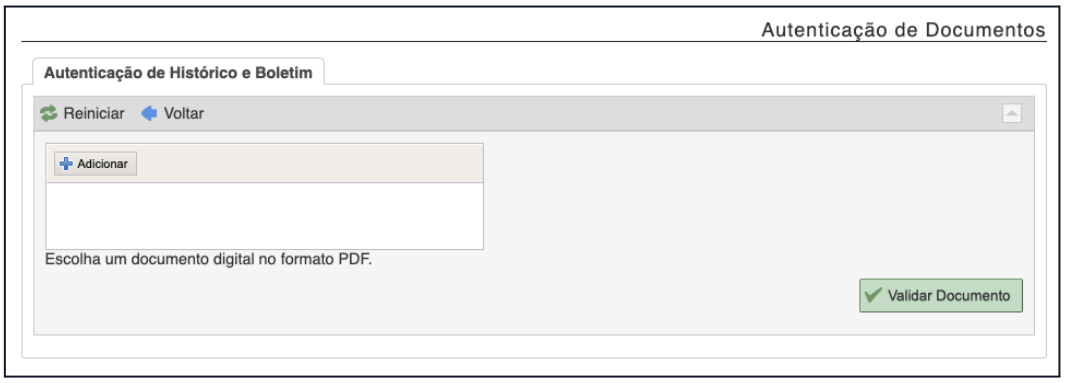
Além da assinatura digital, o identificador de registro do aluno também foi adicionado ao documento emitido. Com isso, é possível resgatar de maneira simples e direta o detentor

do histórico. Dessa forma, no fluxo de autenticação, ao ser realizada a verificação do documento, alguns dados de identificação referentes ao detentor do mesmo (foto, nome completo, etc) são exibidos em tela, garantindo uma camada a mais de validação e segurança.

4.3.1 Autenticação

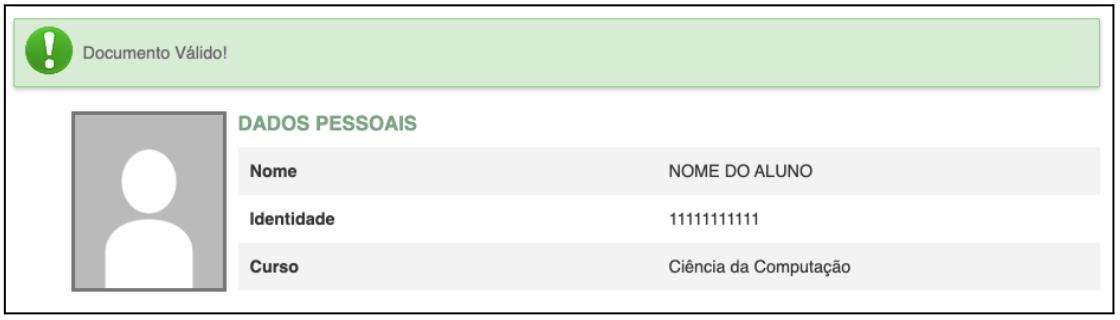
A interface de autenticação do histórico acadêmico, apresentada na Figura 15, é bem mais enxuta do que as interfaces de autenticação dos demais documentos autenticáveis. Por conta de seu caráter automatizado, ou seja, que não necessita do preenchimento manual dos dados presentes no documento, basta que o documento digital seja anexado ao formulário de autenticação para dar início ao processo de verificação.

Figura 15: Interface de autenticação do histórico acadêmico no Portal UFRJ



Ao ser anexado, o documento é submetido ao fluxo de autenticação, que atesta sua validade conforme previamente descrito. No caso de autenticidade atestada, é retornada uma mensagem de sucesso ao usuário junto de um bloco de informações a respeito do aluno portador do documento (Figura 16). Já para o caso de falha, apenas uma mensagem de erro é exibida abaixo do formulário.

Figura 16: Bloco resultante de uma autenticação bem sucedida



5 CONCLUSÃO

Em síntese, a partir das ferramentas matemáticas e computacionais formalmente descritas neste trabalho, foi possível desenvolver um serviço para o Sistema Integrado de Gestão Acadêmica da Universidade Federal do Rio de Janeiro capaz de emitir documentos assinados digitalmente e autenticá-los de maneira transparente ao usuário final. Dessa forma, disponibilizamos um meio de garantir que os históricos acadêmicos do corpo discente da UFRJ possam ser verificados tanto em relação a sua origem quanto no que se refere a integridade da informação vinculada a esse tipo de documento.

Atestar a origem do documento é fundamental se quisermos garantir que apenas o serviço de emissão de documentos seja capaz de gerar relatórios oficiais e válidos para a Universidade. Uma assinatura digital gerada por meio de chave privada garante esse controle. Além disso, é essencial validar a integridade do documento, se certificando de que seus dados não tenham sido alterados visando benefícios ilícitos e vantagens desmerecidas. A fim de se utilizar o histórico acadêmico para análise de desempenho acadêmico de alunos da Universidade, bem como para classificação e tomadas de decisão em concursos internos e externos de acesso ao ensino, é imprescindível que haja controle e segurança da informação sendo vinculada em nome da Instituição.

Apesar de solucionar o problema inicialmente posto, é interessante perceber que esta solução não atende a demandas em massa, uma vez que a validação do documento acontece de maneira sequencial e individual. Como evolução futura para essa aplicação, existe a oportunidade de desenvolver um serviço de autenticação para múltiplos arquivos, capaz de autenticar documentos em lotes. Assim, seria possível, por exemplo, validar todos os documentos cadastrados em um concurso interno de uma única vez, discriminando aqueles que fossem apontados como inválidos.

Além disso, existe ainda o desafio de estender esta solução para o plano analógico, contemplando documentos físicos no fluxo de autenticação. Para isso, seria necessário um passo anterior ao sistema, em que os registros físicos fossem traduzidos para o formato eletrônico, gerando um documento digital equivalente ao original. Será necessário, também, encontrar uma forma de transportar a assinatura de maneira explícita junto ao documento, uma vez que o formato analógico (ainda) não dispõe de propriedades virtuais ou metafísicas.

REFERÊNCIAS

SHARIF, A. GINTING, D. S. DIAS, A. D. "Securing the Integrity of PDF Files using RSA Digital Signature and SHA-3 Hash Function", **2021 International Conference on Data Science, Artificial Intelligence, and Business Analytics (DATABIA)**, Medan, Indonesia, p. 154-159, 2021. Disponível em: https://ieeexplore.ieee.org/document/9650121. Acesso em: 13 de março de 2023

PAAR, C. PELZL, J. "**Understanding Cryptography: A textbook for students and practitioners"**, Springer-Verlag Berlin Heidelberg, 2010.

LUCIANO, D. PRICHETT, G. **Cryptology: From Caesar Ciphers to Public Key Cryptosystems**. The College Mathematics Journal, 1987, volume 18, p.2-17. Disponível em https://www.math.stonybrook.edu/~moira/mat331-spr10/papers/1987%20LucianoCryptology%20From%20Caesar%20Ciphers%20to.pdf. Acesso em: 8 de outubro de 2023.

University of Nottingham (UoN), **Authentication of legal and administrative documents**, 2011. Reading and Understanding Medieval Documents. Disponível em https://www.nottingham.ac.uk/manuscriptsandspecialcollections/researchguidance/medievaldocuments/authentication.aspx. Acesso em: 7 de abril de 2023.

TIROTTI, J. M. TIROTTI, R. "**Manual Prático da Análise Grafotécnica**" São Paulo, 2021

RIVEST, Ronald L. "Cryptography". Em: LEEUWEN, J. V. **Handbook of Theoretical Computer Science, Volume A - Algorithms and Complexity**. ELSEVIER SCIENCE PUBLISHERS B.V., 1990. p.719- 750

UFRJ. **Fatos e Números**, 2022. A plataforma de acesso à informação da instituição. Disponível em: https://ufrj.br/acesso-a-informacao/institucional/fatos-e-numeros/. Acesso em: 7 de maio de 2023

CASTIGLIONE, A., SANTIS, A., SORIENTE, C., "Security and privacy issues in the Portable Document Format" **Journal of Systems and Software**, Volume 83, Issue 10, 2010,

p. 1813-1822. Disponível em https://www.sciencedirect.com/science/article/pii/S0164121210001287. Acesso em 14 de maio de 2023.

YAKSIC, V. O. C. "A study on hash functions for cryptography" **Global Information Assurance Certification Paper, SANS Institute**, 2003. Disponível em https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=ff346d00ccfa9b09ec54f002b656ef6ddb9c225f. Acesso em 20 de maio de 2023.

PRENEEL, B. "**Analysis and design of cryptographic hash functions**", 1993. Tese de Doutorado. Katholieke Universiteit te Leuven. Disponível em https://www.e-reading.club/bookreader.php/141503/Analysis_and_Design_of_Cryptographic_Hash_Functions.pdf. Acesso em 21 de maio de 2023

COUTINHO, S. C. "**Números Inteiros e Criptografia RSA**", 2ª edição. Rio de Janeiro, Brasil. IMPA, 2014.

OPENSSL. **Documentação da ferramenta**, 2023. Disponível em https://www.openssl.org/. Acesso em 16 de julho de 2023.

MLADENOV, V. MAINKA, C. "**1 Trillion Dollar Refund – How To Spoof PDF Signatures**", ACM SIGSAC Conference on Computer and Communications Security (CCS), Londres, 2019. Disponível em https://www.pdf-insecurity.org/download/paper-pdf-signatures-ccs2019.pdf. Acesso em 16 de agosto de 2023.

ORACLE. "**The *keytool* Command**". Documentação do utilitário. Disponível em: https://docs.oracle.com/javase/8/docs/technotes/tools/unix/keytool.html. Acesso em 18 de agosto de 2023.