INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE MINAS GERAIS - *CAMPUS* BETIM
BACHARELADO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Gabriel Keven Domingues de Souza

DESENVOLVIMENTO DE UM SOFTWARE PARA REALIZAR CRIPTOGRAFIA HÍBRIDA AES-RSA E TRANSMITIR ARQUIVOS ATRAVÉS DA WEB

Betim
2025

GABRIEL KEVEN DOMINGUES DE SOUZA

DESENVOLVIMENTO DE UM SOFTWARE PARA REALIZAR CRIPTOGRAFIA HÍBRIDA AES-RSA E TRANSMITIR ARQUIVOS ATRAVÉS DA WEB

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Orientador: Prof. Me. Mauricio Monteiro da Silva

Betim
2025

FICHA CATALOGRÁFICA

S729d Souza, Gabriel Keven Domingues de

Desenvolvimento de um *software* para realizar criptografia híbrida AES-RSA e transmitir arquivos através da *Web* / Gabriel Keven Domingues de Souza. – 2025.

52 f. : il.

Trabalho de conclusão de curso (Bacharelado em Engenharia de Controle e Automação) - Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais, Câmpus Betim, 2025.

Orientação: Prof. Me. Maurício Monteiro da Silva

1. Engenharia de software. 2. Criptografia híbrida. 3. Algoritmos de criptografia. 4. Engenharia de Controle e Automação. I. Souza, Gabriel Keven Domingues de. II. Título.

CDU: 004.41

Ficha catalográfica elaborada pelo Bibliotecário Denísio Pereira Marcos CRB-6/3142

Gabriel Keven Domingues de Souza

DESENVOLVIMENTO DE UM SOFTWARE PARA REALIZAR CRIPTOGRAFIA HÍBRIDA AES-RSA E TRANSMITIR ARQUIVOS ATRAVÉS DA WEB

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Engenharia de Controle e Automação do Instituto Federal de Educação, Ciência e Tecnologia de Minas Gerais *Campus* Betim, como parte dos requisitos para obtenção do título de Bacharel em Engenharia de Controle e Automação.

Aprovado em: 15/ 12 / 2025 pela banca examinadora:

Documento assinado digitalmente
gov.br
MAURICIO MONTEIRO DA SILVA
Data: 29/12/2025 11:23:36-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Mauricio Monteiro da Silva (Orientador) - IFMG

Documento assinado digitalmente
gov.br
VIRGIL DEL DUCA ALMEIDA
Data: 23/12/2025 21:25:48-0300
Verifique em https://validar.iti.gov.br

Prof. Me. Virgil Del Duca Almeida - IFMG

Documento assinado digitalmente
gov.br
FERNANDO CARDOSO DE SOUZA
Data: 15/12/2025 16:19:51-0300
Verifique em https://validar.iti.gov.br

Esp. Fernando Cardoso de Souza - IFMG

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus pela oportunidade de ter realizado esse curso em uma instituição federal. Agradeço também, a Nossa Senhora Aparecida, da qual sou devoto, que não me deixou desistir. Faço aqui também um eterno agradecimento aos meus pais e minha irmã por não desistirem de mim e confiarem que eu alcançaria esse diploma. Sem o apoio deles, tudo isso não seria possível.

Por fim, agradeço aos meus companheiros de curso, assim como eu os ajudei, eles também me ajudaram. Novamente, expresso aqui meu agradecimento a todos que diretamente ou indiretamente me auxiliaram a concluir essa graduação.

RESUMO

O compartilhamento de arquivos sigilosos em instituições públicas, como a Fundação de Ensino de Contagem (Funec), pode ocorrer somente por meios físicos para garantir a segurança. Porém, gera-se ineficiência logística. Este trabalho apresenta o desenvolvimento de um software web para a transmissão segura de arquivos utilizando criptografia híbrida (AES e RSA). O sistema foi desenvolvido em linguagem PHP com o framework CodeIgniter, banco de dados MySQL e JavaScript para execução das operações criptográficas no lado do cliente. A metodologia combinou a eficiência do algoritmo simétrico AES para a cifragem dos arquivos com a segurança do algoritmo assimétrico RSA para a troca das chaves de sessão. Os resultados obtidos, através de testes de tráfego de rede e validação de upload e download, comprovaram que os dados permanecem cifrados durante a transmissão e o armazenamento, sendo acessíveis apenas mediante a chave privada correta do destinatário. Conclui-se que o protótipo atende aos requisitos de confidencialidade e integridade, validando a substituição do transporte físico pela transmissão web segura.

Palavras-chave: Criptografia Híbrida; AES; RSA; Segurança Web; Engenharia de Software.

ABSTRACT

The sharing of confidential files in public institutions, such as the Contagem Education Foundation (Funec), can only occur through physical means to guarantee security. However, this generates logistical inefficiency. This work presents the development of a web software for the secure transmission of files using hybrid cryptography (AES and RSA). The system was developed in PHP with the CodeIgniter framework, MySQL database, and JavaScript for executing cryptographic operations on the client side. The methodology combined the efficiency of the symmetric AES algorithm for file encryption with the security of the asymmetric RSA algorithm for session key exchange. The results obtained, through network traffic tests and upload and download validation, proved that the data remains encrypted during transmission and storage, being accessible only with the correct private key of the recipient. It is concluded that the prototype meets the requirements of confidentiality and integrity, validating the replacement of physical transport with secure web transmission.

Keywords: Hybrid Encryption; AES; RSA; Web Security; Software Engineering.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

FUNEC	Fundação de Ensino de Contagem
NIST	National Institute of Standards and Technology
DES	Data Encryption Standard
3DES	Triple Data Encryption Standard
AES	Advanced Encryption Standard
IDEA	International Data Encryption Algorithm
RC2	Rivest Cipher 2
RSA	Rivest-Shamir-Adleman
IFMG	Instituto Federal de Minas Gerais
MDC	Máximo Divisor Comum
CSS	Cascading Style Sheets ou Folhas de Estilo em Cascata
HTML	HyperText Markup Language ou Linguagem de Marcação de Hipertexto
SSL	Security Sockets Layer

LISTA DE SÍMBOLOS

$\leq$	"menor ou igual a"
$\geq$	"maior ou igual a"
α	alfa
β	beta
$\mathbb{Z}$	conjunto dos números inteiros
$\mathbb{Z}_n$	conjunto dos inteiros módulo n
$\in$	"pertence a"
$\overline{a}$	classe de equivalência do elemento a
$:=$	“é definido como”
$\exists$	"existe"
$\forall$	“para todo”
$\sim$	relação de equivalência
$\equiv$	relação de congruência

SUMÁRIO

1 INTRODUÇÃO

Ao longo das últimas décadas o processo de compartilhamento de arquivos sofreu significativas mudanças desde os anos 40 até os dias atuais. Nesse intervalo de tempo, uma forma comum de compartilhamento de arquivos foi através das mídias removíveis (REZENDE, 2009).

Nesse contexto, a Fundação de ensino de Contagem (Funec) - uma autarquia do município de Contagem - tem como um dos seus objetivos a realização de concursos e processos seletivos para entidades do próprio município ou para outros órgãos públicos. Partindo disso, existe a etapa de criação de provas que envolve o recebimento das respostas e as perguntas que serão aplicadas no dia do certame. O recolhimento desses conteúdos elaborados pelos colaboradores externos, professores, se faz por meio de pessoas autorizadas pela Funec com auxílio de um pen drive (meio físico).

As informações coletadas através de um pen drive dificilmente são interceptadas, isto é, o conteúdo das avaliações está inacessível às pessoas não autorizadas a participar da etapa de confecção de provas, porque poucos envolvidos no processo tem acesso ao *pen drive*. Nessa análise, a etapa do transporte do conteúdo sigiloso apresenta certa segurança. Uma vez que, para tentar obter os dados secretos torna-se necessário primeiro acessar o pen drive. Entretanto, o tempo de deslocamento para obtenção das avaliações pode ser um empecilho na rotina de trabalho do setor responsável e dos próprios professores que desempenham outras tarefas. Além disso, há uma limitação geográfica para a contratação de elaboradores. Diante disso, há evidências suficientes para a comprovação de que a etapa do transporte das perguntas de prova na Funec se constitui como manual.

Com o crescimento no número de computadores ao redor do mundo, somado à expansão de redes de interconexão com largura de banda cada vez maiores e mais popularmente acessíveis, há um estimulo a utilização de sistemas que almejam a interconexão dos recursos presentes nestes computadores, independentemente de sua localização (REZENDE, 2009).

Nesse cenário, a partir do surgimento das redes de computadores, o compartilhamento de recursos tornou-se acessível a todos os usuários conectados na rede, sem dependência da localização física (TANENBAUM, 2003). Nesse sentido, o envio de arquivos pode ser feito através da Internet, principal rede de computadores do mundo. Entretanto, para evitar o vazamento de dados sigilosos se faz necessária a utilização de tecnologias específicas para garantir a integridade e segurança das informações. Dentre elas, destaca-se a criptografia.

A criptografia pode ser conceituada como o estudo e a aplicação de técnicas matemáticas para garantir que um dado se torne secreto (BRAGA; DAHAB, 2015). Desse modo, essa ferramenta pode ser aplicada no transporte de informações sigilosas para permitir segurança nas interações decorrentes da utilização da Internet.

Perante o exposto acima, o objetivo deste trabalho é o desenvolvimento de um sistema informatizado que seja capaz de transmitir essas informações sigilosas, bem como criptografar

esses dados transmitidos. Tudo isso, para atender a Funec e outras empresas ou órgãos públicos que almejam compartilhar arquivos sigilosos entre seus colaboradores ou com o público externo.

1.1 Justificativa

Segundo a nossa constituição Federal de 1988 no inciso XXXIII do artigo 5º:

> "...todos têm direito a receber dos órgãos públicos informações de seu interesse particular, ou de interesse coletivo ou geral, que serão prestadas no prazo da lei, sob pena de responsabilidade, ressalvadas aquelas cujo sigilo seja imprescindível à segurança da sociedade e do Estado;" (Brasil, 1988)

Com base nessa lei o conteúdo de uma prova é uma informação sigilosa e imprescindível à isonomia de processos seletivos que não pode ser divulgada a todos. Estendendo esse raciocínio, além dos órgãos públicos existem entidades privadas que possuem arquivos sigilosos essenciais à sua segurança jurídica e comercial. O sigilo dessas informações não impede a utilização da Internet como meio para compartilhamento desses dados para um público restrito. Desse modo, faz-se necessário o desenvolvimento de um sistema informatizado capaz de criptografar e transmitir esses arquivos entre usuários autorizados para manter as informações menos suscetíveis a ataques de segurança e vazamentos de dados.

1.2 Objetivos

1.2.1 *Objetivo geral*

Criar um sistema informatizado que permita a segurança no compartilhamento de dados através da Internet.

1.2.2 *Objetivos específicos*

Os objetivos específicos são:

- Desenvolver um *software* quer permita a criptografia dos dados;
- Aplicar a criptografia híbrida no processo de envio e recebimento de arquivos;
- Comprovar a eficiência da criptografia híbrida;
- Validar a transmissão de arquivos através da *web*.

1.3 Organização do Texto

O presente trabalho está estruturado em cinco capítulos. O capítulo 1 apresenta a introdução, contextualizando o problema do transporte físico de dados na Funec, a justificativa e os objetivos do projeto. O capítulo 2 aborda o referencial teórico, fundamentando os conceitos de criptografia simétrica e assimétrica, teoria dos números, algoritmos RSA e AES, além das tecnologias web utilizadas. O capítulo 3 descreve a metodologia, detalhando a arquitetura do software, o fluxo da criptografia híbrida e as ferramentas de desenvolvimento. O capítulo 4 expõe os resultados obtidos nos testes de validação de segurança, análise de tráfego e desempenho do sistema. Por fim, o capítulo 5 apresenta as conclusões finais, as limitações do protótipo e sugestões para trabalhos futuros.

2 REVISÃO BIBLIOGRÁFICA

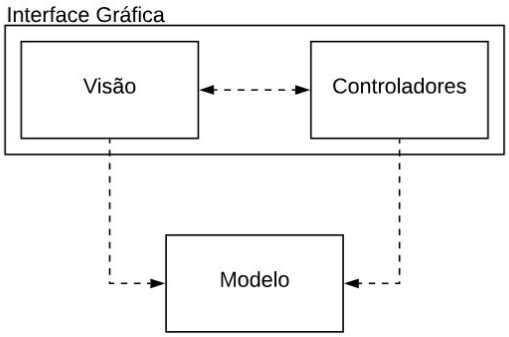
Nesse capítulo serão abordados conceitos aplicados ao presente trabalho sobre criptografia. Mas também, a diferenciação entre criptografia simétrica e assimétrica, cifra de fluxo e cifra de bloco, o algoritmo RSA, o algoritmo AES e as demais tecnologias necessárias para o desenvolvimento do protótipo. Todas essas informações são pertinentes para construção do conhecimento básico e essencial para o desenvolvimento do sistema informatizado. Além disso, esses conceitos serão úteis para a compreensão da metodologia deste trabalho e para avaliar e analisar os resultados finais.

2.1 Ambiente de Desenvolvimento

A arquitetura de software pode ser definida com o desenvolvimento em mais alto nível do sistema considerando sua macro divisão em componentes, camadas, serviços ou pacotes (VALENTE, 2022). Sistemas web são comumente desenvolvidos usando o padrão *Model-View-Controller* (MVC), que segundo Valente (VALENTE, 2022) tem como objetivo a resolução do problema da separação de apresentação e modelo em sistemas de interfaces gráficas. Esse modelo foi proposto no final da década de 1970 e propõe que as classes do sistema devem ser planejadas em três grupos (VALENTE, 2022). Além disso, a figura 1 representa de maneira visual a arquitetura MVC.

- *Visão*: Classe responsável pela representação gráfica do sistema.
- *Controlador*: Classe que realiza a interpretação e faz o tratamento das informações de entrada.
- *Modelo*: Classe com função de armazenar os dados manipulados pela aplicação.

Figura 1 – Arquitetura MVC.



Fonte: VALENTE, 2022.

2.2 Construção de Software Web

Para a implementação do sistema proposto, foi selecionado um conjunto de tecnologias que garantisse não apenas a funcionalidade da aplicação web, mas também a segurança criptográfica. Nesta seção, são detalhadas as linguagens de programação, frameworks, bibliotecas de criptografia e ferramentas de infraestrutura utilizadas no desenvolvimento do protótipo.

Linguagem de Programação PHP

Hypertext Preprocessor (PHP) é uma linguagem script open source utilizada para a construção de aplicações web. Além disso, o código desenvolvido em PHP é executado no servidor com a possibilidade de mesclar trechos seus próprios códigos com a linguagem de marcação de hipertexto, o HTML (The PHP Group, 2025).

Codeigniter

É um framework web full-stack PHP. Essa ferramenta permite a criação otimizada de projetos simples até aplicações mais avançadas. Fornece também um conjunto de bibliotecas utéis para o desenvolvimento de tarefas relativas à finalidade do software (CodeIgniter Foundation, 2025).

MySQL

MySQL é um banco de dados de código aberto. Devido ao seu desempenho e confiabilidade se tornou um dos bancos mais populares em aplicações Web (Oracle Corporation, 2025).

Bootstrap

Trata-se de um *framework* front-end de código aberto (Bootstrap Core Team, 2025) destinado à estilizar às páginas que serão acessadas pelos usuários da aplicação.

JSencrypt

Trata-se de uma biblioteca em javaScript para executar criptografia OpenSSL RSA síncrona e assíncrona. descriptografia e geração de chaves no navegador. Também, é compatível com o ambiente de Node js (TIDWELL, 2025).

Web Crypto API

Conceitua-se como uma interface que permite utilizar recursos criptográficos primitivos para desenvolver sistemas usando criptografia (Mozilla Developer Network, 2025).

Linguagem de marcação de Hipertexto HTML

Trata-se de um bloco que permite marcar os conteúdos a serem exibidos em um navegador web. O Hipertexto faz referência a links que permitem a navegação entre diferentes páginas na Web (MDN Web Docs, 2025a).

Folha de Estilo em Cascata CSS

É um linguagem que tem a função de estilizar e arranjar as páginas web. Alterando aspectos como cor, fonte, tamanho e espaçamento do conteúdo definido nas tags em HTML. Mas também, essa linguagem permite ao programador definir regras a determinados elementos na página na web (MDN Web Docs, 2025c).

Linguagem de programação *javascript*

Javascript defini-se como uma linguagem baseada em objetos com funções de primeira classe. Em outras palavras, trata-se de uma linguagem para criar scripts para as páginas web. Ademais, conceitua-se como multi-paradigma e dinâmica que suporta a orientação a objetos e os estilos imperativo e funcional (MDN Web Docs, 2025b).

Docker

Docker é um software que permite a construção de ambientes em contêineres compartilhando e executando microserviços. Adicional a isso, permite a integração e desenvolvimento entre diferentes linguagens de de programação como PHP e javascript (Docker Inc., 2025).

WireShark

Configura-se como um analisador de pacotes de rede, que permite "capturar"informações transmitidas e recebidas dentro de uma rede. Com ele, possibilita-se a monitoração do que está sendo transmitido e recebido em aplicações web (The Wireshark team, 2025).

Nginx Proxy Manager

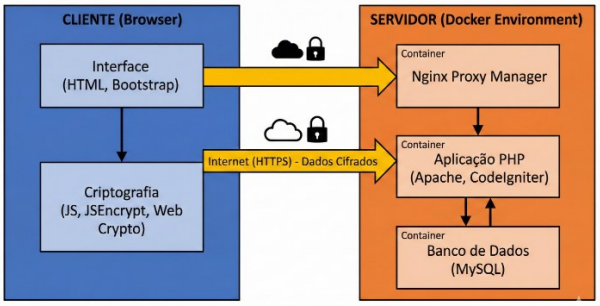
Trata-se de um gerenciador de proxy de código aberto e gratuito com interface gráfica para gerenciar hosts proxy com certificados SSL Let's Encrypt integrados (docker-compose.de, 2025).

A arquitetura do sistema foi projetada para integrar todas essas ferramentas. Nesse sentido, o fluxo inicia-se no Frontend, onde HTML, CSS e Bootstrap constroem a interface visual. Nesse ambiente (lado do cliente), as bibliotecas JSEncrypt e Web Crypto API atuam em conjunto com o JavaScript para executar a criptografia dos dados antes mesmo do envio.

A transmissão ocorre via protocolos seguros gerenciados pelo Nginx Proxy Manager, que encaminha as requisições para o ambiente de Backend. Toda a infraestrutura do servidor é encapsulada em contêineres Docker, garantindo que o servidor de aplicação (PHP com CodeIgniter) e o banco de dados (MySQL) rodem de forma isolada e padronizada. Por fim, ferramentas como o Wireshark foram fundamentais para validar a integridade dessa integração, monitorando o tráfego de rede entre as pontas.

A figura 2 demonstra de forma detalhada onde cada ferramenta foi utilizada no desenvolvimento da aplicação.

Figura 2 – Fluxograma das ferramentas utilizadas no desenvolvimento do protótipo.



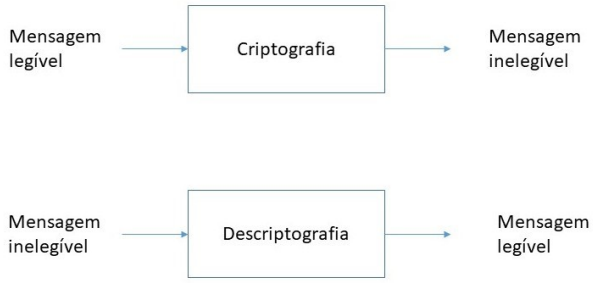
Fonte: Elaborado pelo autor, 2025.

2.3 Criptografia

A criptografia é a ciência destinada ao estudo de técnicas matemáticas para garantir que um dado se torne secreto. A criptografia oferece quatro serviços: confidencialidade(manter as informações secretas), autenticação (validar identidade), integridade (certificar que a informação não foi modificada) e irrefutabilidade (garantir que o autor da mensagem não possa negar sua autoria) (BRAGA; DAHAB, 2015).

Algoritmos que realizam criptografia de informações têm como meta "esconder"dados sigilosos (TERADA, 2008). O processo inverso para "revelar"as informações sigilosas é denominado descriptografia. A Figura 3 resume e ilustra o conceito de criptografia.

Figura 3 – Conceito de criptografia.



Fonte: Elaborado pelo autor, 2025.

Na sua origem a criptografia tinha a função principal de esconder os segredos militares de um Estado. Com avanço dos anos e a modernização dos meios de comunicação, há um diversidade de dados eletrônicos que são manipulados cotidianamente pela população. Nesse aspecto, a criptografia tornou-se uma ferramenta tecnológica indispensável à segurança das pessoas (AMARAL *et al.*, 2021).

2.3.1 Encriptação simétrica e Encriptação assimétrica

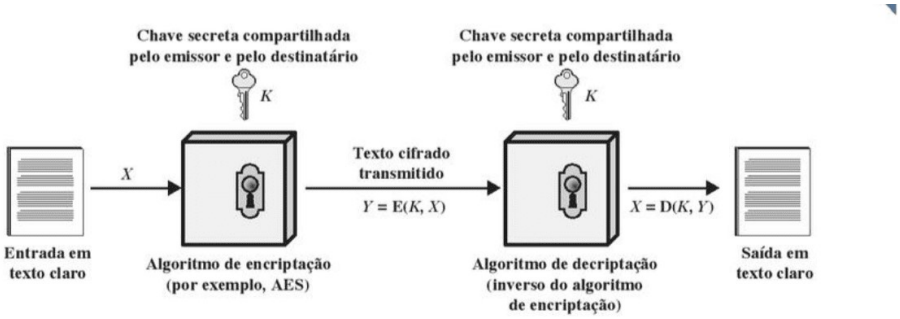
A encriptação simétrica ou encriptação convencional ou ainda a criptografia de chave única consiste na utilização de uma chave idêntica tanto para o processo de criptografia quanto descriptografia (ABOOD; GUIRGUIS, 2018). Por sua vez, a chave assimétrica ou criptografia de chave pública utiliza um "par de chaves". Em que uma chave pública tem a função de criptografar a mensagem e a outra chave privada tem a finalidade de decodificar a informação. (AMARAL *et al.*, 2021).

Cabe ainda mencionar nessa significação de criptografia, outros conceitos inerentes a esse assunto(STALLINGS, 2014).

- Texto claro: mensagem original;
- Texto cifrado: mensagem codificada;
- Encriptação: processo de transformar um texto claro em texto cifrado;
- Descriptação: processo de converter um texto cifrado em texto claro.

Para aplicação de cifragem simétrica é preciso um bom algoritmo de encriptação. Além disso, tanto o emissor quanto o receptor precisam manter as chaves em locais seguros. (STALLINGS, 2014). A figura 4 ilustra de forma simplificada a encriptação simétrica.

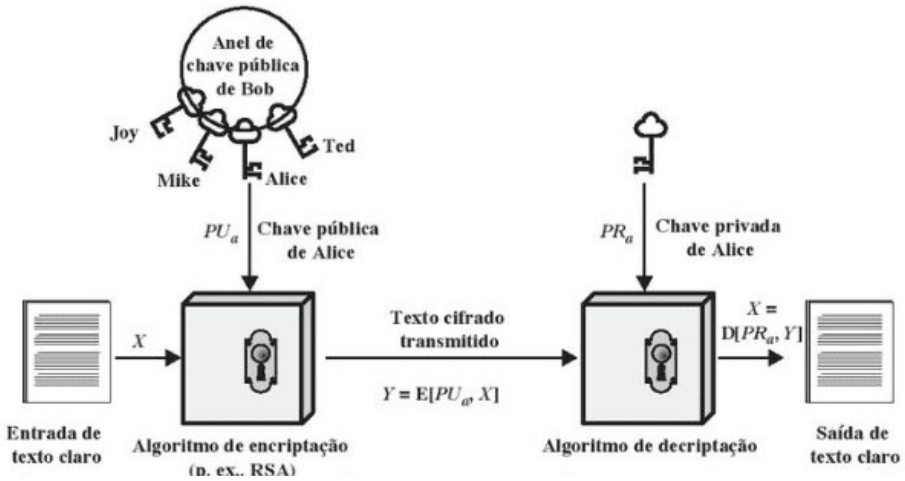
Figura 4 – Modelo de cifragem simétrica.



Fonte: STALLINGS, 2014.

Na criptografia assimétrica, a chave pública fica disponível para qualquer pessoa que necessite realizar uma comunicação efetiva de forma segura. Entretanto a chave privada deve estar acessível apenas para o seu respectivo titular. (OLIVEIRA, 2012). A figura 5 esclarece a encriptação assimétrica.

Figura 5 – Modelo de cifragem assimétrica.



Fonte: STALLINGS, 2014.

Nesse contexto de diferenciação das técnicas de criptografia, o procedimento de utilizar a chave simétrica pode ser mais simples e ágil. Tendo em vista que, se a chave for consideravelmente complexa, o desenvolvimento do algoritmo de cifragem torna-se mais simples. E por consequência disso, ele se torna mais rápido. (OLIVEIRA, 2012). Por sua vez, o processo de cifragem assimétrica permite a identificação do emissor da mensagem. Isso possibilita a aplicação da assinatura digital. Que consiste em uma informação adicionada a mensagem a fim de promover a autenticidade, integridade e não repudiação (COSTA; FIGUEIREDO, 2010). Entretanto, esse mesmo método possui certa complexidade na criação de algoritmos para que possam reconhecer as duas chaves(pública e privada) e relaciona-las.(OLIVEIRA, 2012). A tabela 1 traz um resumo das diferças entre os métodos simétricos e assimétricos.

Tabela 1 – Diferenciação dos métodos criptográficos simétricos e assimétricos.

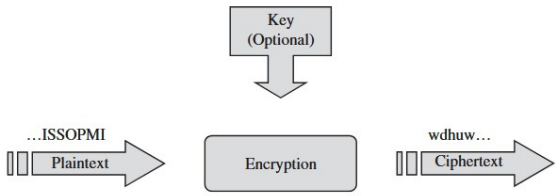
Criptografia simétrica ou chave privada	**Criptografia assimétrica ou chave pública**
Rápida	Lenta
Gerência e distribuição das chaves é complexa	Gerência e distribuição das chaves é simples
Não oferece assinatura digital	Oferece assinatura digital

Fonte: OLIVEIRA, 2012.

2.3.2 Cifra de fluxo e Cifra de bloco

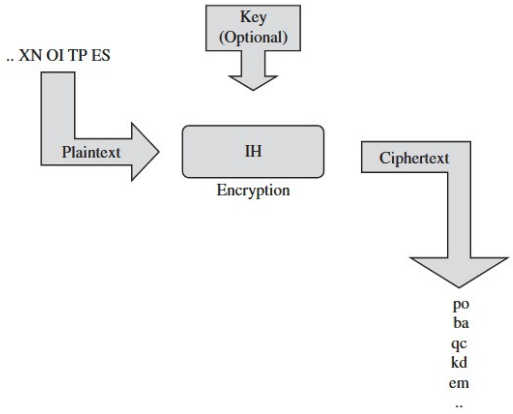
A cifra de fluxo converte caractere por caractere da mensagem clara em texto cifrado. Em outras palavras, a criptografia se faz símbolo por símbolo. Em oposição a isso, a cifra de bloco divide a mensagem em tamanhos fixos(blocos), e cada parte dividida passa pelo processo de criptografia como se fosse uma unidade. (SERAFIM, 2012). A figura 6 demonstra como é o procedimento da cifra por fluxo e a figura 7 detalha a cifra por bloco.

Figura 6 – Cifra de fluxo.



Fonte: PFLEEGER; PFLEEGER; COLES-KEMP, 2023.

Figura 7 – Cifra de bloco.



Fonte: PFLEEGER; PFLEEGER; COLES-KEMP, 2023.

2.3.3 Principais algoritmos de criptografia simétrica e assimétrica

Em primeiro lugar, cabe conceituar o termo algoritmo. Um algoritmo é um conjunto de instruções definidas que quando executadas impactam na resolução de um problema (CASTILHO; SILVA; WEINGAERTNER, 2020). Como por exemplo: uma receita de bolo. A execução das etapas da receita conforme as intruções implica na solução de um problema, que nesse contexto, refere-se a produção de um bolo.

Após o conceito acima, evidencia-se que os algoritmos assimétricos e simétricos buscam garantir a segurança das informações transmitidas. Nesse cenário, a tabela 2 detalha os principais algoritmos de criptografia simétrica. Em contrapartida, a tabela 3 demonstra os principais algoritmos de chave assimétrica (OLIVEIRA, 2012).

Embora os algoritmos apresentados ofereçam soluções robustas, cada abordagem isolada apresenta limitações para o cenário de transmissão web: os simétricos sofrem com o risco no compartilhamento da chave entre os usuários. Em oposição a isso, os assimétricos possuem custo computacional maior para grandes volumes de dados. A solução ideal emerge da combinação de ambos: a criptografia híbrida.

Contudo, para compreender a segurança por trás da troca de chaves do RSA (etapa crítica

desse sistema híbrido), faz-se necessário aprofundar nos fundamentos matemáticos que tornam esse algoritmo irreversível sem a chave privada.

Tabela 2 – Principais algoritmos simétricos.

Algoritmo	**Bits**	**Descrição**
DES	56	Pode permitir cerca de 72 quadrilhões de combinações, seu tamanho de chave (56 bits) é considerado pequeno, tendo sido quebrado por "força bruta"em 1997 em um desafio lançado na Internet. O NIST que lançou o desafio mencionado, recertificou o DES pela última vez em 1993, passando então a recomendar o 3DES.
3DES	112 ou 168	O 3DES é uma simples variação do DES, utilizando três ciframentos sucessivos. É seguro, porém muito lento para ser um algoritmo padrão.
AES	128, 192 ou 256	É um dos algoritmos mais populares, desde 2006, usado para criptografia de chave simétrica com cifra de bloco, sendo considerado como o padrão substituto do DES. O AES tem um tamanho de bloco fixo em 128 bits e uma chave com tamanho de 128, 192 ou 256 bits, ele é rápido tanto em software quanto em hardware, é relativamente fácil de executar e requer pouca memória.
IDEA	128	Segue as mesmas estruturas do DES. Além disso, esse algoritmo é muito utilizado no mercado financeiro. E por fim, uma implementação por software do IDEA é mais rápida do que uma implementação por software do DES.
Twofish	128	O Twofish é uma chave simétrica que emprega a cifra de bloco de 128 bits, utilizando chaves de tamanhos variáveis, podendo ser de 128, 192 ou 256 bits. Esse algoritmo, realiza 16 interações durante a criptografia, sendo um algoritmo bastante veloz.
Blowfish	32 a 448	Algoritmo desenvolvido por Bruce Schneier, que oferece a escolha entre maior segurança ou desempenho através de chaves de tamanho variável. Houve um aperfeiçoamento do Twofish.
RC2	8 a 1024	Utiliza chave de tamanho variável e utiliza o protocolo S/MIME, destinado à criptografia de emails corporativos.
CAST	40 a 128	Algoritmo que utiliza a cifra de bloco com tamanho de 64 bits, realiza entre 12 a 16 iterações na etapa principal. Em que, 16 rounds quando a chave tem mais de 80 bits. Também apresenta uma chave variável de 40 a 128 bits com acréscimo de 8 bits.

Fonte: Adaptado de OLIVEIRA, 2012.

2.4 Teoria dos números

A segurança dos algoritmos assimétricos, como o RSA, reside na complexidade na dificuldade computacional de resolver determinados problemas matemáticos. Para compreender profundamente como são geradas as chaves assimétricas e por que é tão difícil quebrá-las sem a chave privada, faz-se necessário explorar a base matemática que sustenta essa tecnologia. A seção a seguir dedica-se à Teoria dos Números, campo da matemática que fornece os fundamentos, como

Tabela 3 – Principais algoritmos assimétricos.

Algoritmo	**Descrição**
RSA	Algoritmo de criptografia assimétrica mais amplamente utilizado. A premissa por trás do RSA consiste na facilidade de multiplicar dois números primos para obter um terceiro número, mas muito difícil de recuperar os dois números primos a partir desse terceiro. Isso é denominado fatoração. Logo, derivar a chave privada a partir da chave pública envolve fatorar um grande número. Se o número for grande o suficiente e bem escolhido, então ninguém pode fazer isto em uma quantidade de tempo razoável.
ElGamal	O algoritmo envolve a manipulação matemática de grandes quantidades numéricas. Sua segurança advém de algo denominado problema do logaritmo discreto. Assim, o ElGamal obtém sua segurança da dificuldade de calcular logaritmos discretos em um corpo finito, o que lembra bastante o problema da fatoração.
Curvas Elípticas	Os sistemas criptográficos de curvas elípticas consistem em modificações de outros sistemas (o ElGamal, por exemplo), que passam a trabalhar no domínio das curvas elípticas, em vez de trabalharem no domínio dos corpos finitos. Além disso, tem o potencial de promover mais segurança com uma chave menor.

Fonte: Adaptado de OLIVEIRA, 2012.

a fatoração de inteiros e a aritmética modular, essenciais para o funcionamento da criptografia moderna.

2.4.1 Introdução

O estudo dos números foi mencionado nas civilizações antigas com destaque para a Grécia. Nesse campo de estudo, os gregos abordaram o cálculo necessário para obtenção do máximo divisor comum entre dois números. Mas também, a existência de uma infinidade de números primos. E, por último, a determinação dos números primos menores que um inteiro dado (COUTINHO, 2009). Com base nisso, pode-se conceituar a teoria dos números como a ciência que busca entender as propriedades e as relações entre os números. (CAVALCANTE, 2005).

2.4.2 Algoritmo da Divisão e o Teorema da divisão

O algoritmo da divisão tem como entrada dois número inteiros, isto é são números que não possuem a parte decimal, positivos a e b. E como saída, novamente dois número inteiros q e r. A equação abaixo demonstra a relação entre entrada e saída desse algoritmo (COUTINHO, 2009).

$$a = bq + r \quad \text{e} \quad 0 \leq r < b$$

As etapas para execução do algoritmo são as seguintes:

1. Inicie $Q = 0$ e $R = a$.
2. Se $R < b$, escreva: o quociente é Q e o resto é R, e então pare. Caso contrário, prossiga para a terceira etapa.
3. Se $R \geq b$, subtraia b de R, incremente 1 ao Q da etapa 1 e volte para a etapa 2.

Para aplicar esse algoritmo, torna-se necessário a criação de um laço. Em outras palavras, uma repetição de processos até encontrar o resultado que atenda os critérios da segunda etapa descrita acima. Cabe mencionar que, o quociente e resto da divisão serão únicos. Uma vez que, aplicação desse algoritmo por diferentes pessoas resulta nos mesmo valores. (COUTINHO, 2009) Com base no exposto, o algoritmo da divisão permite a conclusão de dois aspectos:

- O quociente e o resto da divisão sempre existem.
- O quociente e o resto são únicos.

2.4.3 Algoritmo euclidiano e Máximo Divisor Comum (MDC)

O MDC de um número pode ser definido como o máximo divisor comum entre dois números inteiros. Isso signfica obter um número capaz de ser divisor tanto do primeiro quanto do segundo número que são inteiros (COUTINHO, 2009).

Se d é MDC dos números inteiros a e b, então

$$d = \text{mdc}(a,b)$$

Além disso, se $\text{mdc}(a,b) = 1$ os números são co-primos(primos entre si).

Basicamente, para encontrar o MDC de um número se faz a divisão de a e posteriormente de b. Com a divisão, obtem-se os divisores de a e b. Em seguida, torna-se necessário obter a intercessão, isto é o maior divisor em comum dos dois números.

Ainda nesse contexto do cálculo do MDC, existe também o algoritmo euclidano estendido. Esse algoritmo é essencial para o desenvolvimento do método de criptografia RSA (COUTINHO, 2009). Considerando novamente dois números inteiros a e b e d o máximo divisor comum desse números. Existem dois números inteiros α e β de modo que:

$$\alpha \cdot a + \beta \cdot b = d$$

Além disso, com há inserção de um inteiro k na equação acima, obtem-se a equação abaixo:

$$(\alpha + kb) \cdot a + (\beta - ka) \cdot b = d$$

Através da equação acima, comprova-se que existe uma inifinidade de pares inteiros que satisfazem a equação. Ademais, ou α ou β será um inteiro negativo (COUTINHO, 2009).

2.4.4 Números Primos

Um número inteiro p é primo se $p \neq \pm 1$ e os únicos divisores de p são ± 1 e $\pm p$. Resumidamente, um número primo se define como um número diferente de $+1$ ou -1. Mas, que tem como divisores os números $+1$, -1, $+p$ e $-p$ (COUTINHO, 2009).

2.4.5 Teorema da Fatoração única

O teorema traz à tona duas afirmações (COUTINHO, 2009):

- Todo inteiro pode ser escrito como produto de potências de primos
- Só é possível uma combinção possível de primos e expoentes para fatoração de um determinado inteiro.

2.4.6 Aritmética Modular (Congruências)

Conhecida também como aritmética dos fenômenos cíclicos, define que que dois números inteiros a e b são congruentes módulo n se $a - b$ é um múltiplo de n (COUTINHO, 2009). Logo,

$$a \equiv b \pmod{n}$$

Em outras palavras, dividindo a e b por n separadamente será obtido o mesmo resto da divisão. Também cabe mencionar que da divisão de a por n, o resultado do resto será b.

2.4.7 Conjunto dos números inteiros módulo n

Seja $\mathbb{Z}$ o conjunto dos números inteiros e considerando a relação de congruência módulo n. Os elementos determinados por essa relação constituem o conjunto $\mathbb{Z}_n$, denominado *conjunto dos inteiros módulo n* (COUTINHO, 2009).

2.4.8 Pequeno Teorema de Fermat

Esse teorema afirma que, para um número primo p e um número inteiro qualquer a, então p divide $a^p - a$ (COUTINHO, 2009), ou seja,

$$a^p \equiv a \pmod{p}.$$

Há também uma versão alternativa desse teorema que enuncia que, para um número primo p e um número inteiro a que não é divisível por p, vale que

$$a^{p-1} \equiv 1 \pmod{p}.$$

2.4.9 Função Totiente de Euler

A função totiente de Euler, denotada por $\varphi(n)$, representa a quantidade de inteiros positivos menores ou iguais a n que são coprimos com n. Definida por Leonhard Euler como uma generalização do Pequeno Teorema de Fermat (ESPíNDOLA; SILVA; SANTOS, 2021).

Essa função fundamenta o Teorema de Euler, que afirma que, se a e n são inteiros positivos tais que $mdc(a,n) = 1$, então:

$$a^{\varphi(n)} \equiv 1 \pmod{n}.$$

A exposição dos fundamentos da Teoria dos Números compõe a base funcional da criptografia assimétrica. Partindo disso, a interconexão entre esses conceitos é vital para compreensão do algoritmo RSA.

Assim, cabe destacar que Teorema da Fatoração Única garante a geração de chaves robustas baseadas em grandes números, enquanto o Algoritmo de Euclides e a Aritmética Modular viabilizam o cálculo eficiente dos componentes dessas chaves. Por fim, o Pequeno Teorema de Fermat e a Função Totiente de Euler fornecem a propriedade matemática essencial que permite cifrar e decifrar mensagens de forma segura e reversível apenas para o detentor da chave privada.

2.5 Algoritmo RSA

2.5.1 Origem

O algoritmo RSA foi inventado em 1978 por Ron Rivest, Adi Shamir e Leonard Adleman. É considerado um dos principais algoritmos assimétricos para assinaturas digitais ou criptografia de banco de dados (ABOOD; GUIRGUIS, 2018). Além disso, seus resultados são provenientes da teoria dos números. Uma vez que ele se baseia na dificuldade de determinar fatores primos de um alvo. Logo, a dificuldade na quebra das chaves, está na fatoração extramente difícil. (PFLEEGER; PFLEEGER; COLES-KEMP, 2023). Um destaque para sua aplicação está no mercado financeiro, na qual as informações precisar ser seguras e assinadas eletronicamente (MILANOV, 2009).

2.5.2 Funcionamento do algoritmo

Para aplicação desse processo de criptografia devem ser feitas as seguintes etapas (CAVALCANTE, 2005):

- Seleção de dois números primos extensos p e q;
- Realização dos cálculos $n = p \cdot q$ e $\phi(n) = (p-1)(q-1)$
- Faz-se a escolha um número d que seja co-primo a $\phi(n)$. Para isso, o $mdc(d, \phi(n)) = 1$
- Calcula-se e: $e \cdot d \equiv 1(mod(\phi(n))$
- O texto simples sofre divisão em blocos para que cada mensagem M fique dentro do seguinte intervalo $0 \leq M < n$

Cabe ressaltar que e é o expoente público utilizado para criptografar a mensagem. Por sua vez, d é o expoente privado utilizado na descriptografia.

Para cifrar a mensagem utiliza-se a a seguinte fórmula:

$$C \equiv M^e (\mod n)$$

Para descifrar a mensagem utiliza-se a seguinte fórmula:

$$M \equiv C^d (\mod n)$$

Diante desse cenário, para cifragem das informações os valores de e e n são primordiais. Portanto, a chave pública consiste no par (e, n). Para o processo inverso, são necessários d e n. Logo, a chave privada é formada pelo par (d, n) (CAVALCANTE, 2005).

2.6 Algoritmo AES

2.6.1 Origem

O AES (Advanced Encryption Standard) surgiu para substituir o algoritmo padrão anterior, o DES (Data Encryption Standard). O DES apresentou limitação em relação ao tamanho da sua chave (56 bits) (SOUZA; OLIVEIRA; CUNHA, 2025).

2.6.2 Funcionamento do algoritmo

O AES é um algoritmo de cifra de bloco simétrico. Dessa maneira, ele opera em blocos de dados de tamanho fixo de 128 bits (SOUZA; OLIVEIRA; CUNHA, 2025). Esses dados são

organizados em uma matriz 4x4 de bytes, chamada de "estado"(state) (SOUZA; OLIVEIRA; CUNHA, 2025).

O funcionamento se baseia na execução de múltiplas "rodadas"(rounds) de transformações sobre esse estado. O número de rodadas varia de acordo com o tamanho da chave utilizada, conforme abaixo (SOUZA; OLIVEIRA; CUNHA, 2025):

Chave de 128 bits: 10 rodadas

Chave de 192 bits: 12 rodadas

Chave de 256 bits: 14 rodadas

Cada rodada (exceto a última) é composta por quatro etapas (transformações) aplicadas sequencialmente:

- SubBytes (Substituição de Bytes): É a única etapa não linear do processo. Cada byte do "estado"é substituído por outro, com base em uma tabela de consulta fixa chamada S-box (caixa de substituição).
- ShiftRows (Deslocamento de Linhas): Esta é uma etapa de permutação. Os bytes em cada linha da matriz "estado"são rotacionados (deslocados) para a esquerda. A primeira linha não sofre rotação; a segunda é rotacionada em 1 byte; a terceira em 2 bytes; e a quarta em 3 bytes.
- MixColumns (Mistura de Colunas): Cada coluna da matriz "estado"é operada de forma independente. Esta etapa realiza uma multiplicação de matrizes sobre um corpo finito $(GF(2^8))$ para "misturar"os bytes dentro de cada coluna, garantindo difusão.
- AddRoundKey (Adicionar Chave da Rodada): O "estado"é combinado com uma "chave de rodada"(Round Key) através de uma operação simples de XOR bit a bit. Esta chave de rodada é derivada da chave de cifragem principal.

Cabe ressaltar que esse ciclo de etapas se faz pelo número de rodadas necessárias. Na última rodada, porém, a etapa MixColumns é omitida (SOUZA; OLIVEIRA; CUNHA, 2025).

2.7 Criptografia híbrida

2.7.1 Motivação

O AES utiliza uma única chave secreta compartilhada tanto para o processo de cifragem quanto para o de decifragem. Além disso, sua principal vantagem está na alta velocidade e eficiência computacional, tornando-a ideal para cifrar grandes volumes de dados. No entanto, sua grande desvantagem é o compartilhamento de chaves. Em outras palavras, garantir que o

transmissor e o receptor compartilhem a mesma chave secreta de forma segura, sem que ela seja interceptada.

O RSA, por sua vez, resolve o problema da distribuição de chaves ao utilizar um par de chaves: uma chave pública (que pode ser distribuída livremente) e uma chave privada (mantida em segredo pelo proprietário). Porém, o RSA é computacionalmente intensivo e significativamente mais lento que o AES, tornando-o impraticável para a cifragem de grandes mensagens.

A criptografia híbrida surge nesse contexto como uma abordagem que combina os pontos fortes de ambas as técnicas para prover uma solução que é ao mesmo tempo segura e eficiente (SUNEETHA; ANURADHA; NARASIMHAM, 2023).

2.7.2 Conceito

De acordo com Suneetha et al. (2023), o funcionamento da criptografia híbrida RSA-AES segue um fluxograma conforme abaixo:

1. **Cifragem dos dados:** A mensagem principal é cifrada utilizando um algoritmo simétrico (AES), com uma chave de sessão gerada aleatoriamente.
2. **Cifragem da chave:** A chave de sessão do AES, posteriormente é cifrada utilizando o RSA.
3. **Transmissão:** O remetente envia a mensagem cifrada com AES e a chave de sessão cifrada com RSA para o destinatário.
4. **Decifragem:** O destinatário usa sua chave privada RSA para decifrar a chave de sessão AES. Em seguida, usa essa chave de sessão AES para decifrar a mensagem principal.

Diante disso, a criptografia híbrida utiliza o RSA apenas para a tarefa segura de troca de chaves e o AES para cifragem dos dados. Permitindo tanto a segurança na transmissão da chave quanto o desempenho no processamento da mensagem (SUNEETHA; ANURADHA; NARASIMHAM, 2023).

3 METODOLOGIA

O Capítulo 2 apresentou a revisão bibliográfica de conceitos pertinentes para a construção do sistema informatizado. Neste capítulo por sua vez, faz-se a abordagem metodológica do trabalho. Nesse sentido, apresentam-se as ferramentas e os procedimentos realizados para desenvolvimento do software.

3.0.1 Definição do fluxograma do software

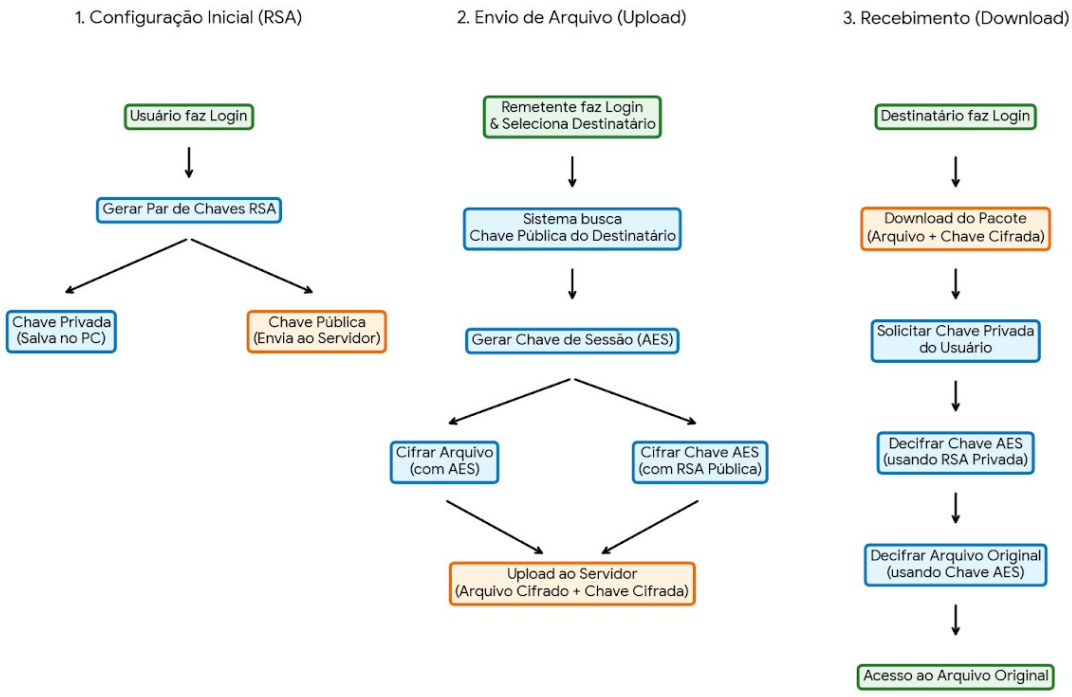
O funcionamento do sistema pode ser dividido nas seguintes etapas: login, geração de chaves, envio de arquivos e recebimento de arquivos. Abaixo, encontra-se uma visão dessas etapas a partir da visão do usuário.

1. **Login e geração de chaves (RSA)**
    a) O usuário acessa o sistema e realiza o login.
    b) Após o login, o sistema oferece a opção de gerar um par de chaves RSA.
    c) A chave privada RSA é baixada automaticamente para o computador do usuário.
    d) A chave pública RSA é enviada ao servidor e vinculada ao perfil do usuário.
2. **Envio de arquivo**
    a) O remetente faz login no sistema.
    b) Seleciona o destinatário desejado.
    c) O sistema recupera a chave pública RSA do destinatário.
    d) O remetente seleciona o arquivo a ser transmitido.
    e) O sistema gera localmente uma chave de sessão simétrica aleatória.
    f) O arquivo original é cifrado com a chave de sessão AES.
    g) A chave de sessão AES é cifrada com a chave pública RSA do destinatário.
    h) O arquivo cifrado e a chave de sessão cifrada são enviados ao servidor.
3. **Recebimento do arquivo**
    a) O destinatário acessa o sistema e realiza o login.
    b) Visualiza a lista de arquivos recebidos e escolhe um para download.
    c) O sistema baixa o arquivo principal e a chave de sessão cifrada.
    d) O navegador solicita que o destinatário informe sua chave privada RSA.
    e) A chave privada RSA é utilizada para decifrar a chave de sessão AES.
    f) Com a chave AES decifrada, o sistema decifra o arquivo principal.

g) O destinatário tem acesso ao arquivo original.

A figura 8 detalha de forma visual o fluxograma de todo o processo desde o login até o download do arquivo descriptografado.

Figura 8 – Fluxograma do processo.

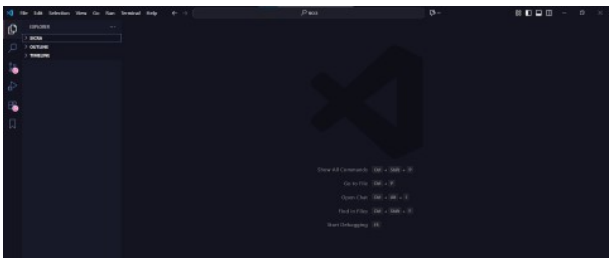


Fonte: Elaborado pelo autor, 2025.

3.0.2 Vscode

Para desenvolvimento do software, utilizou-se o visual studio Code. Essa ferramenta consiste em um editor de código-fonte gratuito. Além disso, ele permite a construção de aplicações modernas em java, javascript, node js entre outras tecnologias que se popularizam nos últimos anos no campo de desenvolvimento software. Para a construção desse mínimo produto viável (MVP), utilizou-se a linguagem PHP com o framework codeigniter em conjunto com javascript. A figura 9 apresenta o vscode.

Figura 9 – Exemplo de tela do Vscode.



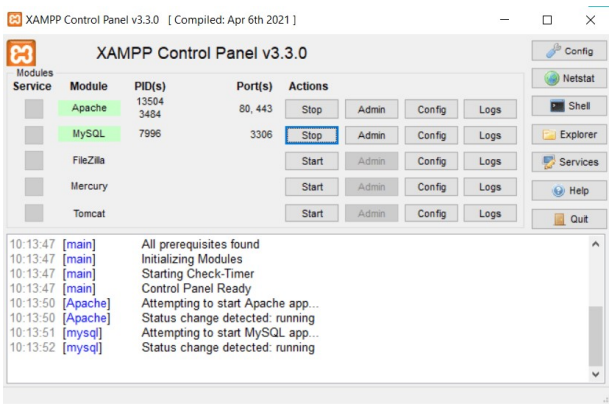
Fonte: Elaborado pelo autor, 2025.

3.0.3 XAMPP

Em virtude do ambiente de testes e desenvolvimento, optou-se pela instalação do XAMPP. O XAMPP é uma aplicação instalável que permite a simulação de um servidor web integrado a um banco dados. Dentro do ambiente de trabalho da Funec, utiliza-se esse servidor para executar as aplicações locais já existentes. Para acessar o servidor, utiliza-se o endereço IP *loopback* ou *localhost* 127.0.0.0.

A figura 10 apresenta a tela do XAMPP após a instalação. Além disso, é possível configurar o servidor web e o banco de dados como serviços do windows para que sejam executados sempre ao ligar o computador.

Figura 10 – Tela inicial do XAMPP.



Fonte: Elaborado pelo autor, 2025.

3.0.4 Construção das views e controllers

Para a construção das telas a serem exibidas pelos usuários optou-se pelo bootstrap. Em resumo, trata-se de um framework front-end, de código aberto que utiliza HTML, CSS e javascript para desenvolver de interfaces de sites e aplicativos web. Ademais, a sua utilização consiste em alterar e incrementar os atributos de tags do HTML de forma atribuir um padrão de estilização.

O controller é o arquivo desenvolvido em PHP responsável por receber as informações enviadas pelo cliente através do javascript e envia-las até o banco de dados e salvar o conteúdo cifrado no servidor.

No desenvolvimento do projeto foram criados controllers para diferentes divisões do sistema. Como por exemplo, um controller para registro dos usuários e outro para login. Ademais, cabe ressaltar que a programação do PHP seguiu o paradigma de orientação objetos (POO). Dessa forma, o backend do projeto foi organizado em classes. A figura 11 demonstra o início de um arquivo em PHP que utiliza POO.

Figura 11 – Exemplo de classe em PHP.

```php
<?php

namespace App\Controllers;
use  \App\Models\UserModel;
use \App\Models\FilesModel;

class CryptoController extends BaseController
{
    public function index(){
        $session = \Config\Services::session();
        if($session->get('logged_in') === NULL ){
            echo view('LoginView');
        }else{
            echo view('CryptoView');
        }
    }
```

Fonte: Elaborado pelo autor, 2025.

3.1 Salvando as informações no banco de dados

O registro dos usuários cadastrados, bem como os detalhes do emissor e do receptor no processo de envio dos arquivos cifrados foram salvos em um banco de dados relacional MySQL. O banco relacional salva as informações em forma de tabelas com identificadores únicos para cada registro. A seguir encontram-se as estruturas das tabelas criadas no banco de dados: A tabela *files* demonstrada na figura 12 e a tabela *users* detalhada na figura 13

Figura 12 – Estrutura da tabela *files*.

#	Nome	Tipo	Colação	Atributos	Nulo	Padrão	Comentários	Extra
1	id	int			Não	*Nenhum*		AUTO_INCREMENT
2	sender_id	int			Não	*Nenhum*		
3	recipient_id	int			Não	*Nenhum*		
4	filename	varchar(255)	utf8mb4_0900_ai_ci		Não	*Nenhum*		
5	file_path	varchar(255)	utf8mb4_0900_ai_ci		Não	*Nenhum*		
6	uploaded_at	datetime			Sim	CURRENT_TIMESTAMP		DEFAULT_GENERATED
7	file_type	varchar(255)	utf8mb4_0900_ai_ci		Sim	*NULL*		

Fonte: Elaborado pelo autor, 2025.

Figura 13 – Estrutura da tabela *users*.

#	Nome	Tipo	Agrupamento (Collation)	Atributos	Nulo	Predefinido	Comentários	Extra
1	**created_at**	datetime			Sim	current_timestamp()		
2	**updated_at**	datetime			Sim	*NULL*		
3	**last_login**	datetime			Sim	*NULL*		
4	**id**	int(11)			Não	*Nenhum*		AUTO_INCREMENT
5	**name**	varchar(150)	utf8mb4_general_ci		Não	*Nenhum*		
6	**email**	varchar(150)	utf8mb4_general_ci		Não	*Nenhum*		
7	**password_hash**	varchar(255)	utf8mb4_general_ci		Não	*Nenhum*		
8	**public_key**	text	utf8mb4_general_ci		Não	*Nenhum*		
9	**private_key_encrypted**	text	utf8mb4_general_ci		Sim	*NULL*		

Fonte: Elaborado pelo autor, 2025.

3.2 Docker

Para ampliar o sistema com a utilização de tecnologias mais modernas, implementou-se o docker. Essa ferramenta, permitiu a construção de um container adequado ao *framework codeigniter* e a linguagem PHP. A figura 14 mostra a tela do docker com a execução do sistema.

Figura 14 – Tela inicial do docker em execução.

	Name	Container ID	Image	Port(s)	CPU (%)	Memory usage...	Memory (%)	Disk	Actions
●	sicra	-	-	-	0.57%	418.8MB / 11.29(	10.88%	157.	
●	sicra_app	74e34aa9deda	sicra-app	8080:80	0%	18.72MB / 3.76GI	0.49%	39.9	
●	sicra_phpmyz	07ac400e58f9	phpmyadm	8081:80	0%	19.08MB / 3.76GI	0.5%	39M	
●	sicra_db	257d3ebd7906	mysql:8.0	3306:3306	0.57%	381MB / 3.76GB	9.89%	78.3	

Fonte: Elaborado pelo autor, 2025.

3.3 Cenário dos testes

Os testes de validação do protótipo, detalhados a seguir no capítulo 4, foram executados em um ambiente de desenvolvimento estritamente local. O sistema foi executado em um notebook equipado com processador Intel Core i7-7500U, 8 GB de RAM a 2133 MHz, GPU integrada de 2 GB e armazenamento de 1,35 TB. Além disso, a plataforma Docker foi utilizada para encapsular e executar o servidor web (Apache), o banco de dados (MySQL) e a aplicação PHP em um contêiner isolado.

É importante ressaltar que o cenário de testes foi controlado e não replicou um ambiente de produção com múltiplos usuários reais em redes distintas. O processo de envio (upload) e recebimento (download) de arquivos foi simulado a partir do mesmo host (máquina local).

4 RESULTADOS

Nesta seção foram realizados testes da aplicação desenvolvida, cabe ressaltar que trata-se de um mínimo produto viável (MVP). Nesse aspecto, os resultados têm por objetivos: validar a criptografia híbrida e a transmissão de arquivos através da web.

Os testes foram realizados em ambiente local através Docker com auxílio de um analisador de pacotes de rede.

4.1 Validação de Upload e Análise de Tráfego de Rede

Nesse teste, realizou-se uma validação do envio de um arquivo com extensão ".txt"com o protocolo HTTP. O objetivo foi identificar a visualização do conteúdo do arquivo transmitido até o servidor.

4.1.1 Envio do arquivo

Para realização deste teste prático, utilizou-se o WireShark. Apos iniciar o procedimento de criptografia, realizou-se o monitoramente dos pacotes da rede loopback.

Com a utilização apenas do protocolo HTTP não há uma camada de segurança extra no envio dos pacotes. Nesse sentido, possibilita-se a visualização do conteúdo do arquivo sendo transmitido do lado cliente para o lado do servidor.

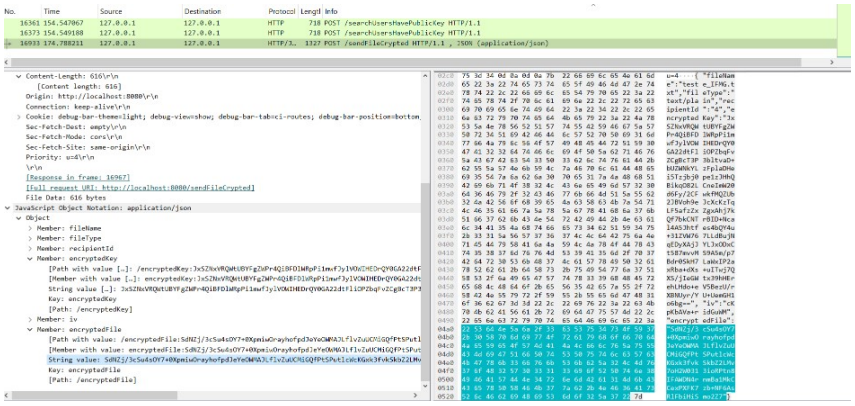
Diante disso, com a utilização da criptografia híbrida (AES-RSA), percebeu-se que o conteúdo do arquivo estava criptografado no momento do conteúdo ser transmitido ao servidor.

A figura 15 mostra o conteúdo do arquivo original. Em seguida, as figuras 16 e 17 apresentam o conteúdo do arquivo criptografado no analisador de pacotes e no servidor local respectivamente.

Figura 15 – Teste - Arquivo original.

Fonte: Elaborado pelo autor, 2025.

Figura 16 – Teste - Arquivo criptografado no analisador de pacotes.



Fonte: Elaborado pelo autor, 2025.

Figura 17 – Teste - Arquivo criptografado no servidor após upload.

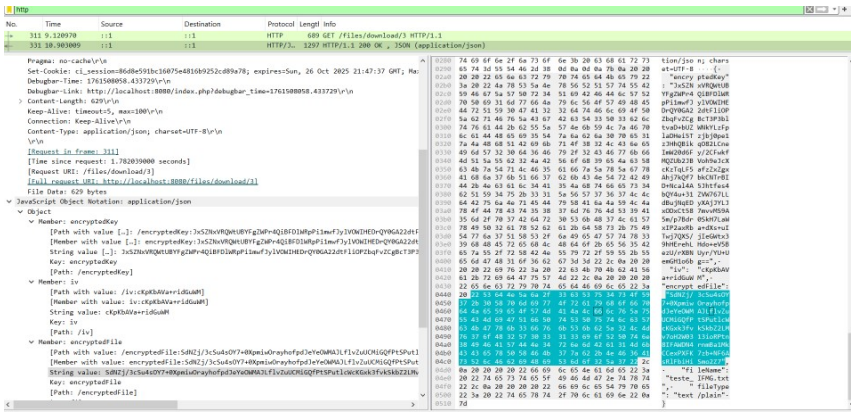
Fonte: Elaborado pelo autor, 2025.

4.1.2 Download do arquivo

A figura 18 demonstra o arquivo criptografado após a requisição de download.

Com efeito, evidenciou-se que o arquivo estava cifrado no servidor e também no processo de transmissão dos dados até a máquina do cliente. Ademais, a figura 19 exibe o resultado final após o processo de descriptografia e download.

Figura 18 – Teste - Arquivo Criptografado no analisador de pacotes após requisição de download.



Fonte: Elaborado pelo autor, 2025.

Figura 19 – Teste - Arquivo após processo de descriptografia.

Fonte: Elaborado pelo autor, 2025.

4.1.3 Análise com HTTPS

Realizou-se também um teste com a camada extra de segurança permitida pelo protocolo HTTPS. Para isso, instalou-se e realizou-se a configuração do Nginx Proxy Manager para permitir a simulação de um certificado SSL. A figura 20 demonstra o arquivo original transmitido.

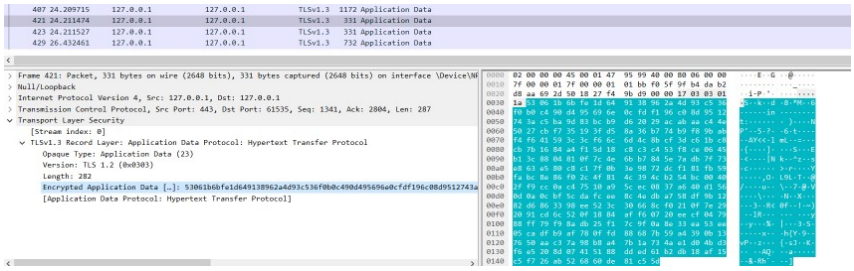
Por sua vez, as figuras 21 e 22 detalham os pacotes cifrados no Wireshark e no momento do download respectivamente.

Por fim, a figura 23 mostra o arquivo original após todo o processo de criptografia.

Figura 20 – Teste com HTTPS - Arquivo original.

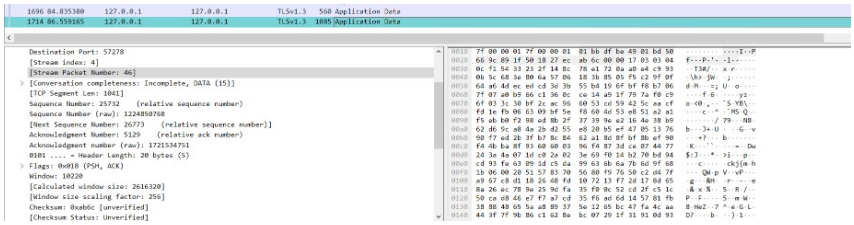
Fonte: Elaborado pelo autor, 2025.

Figura 21 – Teste com HTTPS - Conteúdo dos pacotes cifrados no *upload*.



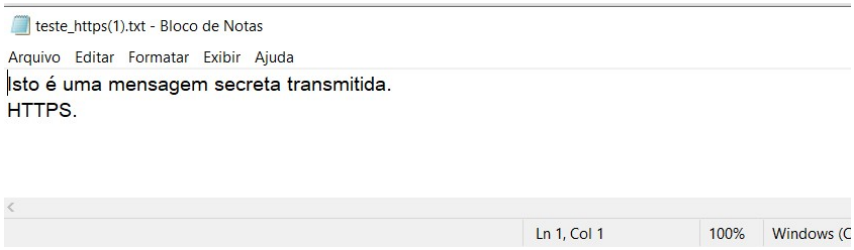
Fonte: Elaborado pelo autor, 2025.

Figura 22 – Teste com HTTPS - Conteúdo dos pacotes cifrados no *download*.



Fonte: Elaborado pelo autor, 2025.

Figura 23 – Teste com HTTPS - Conteúdo original após o processo de transmissão e criptografia.



Fonte: Elaborado pelo autor, 2025.

Diante desses testes, comprovou-se que a cifragem das informações na transmissão dos arquivos no servidor foi realizada com sucesso. Além disso, verificou-se a validação do upload e download dos arquivos.

Diante do exposto, pode-se concluir que os processos sensíveis de cifragem e decifragem das informações ocorreram no lado do cliente com a utilização do javascript.

4.2 Testes de alterações nas chaves pública e privada

Para os testes a seguir foram feitas alterações nas chave privadas ou a utilização incorreta dessas para averiguar se o sistema faz a criptografia ou descriptografia corretamente de acordo com os respectivos pares de chave.

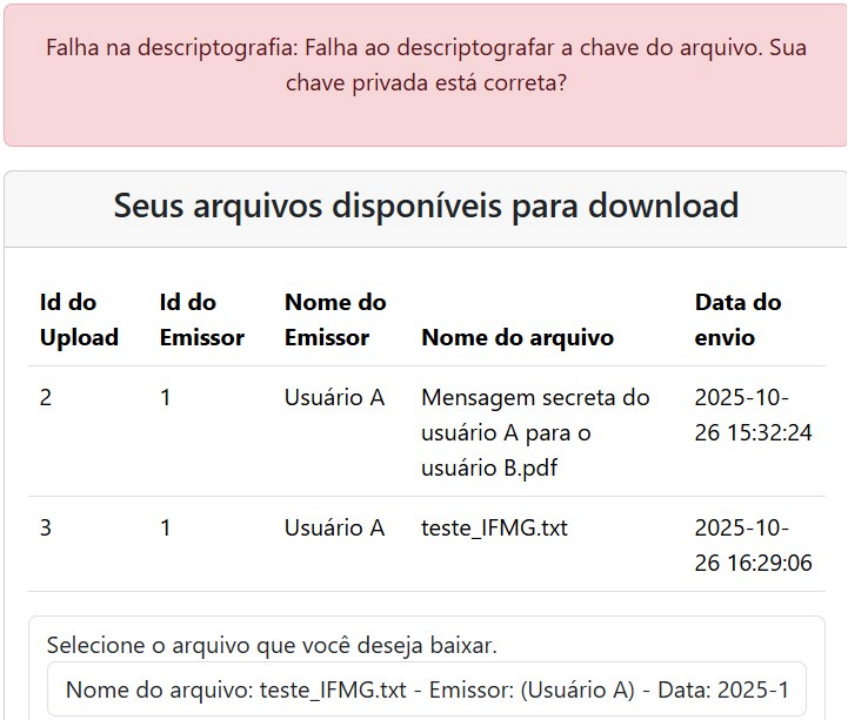
4.2.1 Chave privada incorreta.

Nesse cenário de teste, o usuario A envia um arquivo criptografado para o usuário B. Nesse sentido, a mensagem é cifrada com a chave pública do usuário B e deve ser decifrada somente com a chave privada desse mesmo usuário B.

Partindo disso, em simulação local com ambiente hipotético, a chave privada do usuário A foi enviada para a realizar a descriptografia do arquivo enviado ao invés da chave privada do usuário B.

A figura 24 evidencia que não é possível realizar a descriptografia e consequentemente não há opção para o download do arquivo.

Figura 24 – Teste - Erro em descriptografar o bloco pela inserção da chave privada incorreta.



Fonte: Elaborado pelo autor, 2025.

Com esse teste, mostrou-se que somente o par de chaves do usuário B pode cifrar e decifrar as informações. Em outras palavras, somente o par de chaves correspondente pode realizar a criptografia e posterior descriptografia das mensagens.

4.2.2 A chave privada corrompida

Nesse outro teste, optou-se pela utilização do correto par de chaves do usuário B para cifrar e decifrar as informações.

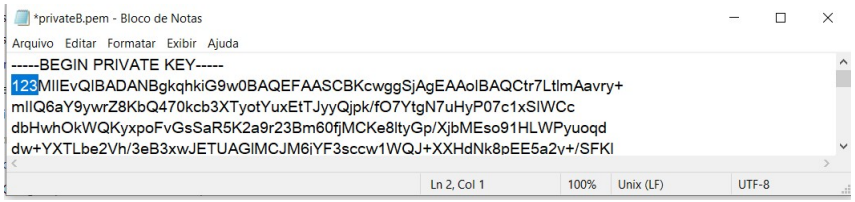
Porém, em um cenário hipotético em que a chave privada desse mesmo usuário B foi alterada ou corrompida, a mensagem também não deverá ser decifrada. A figura 25 detalha a

chave privada original e a figura 26 demonstra essa mesma chave com alteração em seu conteúdo.

Figura 25 – Teste - Trecho da chave privada original do usuário B.

Fonte: Elaborado pelo autor, 2025.

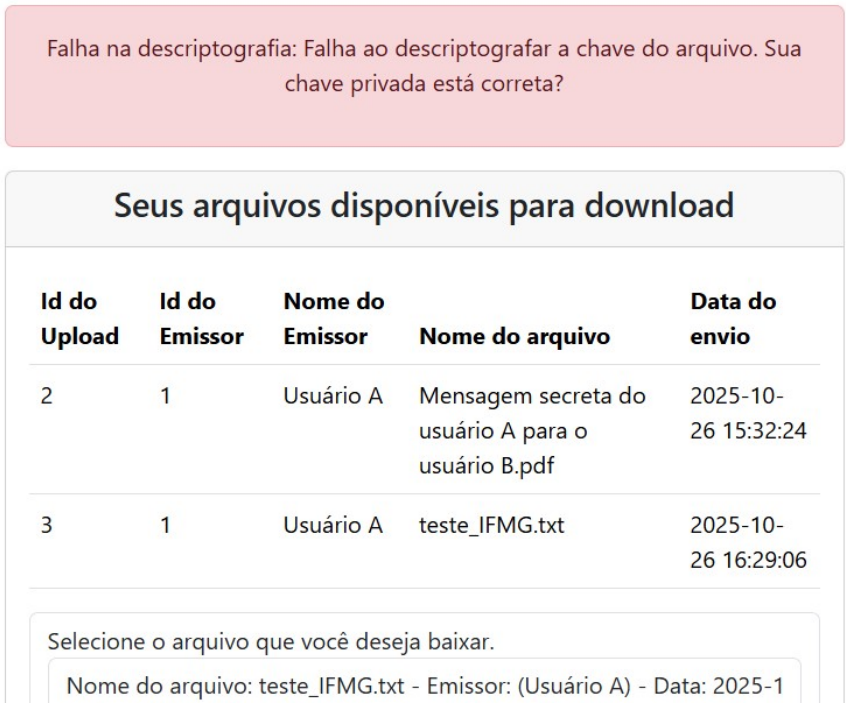
Figura 26 – Teste - Trecho da chave privada alterada do usuário B.



Fonte: Elaborado pelo autor, 2025.

A figura 27, por sua vez, evidencia que qualquer alteração no conteúdo da chave privada do usuário B impossibilita a leitura do texto claro original que foi transmitido criptografado.

Figura 27 – Teste - Erro ao descriptografar a mensagem original em virtude da alteração da chave privada.



Fonte: Elaborado pelo autor, 2025.

4.3 Teste de desempenho

Neste tópico, apresentam-se os testes de desempenho realizados no sistema, cujo objetivo central foi avaliar o impacto dos processos de criptografia, transmissão e descriptografia sobre arquivos de diferentes tamanhos e características. Todas as medições foram executadas em ambiente local, com o sistema em execução dentro de um contêiner Docker.

A aferição do tempo de processamento foi realizada utilizando as ferramentas de desenvolvimento (DevTools) do navegador Google Chrome, especificamente na aba Network. O método consistiu em cronometrar o intervalo entre o início da requisição (disparo do evento de upload ou download) e a resposta final do servidor (status 200 OK), capturando o parâmetro de duração total da requisição.

Esse intervalo total engloba o processo de leitura do arquivo, a execução do algoritmo de criptografia (via JavaScript no cliente) e a transmissão dos dados pela rede local (loopback). Para garantir a consistência estatística e mitigar oscilações pontuais de processamento do sistema operacional, cada arquivo foi submetido a três baterias de testes independentes, sendo calculada a média aritmética simples dos tempos obtidos em cada etapa.

4.3.1 Testes com somente texto

Os testes foram conduzidos utilizando três arquivos no formato “.docx”, com tamanhos de 50 KB, 200 KB e 1000 KB, respectivamente. Para análise do comportamento em diferentes formatos, esses mesmos arquivos — contendo apenas texto — foram posteriormente convertidos para “.pdf”.

A coleta dos tempos de execução das operações foi realizada por meio do DevTools do navegador, utilizando as ferramentas de análise de rede (Network), de modo a registrar dados precisos de envio (upload) e recebimento (download).

Cada arquivo foi submetido a três execuções independentes dos processos de criptografia e descriptografia. Ao final, foram calculadas as médias dos tempos obtidos em cada etapa, permitindo uma análise mais consistente do desempenho do sistema.

Por padrão, o PHP aceita um limite máximo de 8 MB de upload. Nesse sentido, o último arquivo enviado para teste (teste3.pdf) apresentava tamanho superior a esse limite no momento do upload.

Contudo, esse limite pode ser alterado de acordo com a disponibilidade de recursos do servidor e do sistema operaional nas configurações do PHP.

Para os fins desse protótipo, o envio de provas dificilmente ultrapassará esse limite 8MB. Além disso, trata-se de uma configuração que permite alteração para casos excepcionais (se necessário).

As tabelas 4 e 5 detalham respectivamente o tempo médio de criptografia e descriptografia dos arquivos.

Tabela 4 – Avaliação dos tempos de criptografia de arquivos com somente texto.

Arquivo	**Tamanho original (KB)**	**T1 Cripto (s)**	**T2 Cripto (s)**	**T3 Cripto (s)**	**Tempo médio Cripto (s)**
texto1.docx	50	1,93	3,27	3,48	2,89
texto1.pdf	681	1,78	1,78	2,59	2,05
texto2.docx	200	3,83	3,08	3,05	3,32
texto2.pdf	3377	3,53	2,70	2,28	2,84
texto3.docx	1000	3,03	3,23	3,25	3,17
texto3.pdf	17549	-	-	-	-

Fonte: autor, 2025.

Tabela 5 – Avaliação dos tempos de descriptografia de arquivos com somente texto.

Arquivo	**Tamanho cifrado (KB)**	**T1 Decripto (s)**	**T2 Decripto (s)**	**T3 Decripto (s)**	**Tempo médio Decripto (s)**
texto1.docx	67	2,43	2,73	2,60	2,57
texto1.pdf	922	1,82	1,91	2,95	2,23
texto2.docx	271	1,75	1,78	1,87	1,80
texto2.pdf	4520	3,08	3,12	2,49	2,90
texto3.docx	1000	1,79	2,43	2,75	2,32
texto3.pdf	18200	-	-	-	-

Fonte: autor, 2025.

Com base nas tabelas 4 e 5, os resultados demonstram que os tempos de criptografia e descriptografia apresentam variações significativas entre as tentativas. Observou-se também que não houve relação linear direta entre o tamanho dos arquivos e os tempos medidos, uma vez que arquivos maiores, como alguns PDFs, apresentaram tempos médios inferiores aos de arquivos menores. Por fim, pode-se concluir que a descriptografia mostrou-se mais rápida que a criptografia.

Cabe ressaltar que a variabilidade de tempos obtida evidencia a necessidade de ambientes mais controlados, isto é, uma ampliação nos contextos de teste a fim de aumentar a confiabilidade dos resultados e compreender com maior precisão os fatores que influenciam o desempenho do sistema. Porém, com a finalidade de apresentar um protótipo e considerando que o fator tempo não é estritamente essencial para o cenário desse trabalho, os testes se mostraram satisfatórios dentro dos objetivos propostos.

4.3.2 Testes de texto com imagens

Para os testes a seguir, realizou-se o mesmo padrão dos tamanhos do arquivos de extensão ".docx". Mas, acrescentou-se algumas imagens aleatórias aos documentos a fim de verificar o desempenho dentro do cenário já exposto de testes. As tabelas 6 e 7 detalham os tempos no processo de criptografia e descriptografia, bem como os tamanhos dos arquivos após o processo de cifragem.

Tabela 6 – Avaliação dos tempos de criptografia de arquivos com texto e imagens.

Arquivo	**Tamanho original (KB)**	**T1 Cripto (s)**	**T2 Cripto (s)**	**T3 Cripto (s)**	**Tempo médio Cripto (s)**
textoImagem1.docx	50	3,03	1,19	2,95	2,39
textoImagem1.pdf	433	1,73	1,88	2,26	1,97
textoImagem2.docx	200	3,01	2,75	2,66	2,80
textoImagem2.pdf	2192	1,91	2,16	2,72	2,26
textoImagem3.docx	1000	3,36	1,93	2,91	2,73
textoImagem3.pdf	11234	-	-	-	-

Fonte: autor, 2025.

Tabela 7 – Avaliação dos tempos de descriptografia de arquivos com textos e imagens.

Arquivo	**Tamanho cifrado (KB)**	**T1 Decripto (s)**	**T2 Decripto (s)**	**T3 Decripto (s)**	**Tempo médio Decripto (s)**
textoImagem1.docx	68	1,11	1,95	1,84	1,63
textoImagem1.pdf	586	2,19	2,50	2,25	2,31
textoImagem2.docx	271	2,14	2,22	2,61	2,32
textoImagem2.pdf	2969	2,30	2,19	2,55	2,35
textoImagem3.docx	1354	2,16	2,18	2,40	2,25
textoImagem3.pdf	15337	-	-	-	-

Fonte: autor, 2025.

Nos testes envolvendo arquivos com texto e imagens, observou-se também variabilidade nas medições. Em que, embora o formato PDF amplifique significativamente o tamanho dos arquivos, o tempo de processamento não cresce proporcionalmente. Portanto, destaca-se a necessidade de testes em contextos variados para ampliar a análise do real desempenho do sistema.

Para validar se cada arquivo após todo o processo de criptografia e descriptografia se manteve íntegro, realizou-se a execução do comando *diff* no terminal do servidor. Esse comando retorna as diferenças encontradas na comparação dos dois arquivos

Na figura 28, o arquivo original `texto1.docx` foi comparado com o arquivo após o processo de criptografia, transmissão e descriptografia (`texto1_.docx`).

Figura 28 – Teste - Comparação de arquivos atráves do terminal.

Fonte: Elaborado pelo autor, 2025.

Como não houve nenhuma mensagem de retorno no terminal, validou-se a integridade dos arquivos, isto é, eles não sofreram modificações após a transmissão de conteúdo através da web.

Em resumo, pode-se destacar a limitação no envio dos arquivos grandes acima de 8MB para o servidor. Ademais, a necessidade de realizar testes em outros contextos para validar de forma mais exata as reais métricas de desempenho do MVP.

4.3.3 Dados e códigos do trabalho

O código da aplicação, bem como os arquivos utilizados nos testes de desempenho estão disponíveis no Github. O link encontra-se no Anexo A.

5 CONCLUSÃO E TRABALHOS FUTUROS

Neste trabalho, realizaram-se testes que permitiram validar o funcionamento do sistema proposto de criptografia híbrida AES e RSA para transmissão de arquivos via web. A análise do tráfego de rede demonstrou que, mesmo utilizando o protocolo HTTP, o conteúdo transmitido permaneceu cifrado durante o envio e o recebimento dos dados, comprovando a efetividade da camada de criptografia aplicada no lado do cliente (javascript). Com a utilização do protocolo HTTPS, notou-se que o conteúdo também se manteve cifrado tanto no upload quanto no download das informações.

Além disso, a tentativa de descriptografar o arquivo utilizando uma chave privada incorreta ou corrompida resultou em falha, confirmando que apenas o par de chaves original é capaz de restaurar o conteúdo cifrado. Isso evidencia a correta aplicação dos princípios e conceitos de segurança baseados na criptografia assimétrica (chave pública e privada).

Com base nos experimentos realizados, conclui-se que a abordagem de Criptografia Híbrida (AES-RSA) se mostrou a mais adequada para o problema proposto. Os testes de desempenho evidenciaram que o AES manteve tempos de processamento viáveis, enquanto o RSA cumpriu eficazmente o papel de proteger apenas a chave de sessão. Tentar utilizar puramente o RSA para cifrar os arquivos inteiros inviabilizaria o sistema devido à lentidão exponencial com o aumento do tamanho do arquivo.

Portanto, o modelo híbrido validou-se como o equilíbrio ideal entre a performance necessária para a usabilidade do sistema e a segurança criptográfica exigida para o sigilo das provas.

Diante de todo o exposto, é fundamental reconhecer que todo software pode estar sujeito a falhas de segurança. Dentro desse contexto, o sistema que se constitui um MVP, atendeu ao objetivo de validar a eficácia da criptografia híbrida para a transmissão de arquivos.

A solução proposta resolveu a necessidade central do processo realizado pela Funec: realizar o transporte seguro de provas e ampliar a faixa geográfica para contratação de novos colaboradores (professores).

5.1 Vulnerabilidades

Entretanto, como um protótipo focado na validação da criptografia, o sistema apresenta vulnerabilidades inerentes ao processo de uso e à gestão de credenciais, que não fizeram parte do objetivo desse trabalho:

- Gerenciamento de credenciais de acesso: A segurança do sistema depende do sigilo da senha e do login do usuário. O *software* não implementa mecanismos que impeçam o compartilhamento de senhas entre os usuários (por exemplo, um professor compartilhando seu acesso com terceiros).

- Gerenciamento da chave privada: Os testes demonstraram que apenas a chave correta e íntegra pode decifrar os arquivos. Contudo, se um usuário (colaborador) não armazenar seu arquivo de chave privada com segurança ou compartilhá-lo, a confidencialidade dos seus arquivos estará comprometida, bem como a segurança de todo o processo de transmissão de provas.
- Restrição de ambiente: O sistema não possui validação de ambiente, como a vinculação do acesso a um endereço MAC específico ou a restrição por faixas de IP. Isso significa que um usuário autorizado (ou alguém que obteve suas credenciais) pode acessar o sistema e baixar arquivos de qualquer dispositivo conectado à internet.

5.2 Trabalhos Futuros

Apesar do sistema ser seguro dentro do proposto (garantir a confidencialidade dos dados em trânsito), existem vulnerabilidades de processo que indicam pontos claros de melhoria.

Para trabalhos futuros, a implementação da autenticação de dois Fatores (2FA) e da assinatura digital são mecanismos relevantes que ampliarão a segurança do sistema. A 2FA permite uma segurança maior no login do usuário, protegendo contra o acesso não autorizado. Por sua vez, a assinatura digital garante a autenticidade e a integridade dos arquivos trocados.

Por fim, a implementação de uma política para controle das chaves públicas e privadas elevaria a segurança do sistema, bem como a confidencialidade de todo o processo sigiloso de criação e compartilhamento de provas.

REFERÊNCIAS

ABOOD, O.; GUIRGUIS, S. A survey on cryptography algorithms. **International Journal of Scientific and Research Publications**, v. 8, p. 495–516, jul 2018. Citado 2 vezes nas páginas 21 e 28.

AMARAL, ; KREUTZ, D.; ANDRADE, E. R.; LUNARDI, R. C. **Unihacker: fundamentos da segurança I**. Bento Gonçalves, RS: IFRS, 2021. Disponível em: <https://dspace.ifrs.edu.br/xmlui/handle/123456789/726>. Acesso em: 26 maio 2025. Citado 2 vezes nas páginas 20 e 21.

Bootstrap Core Team. **Overview · Bootstrap 4.0**. 2025. Disponível em: <https://getbootstrap.com/docs/4.0/layout/overview/>. Acesso em: 22 jul. 2025. Citado na página 18.

BRAGA, A.; DAHAB, R. Introdução à criptografia para programadores: Evitando maus usos da criptografia em sistemas de software. In: **Anais do XV Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais (SBSeg 2015)**. Florianópolis: [s.n.], 2015. Acesso em: 26 maio 2025. Citado 2 vezes nas páginas 14 e 20.

Brasil. **Constituição da República Federativa do Brasil de 1988**. Brasília, DF: Presidência da República, 1988. Disponível em: <https://www.planalto.gov.br/ccivil_03/constituicao/constituicao.htm>. Acesso em: 14 maio 2025. Citado na página 15.

CASTILHO, M. A.; SILVA, F.; WEINGAERTNER, D. **Algoritmos e Estruturas de Dados I**. Curitiba: Universidade Federal do Paraná, 2020. ISBN 978-65-86233-62-9. Citado na página 23.

CAVALCANTE, A. L. B. Teoria dos números e criptografia. **Revista de Informática - UPIS**, 2005. Disponível em: <https://institucional.upis.br/biblioteca/pdf/revistas/revista_informatica/Cavalcante_teoria_numeros_criptografia_2005_UPIS.pdf>. Acesso em: 05 jun. 2025. Citado 2 vezes nas páginas 25 e 29.

CodeIgniter Foundation. **CodeIgniter 4 User Guide**. [S.l.], 2025. Disponível em: <https://www.codeigniter.com/user_guide/intro/index.html>. Acesso em: 20 jul. 2025. Citado na página 18.

COSTA, C.; FIGUEIREDO, L. M. **Introdução à Criptografia – Volume 1**. Rio de Janeiro: Fundação CECIERJ, 2010. Disponível em: <https://canal.cecierj.edu.br/recurso/4687>. Acesso em: 28 maio 2025. Citado na página 22.

COUTINHO, S. C. **Números Inteiros e Criptografia RSA**. 2. ed. Rio de Janeiro: IMPA, 2009. ISBN 978-8524401244. Citado 3 vezes nas páginas 25, 26 e 27.

docker-compose.de. **Nginx Proxy Manager**. 2025. Disponível em: <https://docker-compose.de/en/proxy/nginx-proxy-manager/>. Acesso em: 15 nov. 2025. Citado na página 19.

Docker Inc. **Docker Desktop**. 2025. Disponível em: <https://docs.docker.com/desktop/>. Acesso em: 15 out. 2025. Citado na página 19.

ESPíNDOLA, E. B. d. M.; SILVA, F. B. M. d.; SANTOS, M. P. d. **Coletânea de estudos de egressos do PROFMAT-UFRPE**. São Paulo: Editora Edgard Blücher Ltda., 2021. Open Access. ISBN 978-65-5550-094-3. Citado na página 28.

MDN Web Docs. **HTML - HyperText Markup Language**. 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/HTML>. Acesso em: 16 jul. 2025. Citado na página 18.

____. **JavaScript**. 2025. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 16 jul. 2025. Citado na página 19.

____. **What is CSS?** 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Learn_web_development/Core/Styling_basics/What_is_CSS>. Acesso em: 16 jul. 2025. Citado na página 19.

MILANOV, Y. **The RSA Algorithm**. 2009. Disponível em: <http://susanka.org/MathPhysics2/RSA_Algorithm_Yevgeny.pdf>. Acesso em: 05 jun. 2025. Citado na página 28.

Mozilla Developer Network. **Web Crypto API**. 2025. Disponível em: <https://developer.mozilla.org/en-US/docs/Web/API/Web_Crypto_API#specifications>. Acesso em: 24 out. 2025. Citado na página 18.

OLIVEIRA, R. Criptografia simétrica e assimétrica: os principais algoritmos de cifragem. **Revista Segurança Digital**, v. 5, p. 11–24, mar 2012. Citado 5 vezes nas páginas 21, 22, 23, 24 e 25.

Oracle Corporation. **About MySQL**. 2025. Disponível em: <https://www.mysql.com/about/>. Acesso em: 22 jul. 2025. Citado na página 18.

PFLEEGER, C. P.; PFLEEGER, S. L.; COLES-KEMP, L. **Security in Computing**. 6. ed. Boston: Addison-Wesley Professional, 2023. ISBN 978-0137891214. Citado 2 vezes nas páginas 23 e 28.

REZENDE, E. d. S. **Modelo estrutural para compartilhamento de arquivos peer-to-peer**. Dissertação (Mestrado) — Universidade Estadual Paulista, Faculdade de Ciências, Bauru, 2009. Disponível em: <https://repositorio.unesp.br/server/api/core/bitstreams/82705100-ec20-47b3-8983-ba7ce5e980df/content>. Acesso em: 22 maio 2025. Citado na página 14.

SERAFIM, V. d. S. **Roteiro 05 – Cifras de Fluxo e Bloco**. 2012. Versão 1 - 22/ago./2012. Disponível em: <http://www.serafim.eti.br/academia/recursos/Roteiro_05-Cifras_de_Fluxo_e_Bloco.pdf>. Citado na página 22.

SOUZA, R. d. A. d.; OLIVEIRA, F. B. d.; CUNHA, G. N. d. **O algoritmo AES: Apresentação e Descrição da Estrutura**. 2025. Disponível em: <https://www.lncc.br/~borges/doc/O%20algoritmo%20AES%20Apresentacao%20e%20Descricao%20da%20Estrura.pdf>. Acesso em: 26 out. 2025. Citado 2 vezes nas páginas 29 e 30.

STALLINGS, W. **Criptografia e segurança de redes: princípios e práticas**. 6. ed. São Paulo: Pearson, 2014. ISBN 9788543005890. Citado 2 vezes nas páginas 21 e 22.

SUNEETHA, K.; ANURADHA, K.; NARASIMHAM, C. S. L. RSA-AES Hybrid Encryption: Combining the strengths of two powerful algorithms for enhanced security. **International Journal of Research and Analytical Reviews (IJRAR)**, v. 10, n. 2, p. 968–971, jun 2023. Disponível em: <https://www.ijrar.org/papers/IJRAR23B1852.pdf>. Acesso em: 20 out. 2025. Citado na página 31.

TANENBAUM, A. S. **Computer Networks**. 4. ed. Upper Saddle River: Prentice Hall, 2003. Citado na página 14.

TERADA, R. **Segurança de Dados: Criptografia em Rede de Computador**. 2. ed. São Paulo: Edgard Blucher, 2008. ISBN 978-85-212-0439-8. Citado na página 20.

The PHP Group. **Manual PHP: Introdução ao PHP**. 2025. Disponível em: <https://www.php.net/manual/pt_BR/introduction.php>. Acesso em: 18 jul. 2025. Citado na página 18.

The Wireshark team. **Wireshark User's Guide - 1.1. What Is Wireshark?** 2025. Disponível em: <https://www.wireshark.org/docs/wsug_html_chunked/ChapterIntroduction.html#ChIntroWhatIs>. Acesso em: 15 nov. 2025. Citado na página 19.

TIDWELL, T. **JSEncrypt: A JavaScript RSA Encryption Library**. 2025. Disponível em: <https://travistidwell.com/jsencrypt/>. Acesso em: 24 out. 2025. Citado na página 18.

VALENTE, M. T. **Engenharia de Software Moderna: Princípios e Práticas para Desenvolvimento de Software com Produtividade**. 1. ed. Brasil: Independente, 2022. 408 p. ISBN 978-65-00-01950-6. Citado na página 17.

ANEXO A – CÓDIGO FONTE

O código fonte da protótipo desenvolvido está disponível no repositório público do GitHub através do link: <https://github.com/Gabriel-Keven/sicra>.