



Universidade Federal do Piauí

Centro de Ciências da Natureza/CCN

Departamento de Computação

Disciplina: Segurança em Sistemas

Docente: Dr. Carlos André Batista de Carvalho

Relatório de Implementação do Envelope Digital

Discentes:

Ciro Olímpio de Melo

Enzo Eduardo Cassiano Ibiapina

Guilherme Eduardo Almeida Martinelis

Teresina – PI

Julho, 2023

Introdução -

O objetivo deste relatório é descrever o objetivo, o funcionamento e as particularidades da nossa implementação do Envelope Digital para a disciplina de Segurança em Sistemas.

O cerne do trabalho consiste em aplicar três algoritmos de cifragem de chave simétrica (DES, AES e RC4) e então criar um sistema de envelope digital (utilizando o sistema criptográfico RSA de chaves assimétricas) para adicionar uma nova camada de proteção. Obviamente, a aplicação gerada deve ser capaz de então decifrar as mensagens que cifrou previamente.

Para tal, usamos a linguagem de programação Java, usando as bibliotecas recomendadas (crypto, io, security, util, entre outras). Por questão de usabilidade, foi gerada uma interface simples utilizando a biblioteca Swing (JPanel, JFrame, JTabbedPane) para que o usuário pudesse utilizar nossos algoritmos sem nenhuma dificuldade.

Os Algoritmos -

Utilizamos três algoritmos de criptografia com chaves simétricas (DES, AES e RC4) e um algoritmo de criptografia com chave assimétrica (RSA), que serão descritos de maneira sucinta a seguir:

DES - O algoritmo *Data Encryption Standard* é um algoritmo de criptografia que utiliza um par de chaves simétricas para cifrar e decifrar dados digitais. Surgiu na década de 70 como sucessor do conjunto de algoritmos de cifragem conhecidos como Lucifer, o DES foi um algoritmo muito influente no desenvolvimento de ferramentas criptográficas modernas. Utiliza uma chave de 56 bits para cifrar blocos de dados de 64 bits através de 16 rodadas de processamento dentro de uma estrutura balanceada de Feistel, com a inclusão de uma permutação prévia e uma posterior. Atualmente, é considerado um algoritmo inseguro devido ao seu curto tamanho de chave, que o deixa suscetível a ataques baseados em força bruta.

AES - O algoritmo *Advanced Encryption Standard* (também conhecido como Rijndael por conta de cifras que lhe originaram) é um algoritmo de criptografia que utiliza um par de chaves simétricas para cifrar e decifrar dados digitais. Surgiu no final da década de 90 e rapidamente sucedeu o DES como algoritmo de criptografia padrão do governo americano. O algoritmo trabalha com blocos de dados de 128 bits e chaves de 128, 192 ou 256 bits, sendo que o tamanho da chave irá afetar o processo de cifragem. O processo de cifragem do AES consiste na expansão da chave inicial para gerar um conjunto de subchaves, então na aplicação da primeira dessas subchaves a mensagem. Após isso, ocorrem múltiplos (número exato relacionado ao tamanho da chave inicial) ciclos que consistem na substituição de bits, deslocamento de linhas, embaralhamento de colunas e adição de outra subchave, sendo que o último desses ciclos não contará com o embaralhamento de colunas. Até os dias de hoje o AES segue sendo utilizado e não foi descoberto nenhum método de quebra viável para sua criptografia.

RC4 - O algoritmo *Rivest Cypher 4* é uma cifra de fluxo, desenvolvida no final da década de 80 e vazada publicamente na década de 90. O RC4 é conhecido por sua simplicidade e velocidade, mas infelizmente também possui inúmeras vulnerabilidades. Para realizar sua cifragem, o RC4 utiliza uma chave de fluxo (um fluxo pseudo aleatório de bits) para criptografar a sua mensagem usando seus dados e operações XOR com a chave. Para gerar essa chave, o RC4 usa uma permutação de todos os 256 bytes possíveis e uma dupla de

ponteiros de índice. Atualmente, o uso do RC4 é fortemente desencorajado, pois diversas vulnerabilidades foram descobertas ao longo dos anos. O vazamento da chave inicial, já que a chave não é totalmente aleatória, compromete a chave secreta usada para a cifragem da mensagem. Atacantes podem descobrir a chave usando análises de correlação se tiverem acesso a múltiplas mensagens que utilizem a mesma chave, entre outras coisas.

RSA - O algoritmo *Rivest-Shamir-Adleman* é uma das mais antigas cifras de chaves assimétricas. Publicado em 1977, é um algoritmo lento, então geralmente não é utilizado para fazer a criptografia direta de um conteúdo, mas sim para fortalecer ainda mais a segurança de uma mensagem já cifrada, como é o caso do envelope digital. Cifras de chaves assimétricas tem esse nome porque a chave utilizada para criptografar e a chave usada para descriptografar são diferentes, sendo uma delas pública e a outra privada, então o emissor utiliza a chave pública do destinatário para cifrar a mensagem, sendo que esta só pode ser decifrada com o uso da chave privada do destinatário, que deve ser mantida em segredo de todos os outros. A geração de chave do RSA inicia com a escolha de dois números primos grandes, distintos e distantes, e a partir deles é realizada uma sequência de operações matemáticas que termina por gerar duas chaves distintas, sendo uma delas a pública e a outra a privada. Atualmente, existem estratégias dedicadas de ataque ao RSA puro, algumas das quais se aproveitam do desempenho pobre do algoritmo, mas isso não surte muito efeito no uso mais corriqueiro do algoritmo, que é como uma camada extra de segurança, e não a principal.

Modo de Uso -

Pré-Requisitos:

Para executar aplicação com o .jar incluso:

- Versão compatível do Java Runtime Environment (Java 11 ou mais recente)

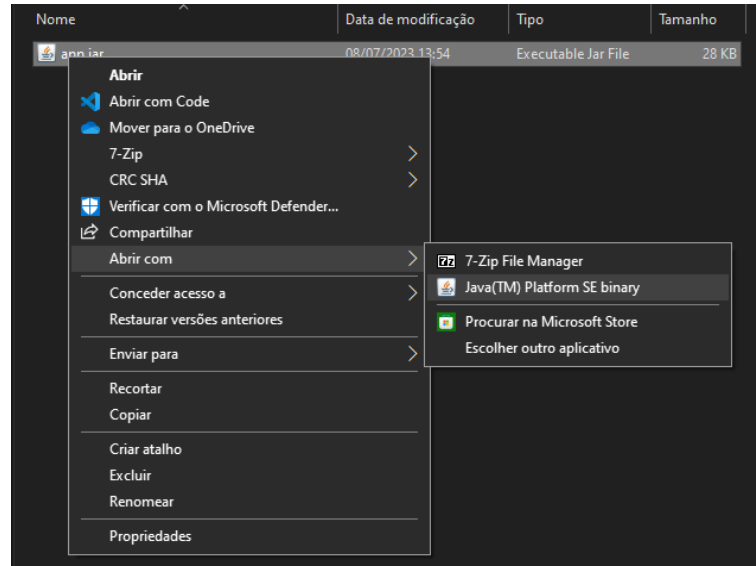
Para executar aplicação via código:

- Uma IDE capaz de rodar códigos em Java

1º Passo - Inicialização da Aplicação

Via app.jar:

- Simplesmente abra a aplicação utilizando o Java da sua máquina



- Opcionalmente, também é possível executar o .jar diretamente através do seguinte comando (realizado no diretório onde o arquivo está localizado)

```
C:\Windows\System32\cmd.exe - java -jar app.jar
Microsoft Windows [versão 10.0.19045.3086]
(c) Microsoft Corporation. Todos os direitos reservados.

C:\Users\PICHAU\Documents\teste>java -jar app.jar
```

Via código fonte:

- Rode o código MainEnv.java no diretório “env-dig-sys\src\envelope” do nosso projeto

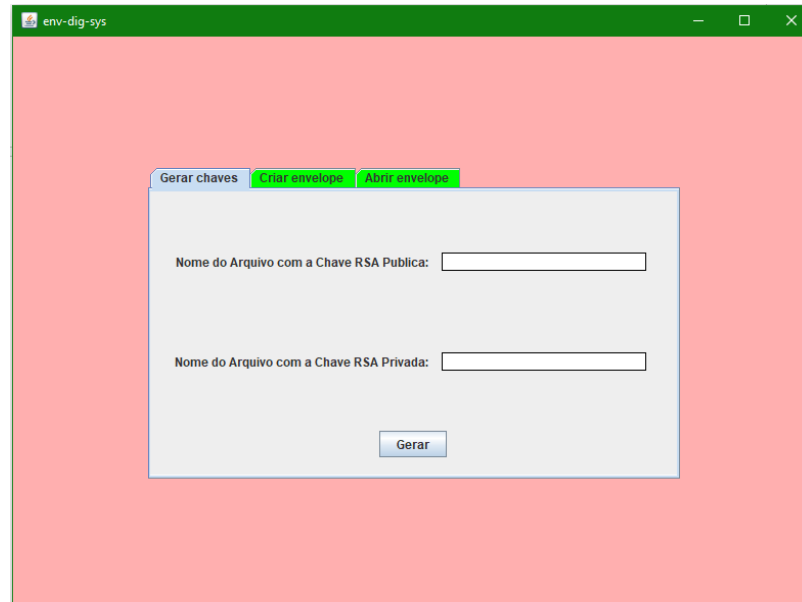
2º Passo - Telas e Funcionalidades

Geral:

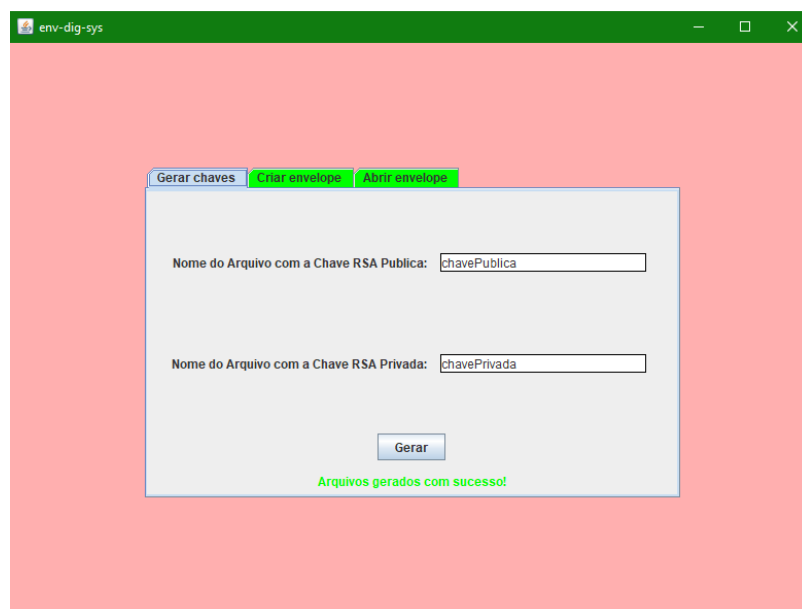
- O aplicativo funciona primariamente com a criação e manipulação de arquivos .txt que serão gerados dentro do diretório do projeto ou do diretório onde está localizado o nosso .jar (dependendo da maneira de execução). Ele presume a existência de um arquivo de texto não vazio inicial que servirá como a mensagem que será cifrada e decifrada pela aplicação.
- Serão exibidas mensagens codificadas por cor (verde em caso de sucesso, vermelho caso seja encontrado algum erro nos inputs do usuário e laranja em caso de algum erro de processamento) na parte inferior da aplicação após cada instrução. Mensagens de erro indicam o motivo que levou aquele erro.

- Existem três telas na aplicação, indicadas por seu título na parte superior: “Gerar chaves” – utilizada para gerar a chave pública e a chave privada que será utilizada pelo RSA –, “Criar envelope” – onde será realizada a criptografia da mensagem pelo algoritmo escolhido e de ambas a mensagem criptografada e a chave cifrada pela chave pública do RSA – e “Abrir envelope” – onde inserimos a chave cifrada, a mensagem duplamente cifrada e a chave privada para enfim decifrar a mensagem, gerando o arquivo com o texto descriptografado. As telas devem ser percorridas de maneira sequencial.

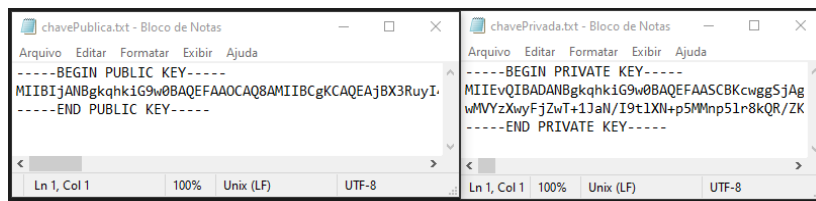
Tela de Gerar Chaves -



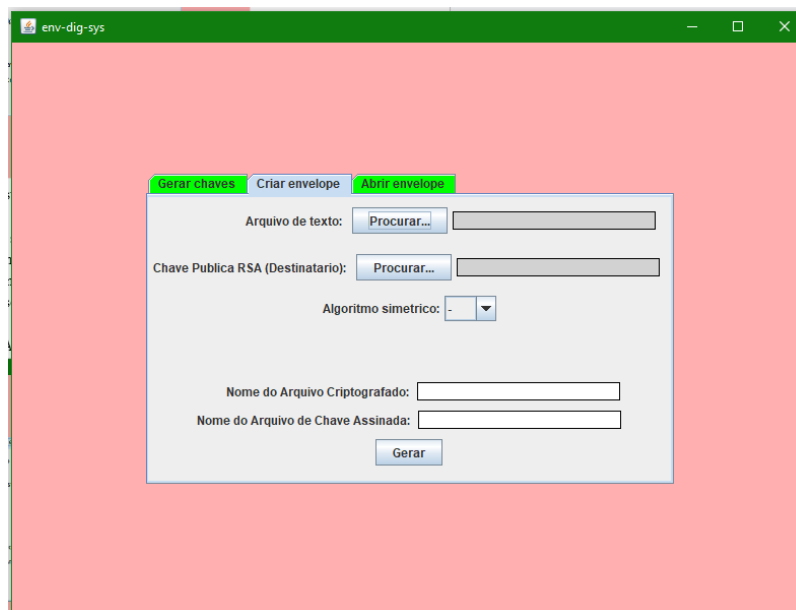
- Essa tela possui dois campos abertos para inputs do usuário. Os inputs serão utilizados para nomear os arquivos .txt das chaves pública e privada do nosso RSA. Os arquivos serão gerados na pasta de onde a aplicação está sendo executada.



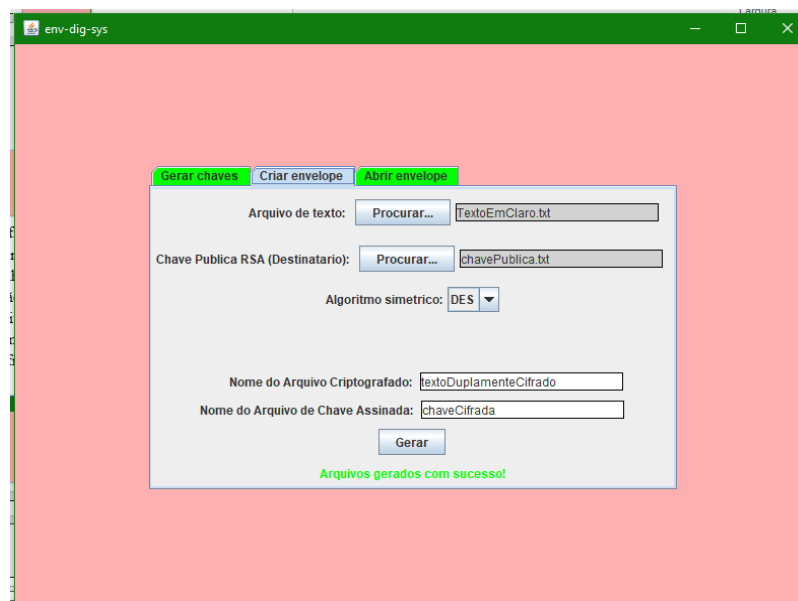
- As chaves geradas seguem o padrão de cabeçalho e rodapé apresentados no RFC 7468 para chave pública e privada, e esse padrão é utilizado para barrar chaves inválidas nas outras telas da aplicação.



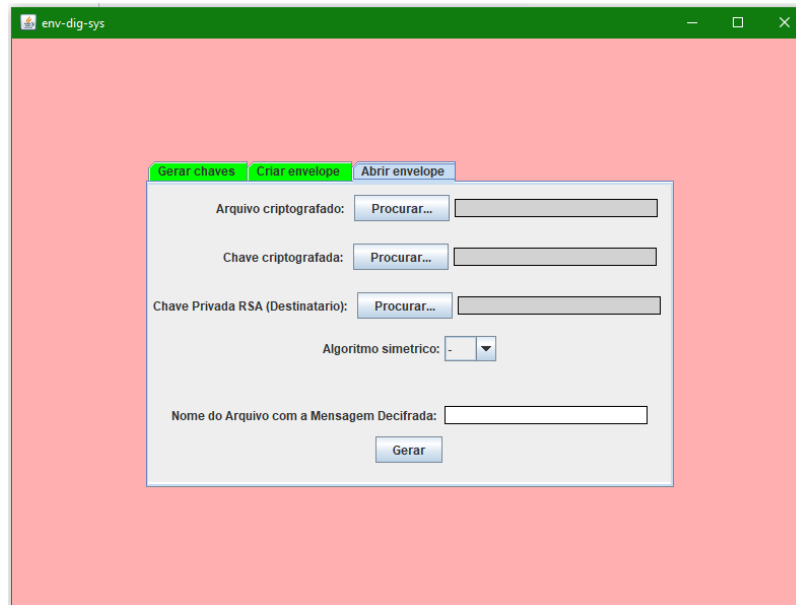
Tela de Criar Envelope -



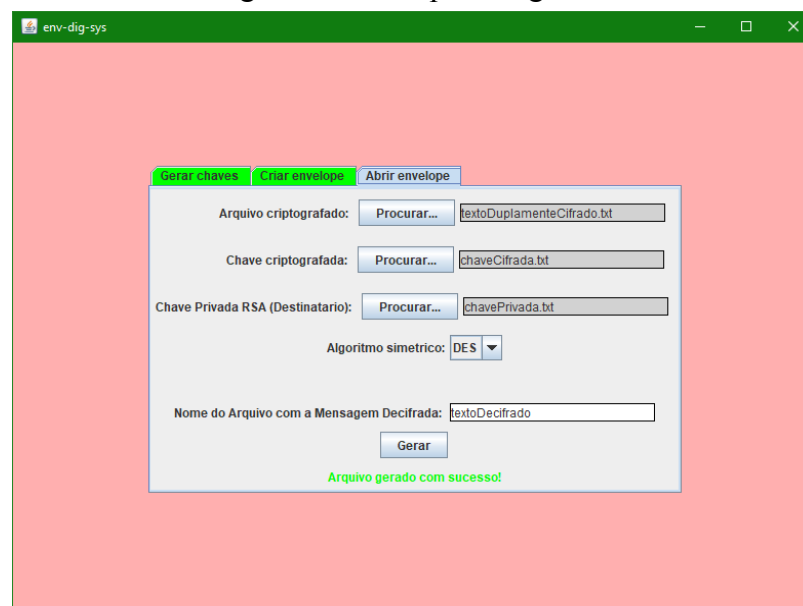
- Essa tela possui duas instâncias de browse for file, uma para procurar por nosso arquivo de texto e outra para buscar pela nossa chave pública gerada anteriormente. Temos um selection box para escolher qual algoritmo de chave simétrica queremos usar na nossa cifragem e então dois espaços abertos para inputs do usuário para inserir o nome dos arquivos .txt que serão gerados referentes a chave do nosso algoritmo de chave simétrica cifrado pelo RSA e o do nosso arquivo de texto que está duplamente cifrado, pela chave simétrica e pela chave pública do RSA.



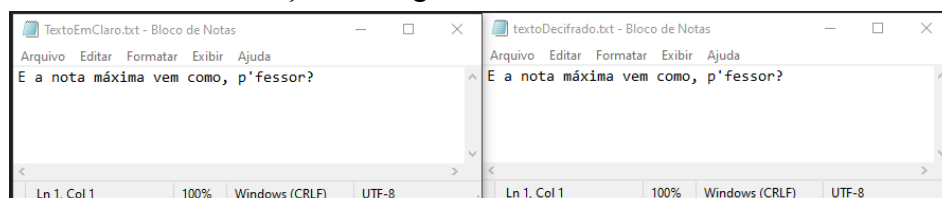
Tela de Abrir Envelope -



- Essa tela possui três instâncias de browse for file, um selection box para selecionarmos o mesmo algoritmo que previamente havia sido selecionado e um espaço aberto para inputs do usuário para inserir o nome do arquivo .txt que irá conter a mensagem decifrada pelos algoritmos.



- Ao fim desse conjunto de execuções, temos que comparar o nosso texto decifrado com o nosso texto cifrado para que nos certifiquemos que tudo correu bem na execução dos algoritmos.



Referências Bibliográficas:

JOSEFSSON, S.; LEONARD, S. Textual Encodings of PKIX, PKCS, and CMS Structures. Disponível em: <<https://www.rfc-editor.org/rfc/rfc7468>>. Acesso em: 8 jul. 2023.

STALLINGS, W. Criptografia e Segurança de Redes: princípios e práticas. 6 ed. São Paulo: Pearson, 2014.

WIKIPEDIA CONTRIBUTORS. Data Encryption Standard. Disponível em: <https://en.wikipedia.org/wiki/Data_Encryption_Standard>.

WIKIPEDIA CONTRIBUTORS. Advanced Encryption Standard. Disponível em: <https://en.wikipedia.org/wiki/Advanced_Encryption_Standard>.

WIKIPEDIA CONTRIBUTORS. RC4. Disponível em: <<https://en.wikipedia.org/wiki/RC4>>.

WIKIPEDIA CONTRIBUTORS. RSA (cryptosystem). Disponível em: <[https://en.wikipedia.org/wiki/RSA_\(cryptosystem\)](https://en.wikipedia.org/wiki/RSA_(cryptosystem))>.