

Projeto Final

Segurança Computacional

Andre de Oliveira Melo - 222008252
Enzo Matschinski Kummer - 222005377

1 Introdução

Este projeto contempla a matéria estudada ao longo do semestre 2024/I em Segurança Computacional. O professor João José Costa Gondim ensinou-nos elementos importantes relacionados à criptografia simétrica e assimétrica, e durante este trabalho tivemos como colocar em prática os conhecimentos outrora adquiridos através da teoria.

A maior dificuldade que enfrentamos foi justamente a experiência na área de criptografia. Nenhum de nós havia tido a oportunidade de construir um algoritmo de cifração e decifração, pois sempre tivemos ao nosso alcance bibliotecas que já tinham isso pronto. Assim, ao realizarmos o trabalho, adquirimos uma experiência que outrora jamais possuiríamos.

Para a parte I do trabalho, pegamos como inspiração o código presente [aqui](#). Deste código retiramos a S-Box e a sua inversa. Além disso, tomamos como inspiração algumas das funções de manipulação.

Para executar a parte um, usa-se o terminal. O comando de inicialização (a ser executado dentro do diretório dos arquivos) deve ser: `python main.py <caminho para o arquivo> <número de rodadas>`. Já para a parte dois, basta rodar o arquivo `RSA.py`.

2 Parte I – Cifra de Bloco e modo de operação CTR

2.1 Descrição da cifra, modo de operação e operações implementadas

O projeto implementa a cifra Advanced Encryption Standard (AES) no modo de operação Counter (CTR), um padrão criptográfico amplamente utilizado para proteger dados confidenciais. O AES é uma cifra de bloco que, no modo CTR, é transformada em uma cifra de fluxo, permitindo a criptografia de dados de tamanho variável. No código, a chave de criptografia é expandida para gerar chaves de rodada, utilizando a substituição de bytes (SubBytes) via S-Box, seguida do deslocamento de linhas (ShiftRows) e da mistura de colunas (MixColumns) para difundir os dados. As chaves de rodada são aplicadas ao estado por meio da operação de XOR (AddRoundKey). O modo CTR utiliza um contador inicial (IV), que é criptografado para gerar um fluxo de chave. Este fluxo é então combinado com os blocos de texto original usando XOR, produzindo o texto cifrado. A descryptografia é realizada de maneira simétrica, aplicando a mesma operação XOR entre o fluxo de chave e o texto cifrado. O código implementa todas as transformações necessárias do AES, incluindo a expansão de chave, as operações de SubBytes, ShiftRows, MixColumns, e AddRoundKey, além da manipulação do contador para gerar o fluxo de chave no modo CTR, assegurando a segurança e integridade dos dados criptografados.

Durante a execução do programa, o usuário é solicitado a fornecer dois parâmetros essenciais: o caminho completo do arquivo a ser criptografado e o número desejado de rodadas para o processo de cifração. Uma vez iniciado o processo, o programa realiza a criptografia do arquivo especificado, gerando um novo arquivo com o prefixo "encrypted" no mesmo diretório do arquivo

original. Subsequentemente, o programa executa automaticamente a operação de descriptografia, criando um terceiro arquivo com o prefixo "decrypted", também no diretório original. Este processo permite a verificação imediata da integridade da cifragem e decifragem, assegurando que o conteúdo do arquivo decifrado corresponda exatamente ao do arquivo original.

2.2 Descrição das Implementações

A implementação da cifra AES com o modo de operação CTR é feita com base em uma série de transformações criptográficas que garantem a segurança dos dados. A chave de criptografia é expandida para gerar uma série de chaves de rodada usando a S-Box, que realiza a substituição de bytes (SubBytes). Em seguida, os bytes são reorganizados através do deslocamento de linhas (ShiftRows), e as colunas do estado são misturadas (MixColumns) para aumentar a difusão. Cada rodada de criptografia aplica uma das chaves de rodada ao estado usando a operação XOR (AddRoundKey). Essas transformações são fundamentais para assegurar que mesmo pequenas mudanças no texto original ou na chave resultem em uma saída completamente diferente, o que dificulta ataques criptográficos.

No modo de operação CTR, o AES é adaptado para funcionar como uma cifra de fluxo, o que é realizado pela criptografia de um contador inicial (IV) para gerar um fluxo de chave. Esse fluxo é então combinado com o texto original através de XOR para produzir o texto cifrado. A descriptografia no modo CTR é simétrica, ou seja, o mesmo processo é aplicado ao texto cifrado para recuperar o texto original. O contador é incrementado a cada bloco processado, garantindo que cada bloco de texto tenha um fluxo de chave único, essencial para a segurança no modo de cifra de fluxo. A implementação inclui a manipulação do contador e as operações de criptografia de blocos necessárias para assegurar que os dados sejam criptografados de forma eficiente e segura.

2.3 Resultado dos testes

2.3.1 Arquivo de texto

Cifrando o arquivo exemplo.txt com o contador 5, obtivemos o resultado:

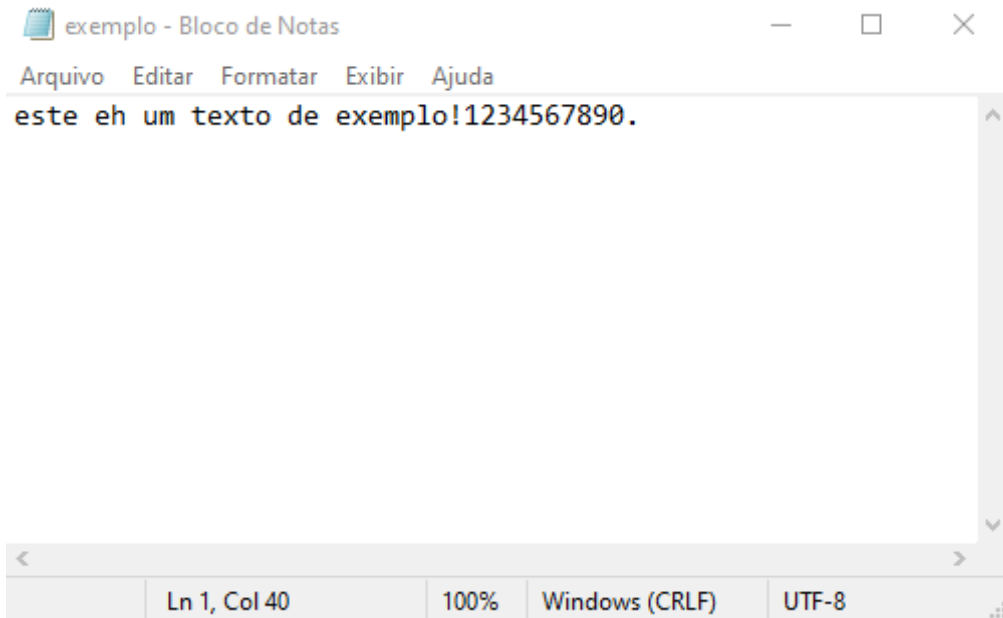


Figure 1: Início do arquivo exemplo.txt

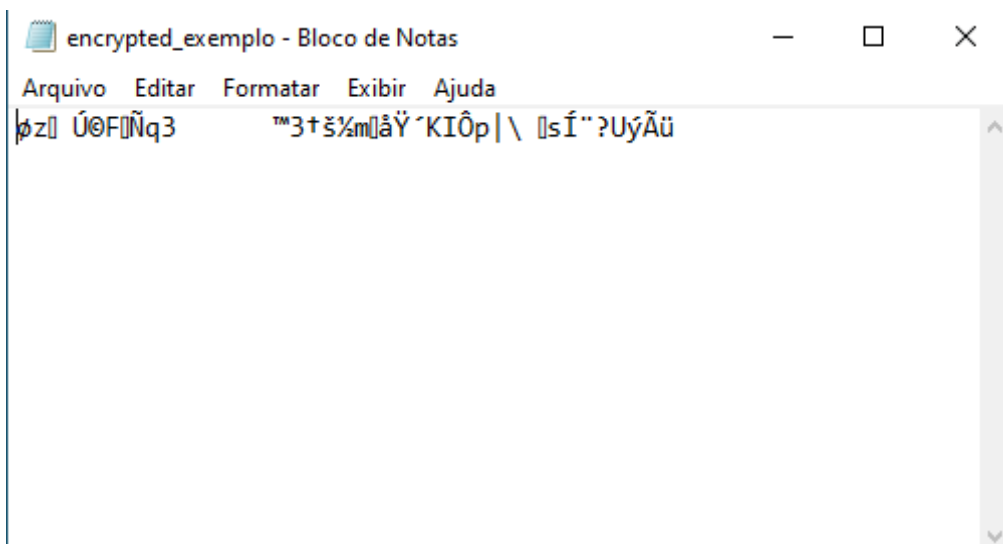


Figure 2: Arquivo criptografado com 5 rodadas

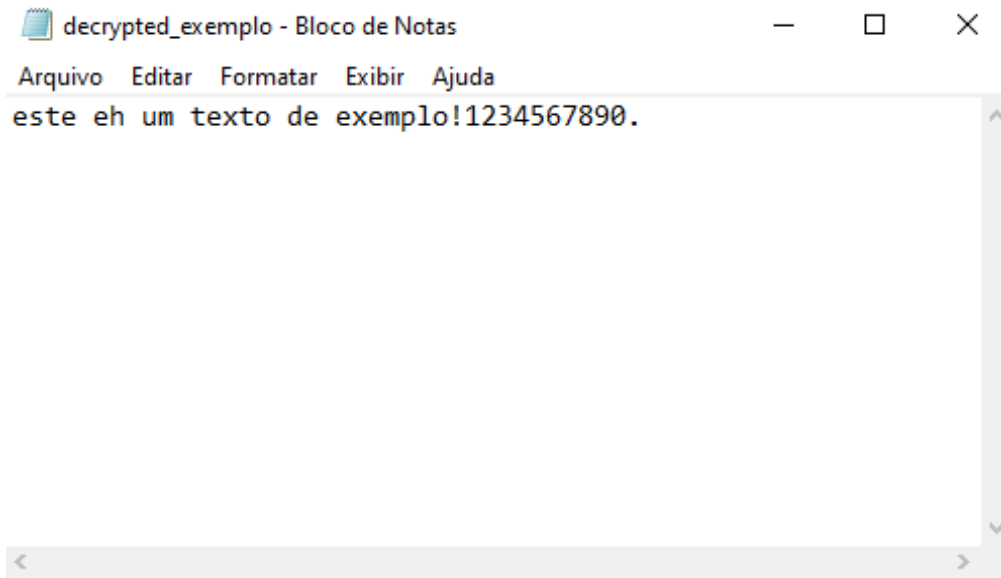


Figure 3: Arquivo descriptografado

As imagens demonstram que o processo de criptografia e descriptografia foi um sucesso. É fato que, com número de rodadas diferente, o conteúdo do arquivo criptografado vai ser diferente. Observemos esse comportamento para o caso de 10 rodadas.

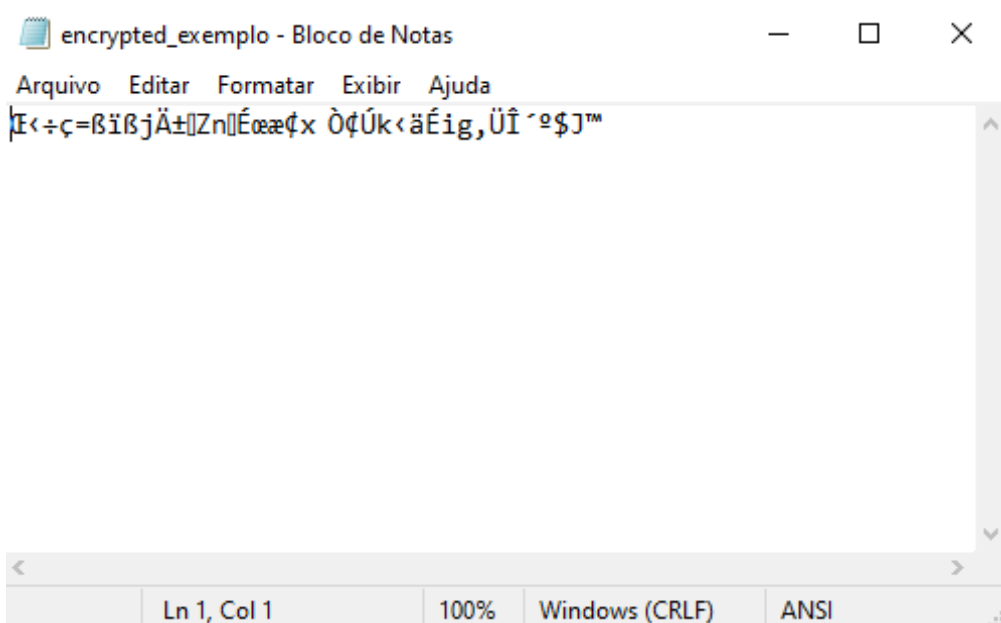


Figure 4: Arquivo criptografado com 10 rodadas

Outro ponto notável é que mesmo alterando muito pouco o conteúdo inicial, o conteúdo criptografado será completamente diferente (propriedade não-linear). Ao alterar o conteúdo do arquivo 'exemplo.txt' para "este eh um texto de exemplo!1234567899." (um algarismo trocou de 0 para 9),

mesmo assim o texto criptografado é muito diferente. Observe:

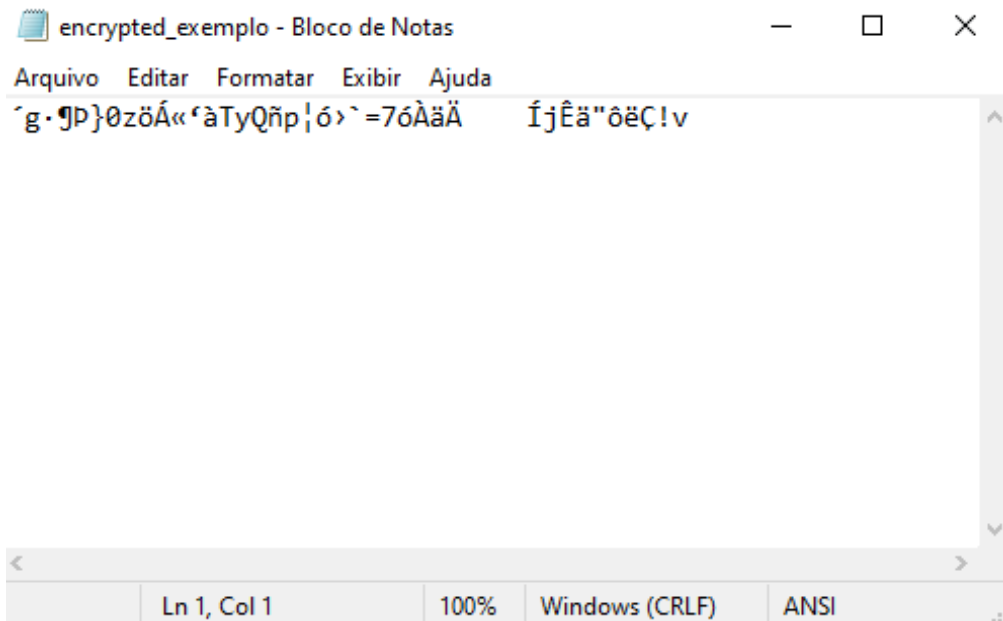


Figure 5: Arquivo criptografado com 10 rodadas

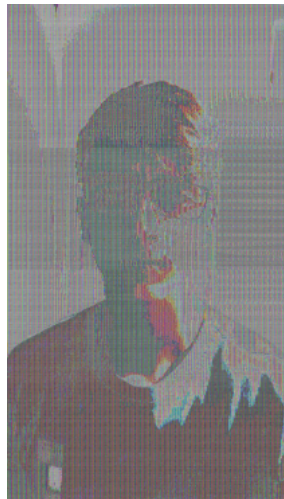
Assim sendo, o resultado dessa sessão se apresentou satisfatório.

2.3.2 Imagem

Podemos observar um fenômeno interessante ao renderizar os arquivos criptografados que correspondiam à selfie do aluno Enzo. Vamos aos testes:



Figure 6: Foto inicial



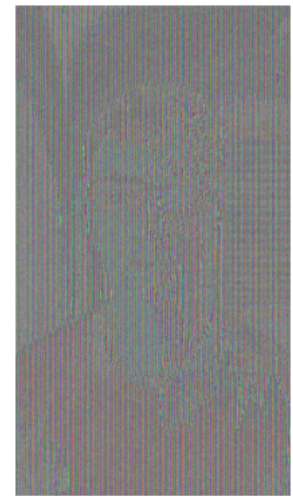
1 RODADA



5 RODADAS



9 RODADAS



13 RODADAS

Figure 7: Foto criptografada



Figure 8: Foto descriptografada

3 Parte II - Gerador/Verificador de Assinaturas

3.1 Descrição da cifra, modo de operação e operações implementadas

O projeto implementa a cifra RSA (Rivest-Shamir-Adleman) no modo de operação assimétrico, amplamente utilizado para proteger dados confidenciais. O RSA é uma cifra de chave pública que utiliza um par de chaves: uma chave pública para cifração e uma chave privada para decifração. A segurança do RSA baseia-se na dificuldade computacional de fatorar o produto de dois números primos grandes. O processo de cifragem e decifragem no RSA envolve operações básicas de geração de chaves, cifragem e decifragem. Na geração de chaves, são gerados dois números primos grandes, p e q , e calculado $n = p * q$. A função totiente de Euler $(n) = (p-1) * (q-1)$ é computada. Um expoente e , coprimo com (n) , é escolhido como chave pública, e d , o inverso multiplicativo de

e $\text{mod } (n)$, é calculado como chave privada. Para a cifragem, dada uma mensagem m , o processo é realizado como:

$$c = m^e \text{ mod } n$$

Na decifragem, para um texto cifrado c , o processo é realizado como:

$$m = c^d \text{ mod } n$$

O projeto implementa o padding OAEP (Optimal Asymmetric Encryption Padding) para aumentar a segurança da cifragem RSA. O OAEP é crucial por adicionar aleatoriedade à mensagem antes da cifragem, evitando que mensagens idênticas produzam o mesmo texto cifrado. Ele também protege contra ataques de texto conhecido e ataques de oráculo de padding, além de garantir que o RSA atinja a segurança semântica, onde um adversário não pode obter qualquer informação parcial sobre a mensagem a partir do texto cifrado. O processo OAEP envolve a geração de uma semente aleatória, o uso de funções de hash (SHA3-512 neste projeto) e uma função de geração de máscara (MGF1), além da aplicação de operações XOR para misturar a mensagem com dados derivados da semente. A implementação do OAEP segue o padrão PKCS1 v2.1, garantindo compatibilidade e segurança robusta.

A assinatura digital RSA é implementada para fornecer autenticidade e integridade às mensagens. O processo de assinatura digital envolve a geração de hash da mensagem, onde esta é processada por uma função de hash criptográfico (SHA3-512 neste projeto) para produzir um digest fixo. Em seguida, ocorre a cifragem do hash, onde o digest é cifrado com a chave privada do remetente, produzindo a assinatura digital. A verificação é realizada pelo receptor, que decifra a assinatura com a chave pública do remetente e compara o resultado com o hash da mensagem recebida. A implementação de assinatura inclui características importantes como o uso do esquema RSASSA-PSS (RSA Probabilistic Signature Scheme) para maior segurança, a implementação de salt aleatório no processo de assinatura para prevenir ataques de extensão de comprimento, e a verificação de tamanho de chave e parâmetros para garantir a força criptográfica adequada.

O projeto implementa todas estas operações necessárias do RSA, OAEP e assinatura digital, incluindo a geração de chaves, as operações de cifragem e decifragem, padding, assinatura e verificação, além da manipulação de grandes números para garantir a segurança do sistema. A implementação cuidadosa destes elementos assegura que o sistema seja resistente a uma variedade de ataques criptográficos conhecidos, proporcionando um alto nível de segurança para comunicações e autenticação de dados. Através da combinação destes componentes - RSA para criptografia assimétrica, OAEP para padding seguro, e assinatura digital para autenticação - o projeto oferece uma solução criptográfica abrangente e robusta, capaz de atender às demandas de segurança em diversas aplicações de comunicação e armazenamento de dados sensíveis.

3.2 Descrição das Implementações

A implementação da cifra RSA com o modo de operação assimétrico é feita com uma série de transformações criptográficas que garantem a segurança dos dados. A chave de criptografia é expandida para um tamanho adequado (mínimo de 1024 bits) usando o teste de primalidade de Miller-Rabin com $k = 40$ para garantir a geração de primos fortes.

O padding OAEP é implementado usando a função de hash SHA3-512 e uma função de geração de máscara (MGF1). Isso adiciona aleatoriedade e dificulta ataques de texto conhecido.

Para a assinatura digital, o hash da mensagem é calculado usando SHA3-512 e então cifrado com a chave privada. A verificação é realizada decifrando a assinatura com a chave pública e comparando o resultado com o hash da mensagem original.

As operações com arquivos são implementadas de forma a processar o arquivo em blocos, permitindo o tratamento de arquivos de tamanho arbitrário. A cifragem, decifragem, assinatura e verificação de arquivos seguem os mesmos princípios das operações com mensagens, mas aplicados a cada bloco do arquivo.

Essas implementações são fundamentais para assegurar que mesmo pequenas mudanças no texto original ou na chave resultem em uma saída completamente diferente, o que dificulta ataques criptográficos.

3.3 Resultado dos testes

```
Digite o número da opção desejada: 1
Digite a mensagem a ser cifrada: teste
Mensagem cifrada: 1272885341716069845033

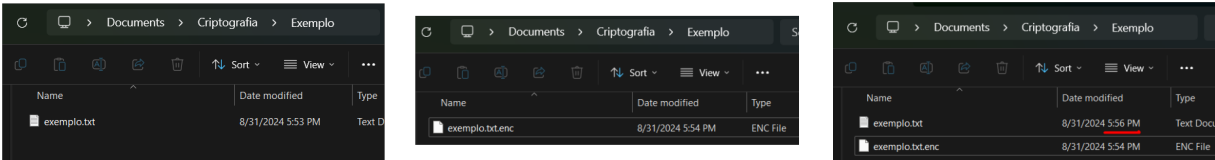
Digite o número da opção desejada: 2
Digite o texto cifrado (número inteiro): 12728
Mensagem decifrada: teste

Digite o número da opção desejada: 3
Digite a mensagem a ser assinada: 12728853417
Mensagem assinada:
-----BEGIN SIGNED MESSAGE-----
MTI3Mjg4NTM0MTcxNjA2OTg0NTAzMzE0ODE5NzM4NjY5MjY5NTM2
-----BEGIN SIGNATURE-----
RI3sAucoV69RsUaShYUEaGCKYH70UizoVT5AnLC9hEuW1rvk/Wv+
-----END SIGNED MESSAGE-----

Digite o número da opção desejada: 4
Cole a mensagem assinada (incluindo cabeçalhos). Pressione Enter para continuar.
-----BEGIN SIGNED MESSAGE-----
MTI3Mjg4NTM0MTcxNjA2OTg0NTAzMzE0ODE5NzM4NjY5MjY5NTM2
-----BEGIN SIGNATURE-----
RI3sAucoV69RsUaShYUEaGCKYH70UizoVT5AnLC9hEuW1rvk/Wv+
-----END SIGNED MESSAGE-----
Assinatura válida!
Mensagem original: 1272885341716069845033148197386692

Digite o número da opção desejada: 5
Digite o caminho do arquivo a ser cifrado: C:\Users\andre\Documents\Criptografia\Exemplo\exemplo.txt
Arquivo cifrado salvo em: C:\Users\andre\Documents\Criptografia\Exemplo\exemplo.txt.enc

Digite o número da opção desejada: 6
Digite o caminho do arquivo a ser decifrado: C:\Users\andre\Documents\Criptografia\Exemplo\exemplo.txt.enc
Arquivo decifrado salvo em: C:\Users\andre\Documents\Criptografia\Exemplo\exemplo.txt
```



4 Conclusão

Pode-se concluir que, dentre o proposto pelo professor, o projeto foi concluído com êxito e o resultado apresentado é satisfatório. Quanto a Parte I, foram implementados os elementos obrigatórios e também o Extra 2, onde com 1, 5 e 9 rodadas, o conteúdo da foto, mesmo criptografado, ainda pode ser reconhecido. Porém, com 13 rodadas, a criptografia esconde muito melhor as informações originais da foto, afinal a sugestão para o número mínimo de rodadas é 10. A parte II também foi executada corretamente e todos os pontos obrigatórios foram contemplados.