211036141 - Cássio Vinícius Teixeira Borges

Github: https://github.com/Cassio-hue/sc-projeto/

Implementação

Foi utilizado a linguagem **Python** na sua **versão 3**.
Bibliotecas utilizadas: **sympy**, **random**, **hashlib**, **base64**.

AES

A cifra Advanced Encryption Standard (AES) é uma cifra de bloco amplamente utilizada que opera em blocos de 128 bits, com chaves de **128, 192 ou 256 bits**. AES é uma cifra simétrica, ou seja, a chave é usada tanto para cifrar quanto para decifrar os dados. Ela utiliza várias etapas como substituição de bytes, deslocamento de linhas, mistura de colunas e adição de chave para transformar o texto original em texto cifrado de forma segura.

OBS: Em específico, para este projeto foi utilizada chave com 128 bits.

O modo CTR transforma uma cifra de bloco em uma cifra de fluxo, permitindo cifrar blocos de tamanho variável. Nesse modo, um contador (counter) é incrementado para cada bloco, que é então cifrado usando AES. O resultado é combinado com o bloco de texto original usando a operação XOR, produzindo o texto cifrado. A mesma operação é usada para a decifragem, tornando o CTR eficiente e adequado para sistemas que requerem alta performance.

AES Cifração/Decifração

- **Expansão da Chave**: A função "*expand_key*" expande a chave original para várias subchaves usadas em cada rodada da cifragem.
- **Substituição de Bytes**: Faz-se uso da *S-Box* (*constants.py*) para substituir cada byte da matriz de estado, garantindo confusão.
- **Rotação de linhas**: Rotaciona as linhas da matriz, contribuindo para a difusão dos dados.
- **Embaralhamento de colunas**: Mistura as colunas da matriz, adicionando

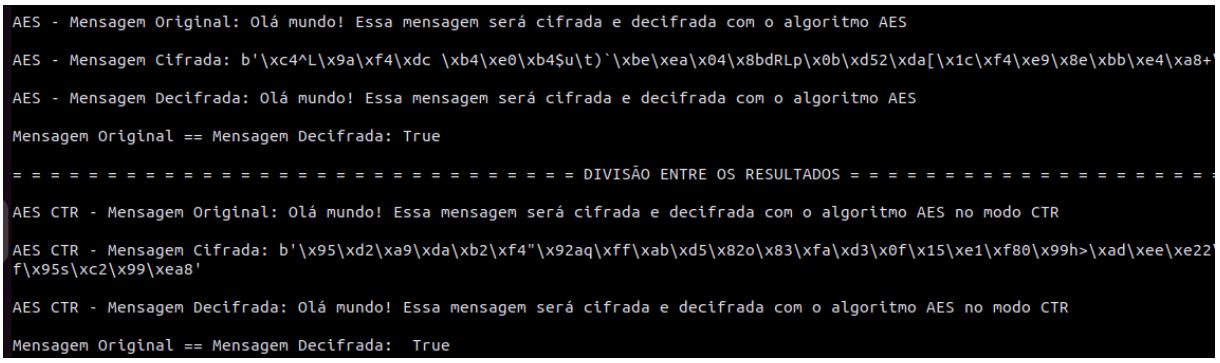
complexidade ao processo.

- **Adição de chave**: É realizado uma operação XOR entre a matriz de estado e a subchave correspondente.
- **Inverso de substituição de Bytes**: A função “reverse_lookup” utiliza a S-Box inversa para reverter a substituição de bytes aplicada durante a cifragem.
- **Inverso de rotação de linhas:** As linhas da matriz são rotacionadas na direção oposta ao processo de cifragem, desfazendo a difusão aplicada.
- **Inverso de embaralhamento de colunas:** A mistura das colunas é desfeita aplicando várias operações de mistura na matriz de estado, revertendo a complexidade adicionada na cifragem.

AES-CTR Cifração/Decifração

- **Incremento do Contador:** O contador é incrementado para cada bloco processado usando a função “*increment_counter*”, e o processo é repetido para o próximo bloco até que todos os blocos tenham sido processados.
- **Divisão em Blocos:** O texto original é dividido em blocos de 16 bytes, já que o AES opera em blocos fixos de 128 bits. Se o último bloco for menor que 16 bytes, ele não precisa de padding (preenchimento), como ocorre em outros modos de operação.
- **Nonce e Contador:** Um valor único (nonce) de 16 bytes é combinado com um contador, criando um valor único para cada bloco. Esse contador é incrementado a cada novo bloco processado. O valor do nonce permanece constante ao longo da execução, enquanto o contador é incrementado em cada iteração.
- **Cifragem do Nonce e Contador:** O valor nonce+contador é cifrado usando AES, gerando um "bloco de chave" que será utilizado no próximo passo. Essa cifragem é realizada através da função “*aes_encrypt*”.
- **XOR com o Texto Original/Cifrado:** O bloco de texto original (plaintext) é combinado com o bloco de chave resultante da cifragem do nonce+contador, usando a operação XOR “*xor_bytes*”. O resultado desta operação é o texto cifrado. Para a **decifragem**, o processo é exatamente o mesmo: o texto cifrado é novamente combinado (usando XOR) com o bloco de chave gerado a partir do nonce + contador, restaurando assim o texto original.

Resultados AES/AES-CTR



RSA

O RSA (Rivest-Shamir-Adleman) é um dos algoritmos de criptografia assimétrica mais amplamente utilizados para transmissão de dados. Ele baseia sua segurança na dificuldade de fatorar números grandes, sendo especialmente útil para cifragem de dados e assinaturas digitais. No RSA, cada usuário possui um par de chaves:

- Chave pública: usada para cifrar mensagens
- Chave privada: usada para decifrá-las

A segurança do RSA depende do uso de números primos grandes e de operações modulares com essas chaves.

Optimal Asymmetric Encryption Padding (OAEP)

O OAEP é um esquema de preenchimento utilizado em conjunto com o RSA para aumentar a segurança do processo de cifragem. Ele adiciona aleatoriedade ao processo de cifragem, tornando-o seguro contra ataques de texto cifrado escolhido.

No processo de codificação OAEP, a mensagem é combinada com uma semente (seed) aleatória e, em seguida, processada por uma função geradora de máscara (MGF1), resultando em uma versão "preenchida" da mensagem. Este preenchimento (padding) garante que a mensagem cifrada final seja única, mesmo se a mesma mensagem original for cifrada múltiplas vezes com a mesma chave pública.

RSA-OAEP Cifração/Decifração

O código utiliza a função “*generate_keys*” para gerar as chaves pública e privada. Dois números primos **p** e **q** são gerados utilizando a função “*gen_prime*”, e o produto desses números é usado como parte das chaves.

A função “*rsa_encrypt*” cifra uma mensagem utilizando a chave pública e o esquema OAEP. A mensagem é codificada usando “*oaep_encode*”, que aplica um preenchimento seguro, antes de realizar a cifração com a operação RSA.

A decifragem é realizada pela função “*rsa_decrypt*”, que primeiro reverte a operação RSA e depois remove o preenchimento OAEP usando “*oaep_decode*”.

Assinatura

A assinatura digital RSA permite que o remetente de uma mensagem prove sua identidade e a integridade da mensagem para o destinatário. No processo de assinatura, o remetente cria um hash da mensagem e, em seguida, cifra este hash com sua chave privada, gerando a assinatura digital.

O destinatário pode verificar a assinatura cifrando-a com a chave pública do remetente e comparando o resultado com o hash da mensagem recebida. Se os valores coincidirem, a assinatura é válida, garantindo que a mensagem não foi alterada e que foi enviada por quem alega tê-la enviado.

O código inclui uma função para assinar arquivos, *“sign_file*”, que primeiro calcula o hash do arquivo usando SHA-256 e, em seguida, cifra esse hash usando a chave privada do RSA para gerar a assinatura. Para verificar a assinatura, existe a função “*verify_file*”, que decifra a assinatura com a chave pública e compara o resultado com o hash do arquivo. Se os valores coincidem, a assinatura é considerada válida.

Resultado RSA

O arquivo utilizado se chama documento.txt e se encontra no repositório.

```
Public Key: (65537, 22929642264050942655977338935922361027969082142887122668469323668717484125507574010102979601345279649648558197
45697522833297354670858737809932551680433993460065693318110122281328571465138027216718197544789706544222153400115302324573632236721
25951477814407003320193095517897475123518254045565175116525851241775278698512298477048 3)

Private Key: (18410329675364459809840663667977396543810871757308396704378531386055419300306827355716904750336277448838169772353564
51032059856306828002251531972312196724535280174046045282588051518038230977301763123529086435721733781746159055178039721380010802345
27517813528412605630113278161678442346857276190203594117601220092386531951997553, 22929642264050942655977338935922361027969082142881
23867357770224672135967742067370945936004771260869872874457354456975228332973546708587378099325516804339934600656933181101222813285
64318767115964426453302778525357067494424844515563513368359424259514778144070033201930955178974751235182540455651751165258512417752
```

```
Mensagem original: Mensagem a ser cifrada usando RSA

Mensagem cifrada: 33906788163694902985357536114950948044904817925599490900146509632143553187129932599057020309679453851977850542026
91397152595330743411557314909780317235750925506336378465949091336585886972902079372729979091601303576339935738582790756020317141627
39534400292771479048096286391001057251773853999075621408579875926945286322869 4835416

Mensagem decifrada: Mensagem a ser cifrada usando RSA

Mensagem Original == Mensagem Decifrada: True


Assinatura em Base64:
X2hDbO0sSktX78yEm1Uw2YSNnqb0rD/PSuqueyuOscQxIymya7wmhNgIGRIOaxgWb3XrSFBbt1GXMFk4OrM4xy4Y3AmbGag4qp5DXNyxJpcGXqUDlylCsZtjiaoRobtwQnv
4IqgHJxOVsUn8UUaSmAerL6bkGfi/XvSkw7jsZ1I3F+VWDoeErh1HVGMsiBb2AVsuRA==
```