Universidade de Brasília

Departamento de Ciência da computação

Trabalho Final de Segurança Computacional

Autores:

Gabriel Brito de França 211020867

Henrique Valente Lima 211055380

Brasília

8 de julho de 2023

Conteúdo

1 Main

1.1 Menu de opções:

- 1 - A primeira é o gerador de chaves, o qual gera as chaves públicas e privadas gerando dois números primos aleatórios, multiplicando-os para gerar 'n' (primeira parte de ambas as chaves), após isso, será gerado um número 'e' que será a segunda parte da chave pública e um número 'd' que será a segunda parte da chave privada.

- 2 – Se houver chaves já geradas nos arquivos e não sendo necessária a geração de chaves novas, essa opção lê as chaves dos arquivos (explicação detalhada logo mais) (as chaves guardadas nos arquivos estão cifradas, favor não guardar não geradas pelo programa).

- 3 - geração de chaves em AES, cifração do arquivo em AES, adicionar o arquivo .txt em '/archives' e inserir o nome, a assinatura que fica salva em '/archives/signature.txt' e a session key cipher (chave cifrada da sessão) em 'archives/`session_key_cypher.txt`'.

- 4 - Decifração do arquivo em AES, verifica a assinatura e a chave da sessão e decifra o texto no caminho dado.

- 5 - Finaliza a execução do programa

1.2 Execução

Para tornar o programa realmente funcional para dois ou mais usuários, seria necessário cada um gerar suas chaves, e trocar entre si as chaves públicas.

Para cifrar: salvar a chave pública do destinatário no caminho "archives/`public_key.txt`"e manter sua chave privada no caminho "archives/`private_key.txt`", assim o texto será cifrado de forma que a chave privada do destinatário consiga decifrar e a chave pública do remetente possa confirmar a origem da mensagem, o texto cifrado será salvo no mesmo arquivo que foi dado, a assinatura e a chave cifrada da sessão serão armazenadas nos arquivos associados.

Para decifrar: o destinatário ao receber o arquivo cifrado, a assinatura e a chave cifrada da sessão, deve salvar a assinatura e a chave cifrada da sessão nos caminhos associados de cada um e salvar o arquivo cifrado na pasta '/archives', salvar a própria chave privada no arquivo associado e a chave pública do remetente da mensagem no arquivo associado a chave pública, após isso, associar o caminho do arquivo a ser decifrado, após a decifração o próprio arquivo apresentará a mensagem decifrada, após isso os arquivos da assinatura e da chave da sessão não serão limpos, opcional apagar o conteúdo.

* Salve suas chaves públicas e privadas em outro lugar, principalmente a pública que será substituída para comportar a chave pública do remententе/destinatário da mensagem.

2 Implementação do AES em CTR

2.1 Expansão da chave e procedimento inicial

- FUNÇÃO CTR - O código se inicia com a função ctr onde recebe a mensagem, chave e o vetor de inicialização. nela o programa divide a mensagem em blocos de 128 bits, expande a chave e cria o counter com o IV. A partir daí ela gera as cifras com o IV e as chaves ao chamar a função cipher. Após isso, o bloco de mensagem e essas cifras são mapeadas com a função addRoundkey formando assim o texto cifrado.

- FUNÇÃO PARA EXPANDIR CHAVE – Essa função recebe a chave de criptografia e retorna a chave expandida, que consiste em uma lista de subchaves para cada rodada do AES. Essa função segue o algoritmo de expansão de chave do AES, que envolve rotações, substituições de bytes e operações XOR.

- DETALHAMENTO DA FUNÇÃO - Na função de expansão da chave a chave é convertida em lista de palavras. O loop começa a partir da quarta palavra, pois as três primeiras palavras são copiadas diretamente da chave original. Para gerar as próximas palavras, o algoritmo segue a seguinte lógica: Se o índice i for múltiplo de 4, é aplicada a operação

  de substituição de bytes (translate(SBOX)) e o resultado é rotacionado (rotate(temp)).

Em seguida, o resultado é combinado com uma palavra da tabela RCON (RCON[i//4]) usando a operação XOR. Isso adiciona uma variação única para cada rodada.

As palavras resultantes são calculadas usando a operação XOR entre a palavra anterior (temp) e a palavra quatro posições anteriores (words[i-4]).

As palavras geradas são adicionadas à lista de palavras words.

As palavras geradas são agrupadas novamente em blocos de 16 bytes, que representam as subchaves a serem usadas nas rodadas do AES. No

final da função, as subchaves são retornadas como uma lista de blocos de 16 bytes.

2.2 Criação e operação no "state"e implementação das rodadas

- RODADA INCIAL - A chave de rodada inicial (keys[0]) é combinada com o bloco de entrada (block) por meio de uma operação XOR chamada addRoundKey. Isso adiciona a chave de rodada ao estado inicial.

- OPERAÇÕES NO ESTADO são executadas 10 rodadas, numeradas de 1 a 10 (range(1, 11)).

  Substituição de bytes (subBytes): Nesta etapa, cada byte do estado é substituído por um valor correspondente da S-box (tabela de substituição).

  Deslocamento de linhas (shiftRows): Nesta etapa, as linhas do estado são deslocadas circularmente. A primeira linha permanece inalterada, a segunda linha é deslocada uma posição à esquerda, a terceira linha é deslocada duas posições à esquerda e a quarta linha é deslocada três posições à esquerda.

  Mistura de colunas (mixColumns): Nesta etapa, as colunas do estado são misturadas por meio de operações de multiplicação e adição modular.

Adição da chave de rodada (addRoundKey): A chave de rodada correspondente (keys[round]) é combinada com o estado atual por meio de uma operação XOR.

3 RSA

3.1 Geração de chaves

- No RSA as chaves são geradas desta maneira:

  1. Escolha de forma aleatória dois números primos grandes p e q, da ordem de 10^{100} no mínimo.
  2. Calcule $n = p \cdot q$.
  3. Calcule a função totiente de Euler, $\phi(n) = (p-1) \cdot (q-1)$.
  4. Escolha um inteiro e, tal que $1 < e < \phi(n)$, de forma que e e $\phi(n)$ sejam relativamente primos entre si (não há número maior que 1 que divida ambos).
  5. Calcule d, de forma que d seja o inverso multiplicativo de e (algoritmo de Euclides).
     Por fim, temos:

     A chave pública: o par (n, e).

     A chave privada: o par (n, d) ou o trio (p, q, d) (no nosso caso apenas o primeiro par).

3.2 Assinatura Digital:

O algoritmo RSA é extensível a este contexto, pelas suas propriedades. Para implementar um sistema de assinaturas digitais com RSA, o utilizador que possua uma chave privada d poderá assinar uma dada mensagem com a seguinte expressão:

$s = m^d mod n$

É difícil descobrir 's' sem o conhecimento de 'd'. Portanto, uma assinatura digital definida conforme esta equação é difícil de forjar. Mas o emissor de 'm' não pode negar tê-la emitido, já que mais ninguém poderia ter criado tal assinatura. O receptor recupera a mensagem utilizando a chave pública e do emissor:

$s^e = (m^d)^e mod n = m mod n$

Como tal, o receptor consegue validar a assinatura do emissor calculando ($s^e mod n$).

Podemos verificar então que o algoritmo RSA satisfaz os três requisitos necessários de uma assinatura digital.

4 Implementação do OAEP

O optimal assymetric encryption padding (OAEP), ou preenchimento de criptografia assimétrica ideal, foi introduzido em 1994 por Bellare e Rogaway combinado ao RSA, esse método garante que o adversário não decifre a mensagem sem realmente saber a chave privada. O OAEP adiciona uma aleatoriedade a um esquema determinística (RSA tradicional) e o transforma em um modelo probabilístico.

No nosso código, ele foi usado para cifrar a chave da sessão (session's key) e a chave cifrada da sessão (session's key cipher) a qual é guardada em arquivo e usada para decifrar a mensagem junto a assinatura, para decifrar a mensagem torna-se obrigatório o uso da chave da sessão gerada na cifração da mensagem, junto a assinatura, a chave pública do remetente para a verficação da assinatura e a chave privada do destinatário, para decifrar a chave da sessão para pegar a chave do AES para decifrar a mensagem.