Trabalho 2 - Gerador/Verificador de Assinatura

Gustavo Dezan Felipe Dantas Borges

10 de julho de 2023

Universidade de Brasília
Segurança Computacional - CIC0201
gustavol.dezan@gmail.com felipedbbsb@gmail.com

1 Visão geral

O trabalho foi implementado utilizando a linguagem Python . O código consiste de 4 módulos: "AES.py", "rsa.py", "oaep.py" e main.py". O objetivo era implementar um gerador e verificador de assinaturas RSA em arquivos.

1.1 AES

Neste módulo, está presente a implementação do algoritmo AES, uma cifra de bloco simétrica. A versão implementada do AES utiliza chaves criptográficas de 128 bits e realiza 10 rodadas para criptografar e descriptografar dados em blocos de 128 bits. Cada rodada é composta por quatro operações: AddRoundKey, SubBytes, ShiftRows e MixColumns.

1. AddRoundKey: Combinação dos dados com a chave de criptografia através de uma operação XOR.
2. SubBytes: Substituição não linear de cada byte do bloco de dados usando uma tabela de substituição.
3. ShiftRows: Deslocamento das linhas do bloco de dados para espalhar os bytes.
4. MixColumns: Transformação linear de cada coluna do bloco de dados para aumentar a difusão.

Essas operações são aplicadas em várias rodadas, exceto MixColumns na última rodada. Juntas, essas etapas garantem a segurança e confidencialidade dos dados criptografados pelo AES.

Além disso, nesta implementação, é utilizado o modo de operação chamado Galois Counter Mode (GCM), que é um modo utilizado em criptografia simétrica. O GCM utiliza uma representação especial de um grupo multiplicativo associado a uma curva elíptica. Esse modo de operação combina a autenticação de mensagem com a criptografia, permitindo alcançar tanto confidencialidade quanto integridade dos dados.

1.2 RSA

No algoritmo RSA, é essencial gerar chaves seguras e robustas. Isso envolve a geração de dois números primos grandes, p e q, com um mínimo de 1024 bits cada. A segurança do RSA é baseada na dificuldade de fatorar esses números primos.

Para garantir que os números gerados sejam primos, é comum utilizar o teste de primalidade de Miller-Rabin. Esse teste probabilístico permite determinar se um número é composto ou provavelmente primo.

O teste de Miller-Rabin envolve a escolha aleatória de uma base e a aplicação de um conjunto de testes ao número em questão. Se o número passar em todos os testes, ele é considerado provavelmente primo. Caso contrário, é definitivamente composto.

Ao repetir o teste com várias bases diferentes, aumentamos a confiabilidade do resultado. Quanto mais iterações forem realizadas, menor será a probabilidade de um número composto ser erroneamente identificado como primo.

Após a geração bem-sucedida dos números primos p e q, as chaves pública e privada podem ser calculadas usando esses valores.

1.3 OAEP

O OAEP, ou Preenchimento de Criptografia Assimétrica Ótima, é um padrão de preenchimento utilizado em conjunto com o RSA para aumentar a segurança das operações de criptografia.

O objetivo do OAEP é adicionar aleatoriedade e proteção contra ataques de criptoanálise aos dados antes da criptografia. Ele usa funções hash, como o SHA-3, para realizar o preenchimento.

O processo de preenchimento do OAEP envolve duas etapas principais: preenchimento e aleatorização. Durante o preenchimento, são adicionados bits extras aos dados originais para torná-los mais seguros contra ataques de força bruta e análise de padrões. Em seguida, a aleatorização adiciona um valor aleatório aos dados para garantir que diferentes mensagens criptografadas resultem em saídas diferentes.

O OAEP é amplamente utilizado para fortalecer a segurança do RSA, adicionando uma camada adicional de proteção aos dados antes da criptografia.

No processo de cifração assimétrica RSA, o OAEP é usado em conjunto com o algoritmo RSA para criptografar e descriptografar mensagens.

Para criptografar uma mensagem usando RSA e OAEP, o remetente primeiro aplica o preenchimento OAEP à mensagem original. Em seguida, ele usa a chave pública do destinatário para realizar a criptografia RSA dos dados preenchidos.

Ao receber a mensagem criptografada, o destinatário aplica a operação de descriptografia RSA usando sua chave privada correspondente. Em seguida, ele reverte o processo de preenchimento OAEP para obter a mensagem original.

Esse processo garante que a mensagem só possa ser descriptografada pelo destinatário correto, que possui a chave privada correspondente à chave pública usada na criptografia.

A combinação do RSA com o OAEP fornece uma criptografia assimétrica forte e confiável, adequada para proteger a comunicação e os dados sensíveis.

1.4 MAIN

Neste módulo, estamos simulando uma transmissão segura de dados usando os algoritmos AES e RSA. Vamos descrever a situação simulada:

No início, o remetente gera uma mensagem e as chaves para os algoritmos AES e RSA. Em seguida, a mensagem é criptografada usando o AES. Além disso, a chave usada na criptografia AES e o hash da mensagem são assinados com o RSA.

O remetente envia os seguintes dados para o destinatário: a chave pública, a mensagem cifrada com AES, a chave da criptografia AES assinada com RSA e o hash da mensagem assinado com RSA.

O destinatário recebe os dados enviados e usa o RSA para recuperar a chave de criptografia AES. Em seguida, ele decifra a mensagem usando o AES com a chave recuperada.

Para verificar a integridade da mensagem, o destinatário usa o RSA para recuperar o hash original da mensagem. Ele então calcula um novo hash para a mensagem decifrada e verifica se o hash calculado coincide com o hash recuperado.

Essa simulação demonstra como ocorre uma transmissão segura de dados, onde a mensagem é criptografada com o AES para garantir a confidencialidade, e a integridade é verificada usando a assinatura digital RSA. A combinação desses algoritmos proporciona um ambiente seguro para a comunicação entre o remetente e o destinatário.

A seguir o resultado dos executáveis:

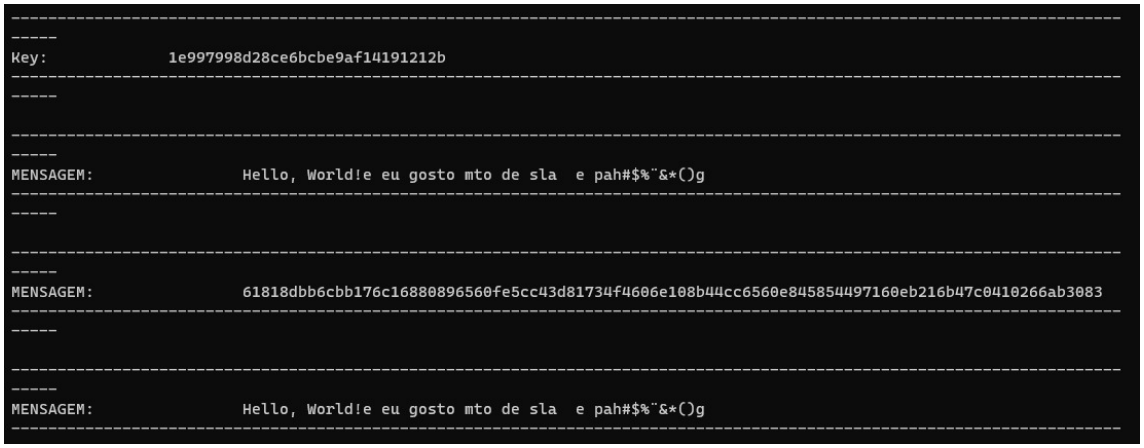

Figura 1: executável da AES

Figura 2: executável da main