Trabalho de Implementação 3 – Gerador/Verificador de Assinaturas
2023/2

Alan Fontenele Véras – 200013335

Welder Cavalcante de Oliveira – 190039582

Dep. Ciência da Computação – Universidade de Brasília (UnB)

1. Introdução

Este relatório documenta a elaboração do Gerador/Verificador de Assinaturas RSA em arquivos, parte do Trabalho de Implementação 3 da disciplina CIC0201 - Segurança Computacional. Dividido em três partes, o projeto envolve a geração de chaves e cifração/decifração RSA com OAEP, assinatura da mensagem utilizando SHA-3 e verificação com BASE64.

Seguindo as diretrizes estabelecidas, utilizei bibliotecas específicas, evitando OpenSSL. A avaliação considerará a implementação de primitivas como geração de chaves, cifração RSA, OAEP e formatação/parsing. Desenvolvimento em Python. O relatório oferece uma visão geral do processo, destacando aspectos técnicos do desenvolvimento. Parte I: Geração de chaves e cifra

2. Descrição e implementação

Parte I: Geração de chaves e cifra

O código apresenta uma implementação abrangente para a geração de chaves RSA e o processo de cifração utilizando o esquema de padding OAEP (Optimal Asymmetric Encryption Padding). O módulo miller_rabin_test é empregado para verificar a primalidade de números, essencial na escolha de primos para a geração de chaves. A função gerar_chave_prima utiliza o teste de Miller-Rabin para garantir a obtenção de números primos. A função gerar_e é responsável por gerar um expoente 'e' coprimo em relação ao totiente 't' e 'n'.

A geração de chaves RSA é efetuada pela função gerar_chave_rsa, que inicialmente obtém dois números primos, 'p' e 'q'. Em seguida, calcula o produto 'n' e o totiente de 'n' ('phi_n'). A função gerar_e é utilizada para escolher um expoente público 'e', e o expoente privado 'd' é calculado pela função calcular_expoente_privado. As chaves pública e privada são retornadas.

Para a cifração de uma mensagem, a função cifrar é empregada. Primeiramente, a mensagem é padronizada usando o padding OAEP através da função pad_message. Posteriormente, a mensagem é convertida para um inteiro e elevada à potência do

expoente público 'e', módulo 'n'. O resultado é convertido de volta para bytes e retornado como a mensagem cifrada.

Essa implementação se destaca por sua abordagem completa na geração de chaves e cifração, seguindo boas práticas criptográficas, como a escolha cuidadosa de números primos e o uso do padding OAEP para aumentar a segurança do processo de cifração.

Parte II: Assinatura

Na implementação da assinatura (Parte II), o código utiliza a função de hash SHA-3 (SHA3-256) para calcular o hash da mensagem original. A função assinar_mensagem recebe a mensagem original e a chave privada RSA como entrada. Inicialmente, é calculado o hash da mensagem utilizando SHA-3, resultando em uma sequência de bytes.

A chave privada é desempacotada, e o hash da mensagem é convertido para um número inteiro. Em seguida, a assinatura é calculada elevando o hash à potência do expoente privado 'd', módulo 'n'. O resultado é convertido de volta para bytes e retornado como a assinatura da mensagem.

Parte III: Verificação

Na Parte III, destinada à verificação, o código se concentra em assegurar a autenticidade e integridade das mensagens assinadas. O processo é dividido em etapas distintas.

A função verificar_assinatura é responsável por calcular o hash da mensagem original utilizando a função de hash SHA-3 (SHA3-256). Em seguida, a chave pública RSA é desempacotada, e a assinatura fornecida é convertida de bytes para um número inteiro.

O valor esperado da assinatura é então calculado elevando a assinatura à potência do expoente público 'e', módulo 'n'. O resultado é comparado com o hash da mensagem original. Se os valores coincidirem, a assinatura é considerada válida, indicando que a mensagem não foi alterada e foi realmente assinada pela entidade correspondente à chave privada associada à chave pública fornecida.

A implementação segue princípios essenciais de verificação de assinatura, garantindo que apenas a entidade detentora da chave privada associada à chave pública utilizada pode produzir uma assinatura válida. Essa abordagem reforça a segurança do sistema, oferecendo confiança na origem e integridade das mensagens assinadas.

3. Conclusão

Em conclusão, o desenvolvimento do Gerador/Verificador de Assinaturas RSA em arquivos foi um desafio que demandou esforço e dedicação. Ao longo deste processo, buscamos implementar as primitivas necessárias, enfrentando obstáculos e explorando soluções. Apesar de ainda não ter alcançado o resultado desejado pois tenha faltado a verificação de forma completa da assinatura, este trabalho representou uma oportunidade valiosa para o aprimoramento de habilidades em programação e

compreensão aprofundada de conceitos criptográficos. Aonde conseguimos compreender a geração e verificação de assinatura, junto a RSA e OAEP. O enfrentamento das dificuldades encontradas proporcionou aprendizado, contribuindo para nosso desenvolvimento como estudantes e aspirantes a profissionais na área de Segurança Computacional.

Referências

1. CONTRIBUIDORES DOS PROJETOS DA WIKIMEDIA. **RSA (sistema criptográfico)**. Disponível em: <https://pt.wikipedia.org/wiki/RSA_(sistema_criptogr%C3%A1fico)>.
2. **Optimal asymmetric encryption padding**. Disponível em: <https://en.wikipedia.org/wiki/Optimal_asymmetric_encryption_padding>.