Trabalho de Implementação 2
Gerador e Verificador de Assinaturas RSA

Vinícius Dos Santos Tomé - 232006780
232006780@aluno.unb.br

7 de outubro de 2025

Sumário

1 Introdução

No contexto da Segurança Computacional, a garantia de que a informação digital não foi alterada (integridade) e que provém de uma fonte legítima (autenticidade) é fundamental. As assinaturas digitais surgem como uma das principais ferramentas criptográficas para prover estes serviços de segurança, sendo essenciais em transações financeiras, comunicações seguras e na distribuição de software. Este trabalho aborda a implementação prática de um sistema de geração e verificação de assinaturas digitais baseado no algoritmo de chave pública RSA [1].

O objetivo deste projeto é desenvolver uma ferramenta de linha de comando funcional que aplique os conceitos teóricos de criptografia assimétrica para criar um mecanismo robusto de assinatura de ficheiros. A aplicação desenvolvida permite a um utilizador gerar o seu próprio par de chaves, assinar qualquer ficheiro digital e, subsequentemente, permitir que terceiros verifiquem a validade dessa assinatura.

Este relatório está estruturado da seguinte forma: a secção 2 descreve a fundamentação teórica por trás da criptografia RSA e das assinaturas digitais. A secção 3 detalha o ambiente experimental utilizado e apresenta a análise dos resultados obtidos nos testes de validação. Finalmente, a secção 4 apresenta as conclusões técnicas sobre o trabalho realizado, seguida pela bibliografia consultada.

2 Fundamentação Teórica

Para a construção da ferramenta, foram aplicados diversos conceitos criptográficos que, em conjunto, formam um sistema de assinatura digital seguro e moderno.

2.1 Criptografia Assimétrica e o Algoritmo RSA

Diferente da criptografia simétrica, que utiliza uma única chave para cifrar e decifrar, a criptografia assimétrica emprega um par de chaves matematicamente relacionadas: uma chave pública, que pode ser distribuída livremente, e uma chave privada, que deve ser mantida em segredo pelo seu proprietário. O algoritmo RSA, proposto por Rivest, Shamir e Adleman, é o padrão mais difundido para criptografia de chave pública [1]. A sua segurança reside na dificuldade computacional de fatorar números inteiros muito grandes. O processo de geração de chaves envolve a escolha de dois números primos grandes, p e q, a partir dos quais se calcula o módulo $n = p \times q$ e a função totiente $\phi(n) = (p - 1)(q - 1)$. Um expoente público e é escolhido, e o expoente privado d é calculado como o inverso modular de e em relação a $\phi(n)$.

2.2 Assinatura Digital

Uma assinatura digital é um mecanismo que vincula a identidade do signatário a um documento digital. O processo de assinatura consiste em duas etapas principais: primeiro, calcula-se um resumo criptográfico (hash) da mensagem; segundo, este hash é cifrado com a chave privada do signatário. Para verificar a assinatura, qualquer pessoa com a chave pública do signatário pode decifrar a assinatura para obter o hash original e compará-lo com um novo hash calculado a partir da mensagem recebida. Se os hashes forem idênticos, a assinatura é válida, garantindo tanto a autenticidade (só o detentor da chave privada poderia ter cifrado o hash) quanto a integridade (qualquer alteração na mensagem resultaria num hash diferente) [2].

2.3 Funções de Hash: SHA-3

Uma função de hash criptográfica mapeia dados de tamanho arbitrário para uma saída de tamanho fixo. Para ser segura, deve ser resistente a colisões (difícil encontrar duas entradas que gerem a mesma saída) e unidirecional (impossível reverter o hash para obter a entrada original). Este projeto utiliza o SHA-3 (Secure Hash Algorithm 3), o mais recente padrão do NIST, escolhido pela sua robustez contra ataques que afetam padrões mais antigos [3].

2.4 OAEP e Codificação Base64

O RSA "textbook"é vulnerável a certos ataques. O OAEP (Optimal Asymmetric Encryption Padding) é um esquema de preenchimento que adiciona aleatoriedade à mensagem antes da cifragem, tornando o RSA semanticamente seguro [4]. Por fim, a codificação Base64 é utilizada para converter dados binários (como a assinatura) num formato de texto ASCII, permitindo que o documento assinado seja armazenado ou transmitido em meios que lidam apenas com texto, sem risco de corrupção.

3 Ambiente Experimental e Análise de Resultados

3.1 Descrição do Cenário

- **Hardware:** O desenvolvimento e os testes foram realizados num computador pessoal (desktop com sistema operativo Windows 11).
- **Software:** A linguagem de programação utilizada foi Python na versão 3.11. A principal dependência externa foi a biblioteca `pycryptodome`, utilizada para a manipulação de chaves no formato padrão PEM. A interface da aplicação foi construída com o módulo nativo `argparse`.
- **Topologia de Rede:** Por se tratar de uma aplicação para criptografia de ficheiros locais, não foi utilizada uma topologia de rede. Todas as operações de assinatura e verificação foram realizadas no sistema de ficheiros do computador local.
- **Configurações:** A ferramenta foi configurada para operar via linha de comando, aceitando subcomandos para as diferentes funcionalidades (`gerar-chaves`, `assinar`, `verificar`). As chaves geradas são salvas em ficheiros com a extensão `.pem`, um formato padrão que facilita a interoperabilidade.

3.2 Análise de Resultados

Foi executado um roteiro de testes para validar cada funcionalidade exigida, conforme descrito abaixo.

- **Atividade 1: Geração de Chaves.** Executou-se o comando `py geradorVerificador.py gerar-chaves`. O resultado foi a criação bem-sucedida dos ficheiros `chave_privada.pem` e `chave_publica.pem` no diretório de trabalho, confirmando que a geração e o armazenamento de chaves funcionam como esperado.
- **Atividade 2: Assinatura e Verificação bem-sucedida.** Um ficheiro de texto `mensagem.txt` foi criado e assinado com o comando `py geradorVerificador.py assinar`

`mensagem.txt`. Em seguida, o ficheiro resultante, `documento.assinado`, foi verificado com `py geradorVerificador.py verificar documento.assinado`. O resultado foi "ASSINATURA VÁLIDA", e o conteúdo original foi exibido corretamente, validando o fluxo principal do sistema.

- **Atividade 3: Teste de Integridade.** O ficheiro `documento.assinado` foi aberto num editor de texto e um único caractere da sua string Base64 foi alterado. Ao tentar verificar o ficheiro modificado, o programa retornou um erro de decodificação ou "ASSINATURA INVÁLIDA", comprovando que o sistema deteta qualquer adulteração no documento.
- **Atividade 4: Teste de Autenticidade.** Um novo par de chaves foi gerado. Tentou-se verificar um documento assinado com a chave privada antiga, mas usando a nova chave pública. O resultado foi "ASSINATURA INVÁLIDA", o que confirma que a assinatura está matematicamente ligada a um par de chaves específico.
- **Problemas Encontrados:** Durante a configuração do ambiente no Windows, foi encontrado um problema comum em que o comando `python` invocava a Microsoft Store em vez da instalação existente. A solução foi desabilitar os "aliases de execução de aplicativos"do Windows ou, de forma mais robusta, utilizar o lançador `py`, que deteta automaticamente a instalação correta.

4 Conclusões

A implementação da ferramenta de assinatura digital RSA foi concluída com sucesso, cumprindo todos os requisitos funcionais e de segurança propostos. O trabalho permitiu a aplicação prática de conceitos teóricos fundamentais da criptografia assimétrica, como a geração de chaves, o uso de funções de hash e a lógica por trás da assinatura e verificação. A ferramenta desenvolvida é robusta, validando corretamente a integridade e a autenticidade de ficheiros e rejeitando assinaturas inválidas ou documentos adulterados. O projeto reforça a compreensão de como as assinaturas digitais fornecem uma base de confiança essencial para a segurança da informação no mundo digital.

5 Bibliografia

Referências

[1] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[2] W. Stallings. *Cryptography and Network Security: Principles and Practice*. 7th ed. Pearson, 2017.

[3] National Institute of Standards and Technology (NIST). FIPS PUB 202: SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. 2015.

[4] M. Moriarty, B. Kaliski, J. Jonsson, A. Rusch. PKCS #1: RSA Cryptography Specifications Version 2.2. RFC 8017, 2016.

[5] PyCryptodome Developers. PyCryptodome Documentation. Disponível em: https://www.pycryptodome.org/en/latest/.

A Código Fonte

O código fonte completo do projeto está disponível no repositório do GitHub no link a seguir: https://github.com/viniciustome61/Gerador-assinatura.git