Gerador e Verificador de Assinaturas RSA

Trabalho de Implementação 2

Vinícius Dos Santos Tomé

Segurança Computacional - CIC0201

8 de outubro de 2025

Agenda

O Problema: Confiança no Mundo Digital

- Como garantir que um documento digital é autêntico?
- Como saber que não foi alterado no caminho?

Solução

Assinaturas Digitais: um mecanismo criptográfico para prover:

- **Autenticidade:** Prova de quem é o autor.
- **Integridade:** Garantia de que o conteúdo não foi modificado.
- **Não-repúdio:** O autor não pode negar a autoria.

Criptografia Assimétrica e RSA

Par de Chaves:

- **Chave Privada:**
  - Secreta, conhecida apenas pelo dono.
  - Usada para assinar documentos.
- **Chave Pública:**
  - Pode ser partilhada livremente.
  - Usada para verificar assinaturas.

Algoritmo RSA:

- O padrão mais utilizado para criptografia de chave pública.
- Segurança baseada na dificuldade de fatorar números primos muito grandes.

Como Funciona a Assinatura Digital?

Processo de Assinatura (com a Chave Privada)

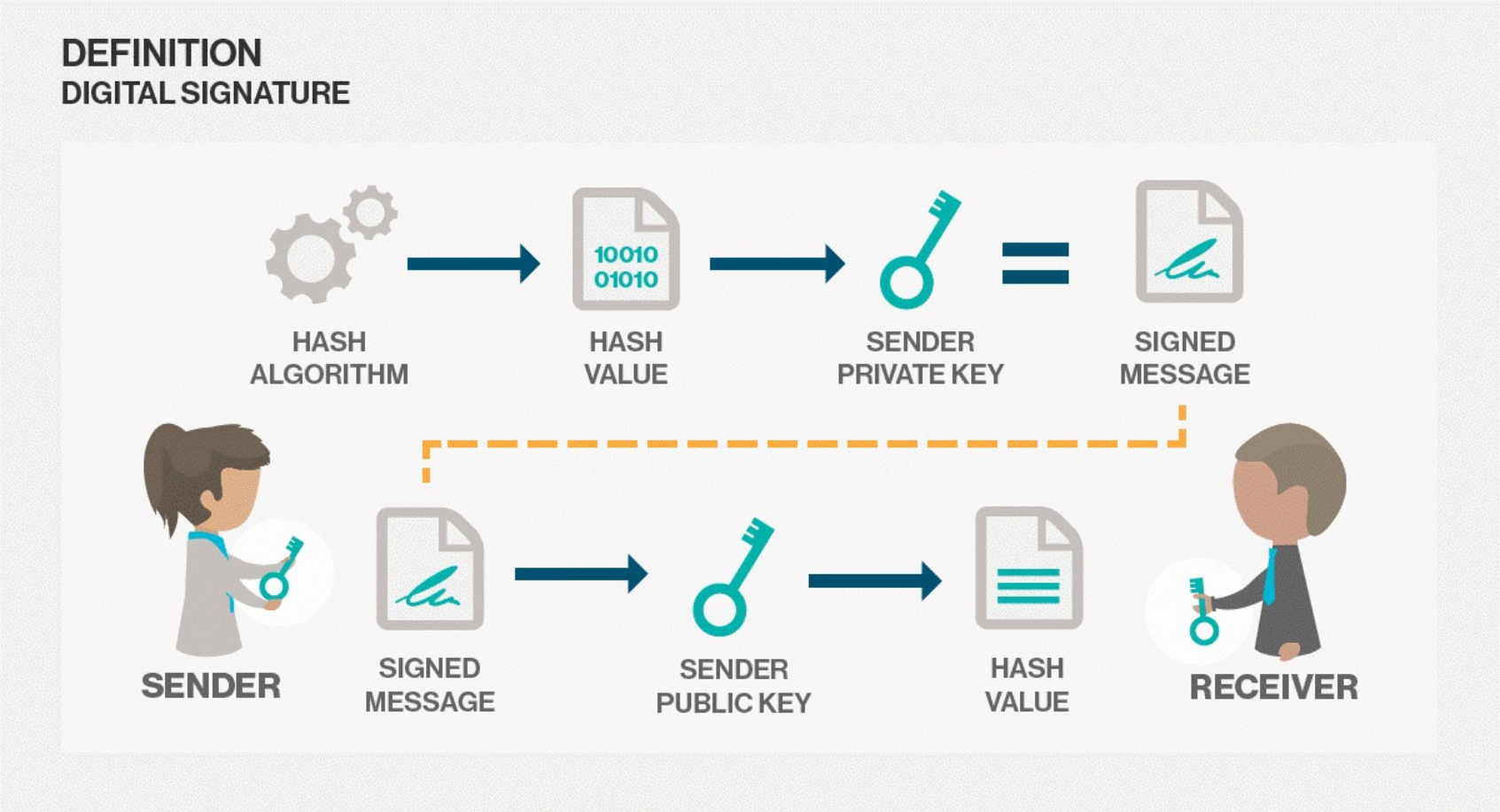

Figura: Documento $\rightarrow$ Hash $\xrightarrow{\text{Cifrar com Chave Privada}}$ Assinatura

Tecnologias de Suporte

- **SHA-3:** Função de hash moderna e segura para criar o "resumo"do documento.
- **OAEP (Optimal Asymmetric Encryption Padding):** Padrão de preenchimento que adiciona aleatoriedade antes da cifragem, tornando o RSA mais robusto contra ataques.
- **Base64:** Codificação que transforma os dados binários da assinatura em texto simples, facilitando o armazenamento e a transmissão.

Estrutura do Projeto

- Desenvolvido em **Python** como uma ferramenta de linha de comando (CLI).
- Utiliza a biblioteca nativa `argparse` para a interface.
- **Código modularizado:**
  - `parte1.py`: Lógica criptográfica de baixo nível (geração de primos, RSA, OAEP).
  - `parte2.py`: Funções de alto nível para assinar e verificar.
  - `parte3.py`: Lógica para analisar (parsing) o documento assinado.
  - `geradorVerificador.py`: Ponto de entrada da aplicação.

Demonstração: Geração de Chaves

Comando

```
py geradorVerificador.py gerar-chaves
```

Resultado

- Cria os ficheiros `chave_privada.pem` e `chave_publica.pem`.
- As chaves são persistentes e reutilizáveis.

Demonstração: Assinar um Ficheiro

Comando

```
# Criar um ficheiro de teste
echo "Mensagem de teste" > mensagem.txt

# Assinar o ficheiro
py geradorVerificador.py assinar mensagem.txt
```

Resultado

- Gera o ficheiro `documento.assinado`.
- Contém a mensagem original e a assinatura, codificadas em Base64.

Demonstração: Verificar uma Assinatura

Comando

```
py geradorVerificador.py verificar documento.assinado
```

Resultado

Saída Esperada

```
Resultado: ASSINATURA VÁLIDA.
Conteúdo do ficheiro verificado:
---
Mensagem de teste
---
```

Conclusões

- O projeto foi concluído com sucesso, cumprindo todos os requisitos propostos.
- A ferramenta desenvolvida é uma aplicação prática e funcional dos conceitos de criptografia assimétrica.
- O sistema é robusto, validando corretamente a **integridade** e a **autenticidade** de ficheiros e rejeitando assinaturas inválidas.
- O trabalho reforça a compreensão sobre a importância das assinaturas digitais como pilar da segurança da informação.

Obrigado!

Perguntas?