## **Criptografia: Simétrica, Assimétrica e Funções Hash**

### **Criptografia Simétrica e Assimétrica**

A diferença essencial entre a criptografia simétrica e a assimétrica reside no **número de chaves** empregadas no processo de proteção de dados.

**Criptografia Simétrica (Chave Secreta/Compartilhada):** Utiliza uma única chave para cifrar e decifrar a informação. Suas principais vantagens são a alta velocidade e a eficiência no uso de recursos computacionais, tornando-a ideal para grandesvolumes de dados. No entanto, sua segurança depende da trocaseguradachave entre o emissor e o receptor, e ela nãoofereceirretratabilidade (prova de autoria). Algoritmos notórios incluem o **AES** (Advanced Encryption Standard) e o 3DES.

* **Aplicações da Simétrica:** Criptografia de disco (como BitLocker), criptografia de tráfego de dados em massa após o *handshake* inicial em conexões VPN, e proteção de grandes volumes de dados.
* **Exemplo Prático (Python):** Na criptografia simétrica é utlizada a mesma chave para criptografar dados e descriptografar os mesmos. O algoritmo utilizado é um comumente utilizado e muito seguro.

No exemplo é utilizado a biblioteca Cryptography com uma implementação do Fernet baseada na criptografia simétrica AES.

Link do exemplo:

https://github.com/VictorDomingosMoreiraDEV/Projetos-A3/blob/main/fernet.py

**Criptografia Assimétrica (Chave Pública/Privada):** Utiliza um **par de chaves**: uma pública (que pode ser amplamente divulgada) e uma privada (mantida em sigilo). Uma mensagem cifrada com a chave pública só pode ser decifrada pela chave privada correspondente. Embora seja **mais lenta** e exija mais recursos, ela resolve o problema da troca de chaves e permite a irretratabilidade por meio de assinaturasdigitais. O algoritmo mais conhecido é o **RSA**, que utiliza chaves longas (2048 bits ou mais).

* **Aplicações da Assimétrica:** **Assinaturas Digitais** (uso da chave privada para provar a autoria), **Certificados Digitais** (SSL/TLS) para autenticar servidores e, crucialmente, para a **troca segura da chave simétrica** em sistemas híbridos.
* **Exemplo Prático (Python):** Implementações **Hazmat** da biblioteca cryptography são usadas para gerar e utilizar o par de chaves **RSA**.

Link do exemplo:

https://github.com/VictorDomingosMoreiraDEV/Projetos-A3/blob/main/hazmat.py

### **Funções Hash Criptográficas**

Uma função hash criptográfica pega uma entrada de qualquer tamanho e produz uma saídadetamanhofixo (o *hash* ou *digest*). Este é um processo unidirecional (não é possível reverter o *hash* para a mensagem original), essencial para garantir a integridade e autenticidade dos dados.

**Propriedades Chave:**

* **Determinismo:** A mesma entrada sempre gera o mesmo *hash*.
* **Resistência à Colisão:** É inviável encontrar duas entradas diferentes que produzam o mesmo *hash*.
* **Resistência à Pré-imagem:** É inviável reverter o *hash* para a mensagem original.
* **Efeito Avalanche:** Pequenas mudanças na entrada resultam em grandes alterações no *hash* de saída.

**Padrões Atuais:** Enquanto **MD5** e **SHA-1** são obsoletos e inseguros, a família **SHA-2** (**SHA-256** é o mais comum) é o padrão atual, amplamente utilizado em segurança de senhas, verificação de integridade de arquivos e na tecnologiaBlockchain. O SHA**-**3 representa a geração mais recente. É vital acompanhar as recomendações de entidades como o NIST e migrar para algoritmos mais robustos.

* **Exemplo Prático (Python):** O módulo padrão **Hashlib** do Python é usado para implementar funções como o SHA**-**256 (Secure Hash Algorithm 256-bit).

Link do exemplo:

https://github.com/VictorDomingosMoreiraDEV/Projetos-A3/blob/main/hashlib.py

### **Conclusão: A Segurança como uma Equipe Coordenada**

A escolhacorretaeaimplementaçãocuidadosa desses mecanismos criptográficos tornam um sistema robusto e confiável. A segurança moderna funciona como uma equipe bem coordenada:

1. **Velocidade e Sigilo:** Os métodos simétricos (como o Fernet) agem rapidamente para proteger nossos dados mais importantes, garantindo a confidencialidade.
2. **Identidade e Autenticidade:** Os métodos assimétricos (como o RSA) criam uma "identidade digital" segura, permitindo que as partes se autentiquem.
3. **Integridade e Prova:** As funções*hash* (como o SHA-256) funcionam como uma prova de que a mensagem não foi adulterada, garantindo a verdade do que foi enviado.

Muitas vezes, a segurança só é lembrada após um vazamento de dados ou uma fraude. A verdade é que a prevençãoeacompreensãobásica desses mecanismos são o que nos protege antes mesmo de o risco aparecer. Compreender a importância da criptografia — mesmo sem ser um programador — é essencial para manter sua privacidade**,** suaidentidadeesuasinformações sob controle em um ambiente cada vez mais exposto.

Referencias:

<https://www.totvs.com/blog/gestao-para-assinatura-de-documentos/criptografia-simetrica-e-assimetrica/>

<https://www.ssl.com/pt/artigo/o-que-%C3%A9-uma-fun%C3%A7%C3%A3o-de-hash-criptogr%C3%A1fico/>

Victor Domingos Moreira – 825155879

Lucas Gonçalves da Silva – 825113362

Victor de Morais Nelson – 825243925

Felipe Duarte Battaglini – 825165863

Cauã Guidio Viana - 825168423