TEORIA

Segurança da Encriptação: a segurança da cifra depende do segredo da **chave** e não do algoritmo. O algoritmo é público, a **chave** é secreta.

Atacantes

- **Atacante Passivo**: escuta a comunicação entre duas partes, mas não interfere na comunicação. O atacante tenta obter a mensagem original.
- **Atacante Ativo**: escuta a comunicação entre duas partes e interfere na comunicação. O atacante tenta obter a mensagem original e alterar a mensagem cifrada.
- **Atacante computacionalmente limitado**: apenas consegue aplicar um certo número de algoritmos.
- **Atacante computacionalmente ilimitado**: consegue aplicar qualquer algoritmo sem qualquer limite a nível de memória ou tempo.

---

1. Classical Ciphers

- **Cifra de César**: uma das mais simples e conhecidas, onde cada letra é deslocada um número fixo de posições no alfabeto. Sensível a ataques de força bruta.
- **Cifra de Vigenère**: utiliza uma palavra-chave para determinar o deslocamento de cada letra, tornando-a mais segura que a cifra de César.
- **OTP**: um método de criptografia que utiliza uma chave única e aleatória do mesmo comprimento. (**tamanho fixo**) que a mensagem e que é utilizada apenas uma única vez, garantindo segurança perfeita (**absoluta**). Garante **integridade** (HOTP) e **autenticidade**. No entanto, é difícil de implementar na prática devido à necessidade de gerar e distribuir chaves longas e aleatórias.

---

2. Segurança

Propriedades:

- **Confidencialidade**: apenas o destinatário pode ler a mensagem.
- **Integridade**: a mensagem não foi alterada durante a transmissão.
- **Autenticidade**: o remetente é quem diz ser.
- **Não repúdio**: o remetente não pode negar que enviou a mensagem.
- **Disponibilidade**: a mensagem está disponível para o destinatário sempre que necessário.
- **Anonimidade**: o remetente não pode ser identificado.

---

3. Criptografia Simétrica

Stream ciphers + block ciphers + one way functions + key management. **A mesma chave para cifrar e decifrar** length(key) >= length(message)

3.1 Stream Ciphers

Cifram mensagens bit a bit ou byte a byte. A chave é gerada a partir de um algoritmo que gera uma sequência de bits pseudo-aleatória (keystream). O algoritmo é conhecido, mas a chave é secreta. A cifragem é feita através de XOR entre a mensagem e o keystream.

Podem ser:

- **Síncronos**: o keystream é gerado independentemente da mensagem. Perda de sincronismo resulta na perda de dados.
- **Self-síncronos**: capazes de recuperar o sincronismo quando um bit é perdido. O keystream depende dos bits anteriores do ciphertext.

É necessário utilizarmos um NONCE (número usado uma vez) para garantir que a chave é única para cada mensagem. **O NONCE é enviado junto com a mensagem cifrada e não precisa de ser aleatório.**

Exemplo de stream cipher: RC4 e ChaCha20.

3.2 Block Ciphers

Block ciphers dividem a mensagem em blocos de tamanho fixo e aplicam uma função de encriptação a cada bloco. O block cipher mais conhecido é o AES (Advanced Encryption Standard), que utiliza blocos de 128 bits e chaves de 128, 192 ou 256 bits. A cifra de bloco é mais segura que a cifra de fluxo, pois não depende da sincronização entre o remetente e o destinatário. No entanto, é mais lenta e requer mais recursos computacionais.

AES: confidencialidade apenas!

Obs: Stream ciphers são mais rápidos que block ciphers, mas menos seguros. Block ciphers são mais seguros, mas mais lentos.

Existem dois designs distintos para block ciphers:

- **Diffusion**: garante que uma pequena alteração na mensagem resulta numa grande alteração no ciphertext. Isto é conseguido através de substituições e permutações.
- **Confusion**: garante que a relação entre a chave e o ciphertext é complexa e não linear. Isto é conseguido através de operações matemáticas complexas, como multiplicação e adição.

Padding: Para cifrar mensagens de tamanho variável, é necessário adicionar padding aos blocos para que tenham o tamanho correto, uma vez que alguns modos de operação requerem que o tamanho da mensagem seja um múltiplo do tamanho do bloco. O padding mais comum é o PKCS#7, que adiciona bytes com o valor do número de bytes de padding.

Modos de operação:

- **ECB (Electronic Codebook)**: cada bloco é cifrado de forma independente. Vulnerável a ataques de repetição e análise de frequência, pois blocos idênticos resultam em blocos cifrados idênticos. Requer padding. Deve ser apenas utilizado para cifrar mensagens de tamanho equivalente ao tamanho de um bloco.
- **CBC (Cipher Block Chaining)**: cada bloco é cifrado com o bloco anterior (dependência), garantindo que blocos idênticos resultem em blocos cifrados diferentes (XOR). Utiliza um vetor de inicialização

(IV) para o primeiro bloco, que deve ser único e aleatório para cada mensagem. O IV é enviado junto com a mensagem cifrada. Desencriptação pode ser feita em paralelo, mas a encriptação não pode. O último bloco pode ser um *CBC-MAC*, que é uma assinatura digital do bloco anterior, ou seja, o último bloco é cifrado com a chave e o bloco anterior, garantindo a integridade da mensagem. Isto pois, qualquer mudança nos bits do plaintext fazem com que o ciphertext seja alterado de forma completamente não previsível. **Ataques de padding, devemos utilizar AES-GCM com padding ou HMAC.**

- **CFB (Cipher Feedback)**: semelhante ao CBC, mas permite cifrar mensagens de tamanho variável. Utiliza um vetor de inicialização (IV) para o primeiro bloco, que deve ser único e aleatório para cada mensagem (funciona como um NONCE). O IV é enviado junto com a mensagem cifrada. O XOR acontece depois. Implementa um stream cipher auto-síncrono, onde o cifrador é alimentado com o ciphertext do bloco anterior. Isto permite que a cifra de bloco seja usada como uma cifra de fluxo, mas com a segurança adicional de que blocos idênticos resultam em blocos cifrados diferentes. Não necessita de padding, pois a cifra de bloco é aplicada a um vetor de inicialização (IV) e não aos blocos da mensagem. O keystream depende da chave, do IV e do plaintext anterior.
- **OFB (Output Feedback)**: semelhante ao CFB, mas a cifra de bloco é aplicada a um vetor de inicialização (IV), que deve ser uma espécie de NONCE, e não aos blocos da mensagem.
    - síncrono;
    - não necessita de padding;
    - o keystream nao depende do plaintext anterior;
- **CTR (Counter)**: utiliza um contador para gerar uma sequência de bits pseudo-aleatória, que é XORed com a mensagem. O contador é incrementado para cada bloco, garantindo que blocos idênticos resultem em blocos cifrados diferentes. O contador é enviado junto com a mensagem cifrada.
    - síncrono;
    - não necessita de padding;
    - o keystream não depende do plaintext anterior;

Authenticated Encryption (AE)

Modos de operação que garantem a **autenticidade**, **confidencialidade** e **integridade** da mensagem cifrada, como o GCM (Galois/Counter Mode) e o CCM (Counter with CBC-MAC). Estes modos de operação utilizam uma função de autenticação para garantir que a mensagem não foi alterada durante a transmissão.

Advanced Encryption Standard (AES)

AES é um algoritmo de cifra de bloco que utiliza chaves de 128, 192 ou 256 bits e blocos de 128 bits. É amplamente utilizado em aplicações de segurança, como VPNs, TLS e criptografia de disco.

3.3 One-Way Functions

Funções unidirecionais são funções matemáticas que são fáceis de calcular, mas difíceis de inverter. São utilizadas em criptografia para garantir a **integridade** e **autenticidade** das mensagens. Hash functions por si só não garantem integridade.

Exemplos: SHA-256, SHA-3, BLAKE2.

Podem existir duas mensagens distintas que geram o mesmo hash.

Message Authentication Codes (MACs): MACs são códigos de autenticação de mensagem que garantem a **integridade** e **autenticidade** das mensagens. São gerados a partir de uma função unidirecional e uma chave secreta. O código gerado é baseado num segredo secreto que apenas as partes legítimas (remetente e destinatário) conhecem. Utiliza uma chave simétrica.

Exemplo: Poly1305, CMAC.

HMACs são utilizados em protocolos de segurança, como TLS e SSH, para garantir que as mensagens não foram alteradas durante a transmissão. Construção de um MAC a partir de uma função de hash. Neste caso, até são performadas duas passagens pela função, de forma a evitar ataques de length extension.

Key-Derivation Functions (KDF): Permitem a derivação de chaves a partir de material criptográfico secreto. Por exemplo, derivar uma chave a partir de uma *password* (baixa entropia e vulnerável a ataques de enumeração. Para minimizar tais problemas, deve ser utilizado um salt) ou *passphrase* ou até mesmo de um master secret acordado a partir de um protocolo de handshaking. Exemplo: PBKDF2, bcrypt, scrypt.

salt - número aleatório adicionado à *password* antes de ser processada pela função de hash. O salt é enviado junto com o hash, permitindo que o destinatário verifique a *password* sem precisar conhecer o salt.

3.4 Key Management

A gestão de chaves é um aspecto crítico da criptografia simétrica. Envolve a geração, distribuição, armazenamento e revogação de chaves secretas. A segurança da criptografia simétrica depende da segurança das chaves secretas. Se uma chave secreta for comprometida, todas as mensagens cifradas com essa chave também serão comprometidas.

Key handling:

- O período de vida de uma chave deve ser o mais curto possível;
- static keys => long-term keys, que são utilizadas durante um período prolongado de tempo;
- ephemeral keys => session keys, que são utilizadas apenas durante uma sessão de comunicação; (criptografia simétrica)

As chaves não devem ser utilizadas para diversos propósitos, pois isso aumenta o risco de comprometimento. Por exemplo, uma chave utilizada para cifrar mensagens não deve ser utilizada para autenticar mensagens. Devemos utilizar KDFs para derivar chaves a partir de um master secret ou de uma *password*.

CMKs são sistemas de gestão de chaves que permitem a geração, distribuição, armazenamento e revogação de chaves secretas. Estes sistemas são utilizados para garantir a segurança das chaves secretas e para facilitar a gestão de chaves em ambientes complexos.

---

4. Criptografia Assimétrica

A criptografia assimétrica utiliza um **par de chaves**: uma **chave pública** e uma **chave privada**. A chave pública é utilizada para **cifrar** mensagens, enquanto a chave privada é utilizada para **decifrar** mensagens. A segurança da criptografia assimétrica depende da dificuldade de fatoração de números grandes. A criptografia assimétrica é mais lenta que a criptografia simétrica, mas oferece vantagens em termos de segurança e gestão de chaves. É amplamente utilizada em protocolos de segurança, como TLS, SSH e PGP.

Autenticação + confidencialidade

4.1 Key Agreement

Diffie-Hellman é um protocolo (**que não garante autenticidade**) de troca de chaves que permite que duas partes estabeleçam um segredo compartilhado através de um canal inseguro. **Não garante autenticidade**. O protocolo funciona da seguinte forma:

1. Definem-se parâmetros públicos: um número primo grande (p) e uma base (g), que é uma raiz primitiva módulo p.
2. Cada parte escolhe um número secreto privado (por exemplo, a e b).
3. São calculadas as chaves públicas:
    - Parte A: A = g^a mod p
    - Parte B: B = g^b mod p
4. As chaves públicas (A e B) são trocadas entre as partes.
5. Cada parte utiliza a sua chave privada e a chave pública recebida para calcular o segredo:
    - Parte A: s = B^a mod p
    - Parte B: s = A^b mod p

Como resultado, ambas as partes obtêm o mesmo valor s, que pode ser usado como chave de sessão para criptografia simétrica. Mesmo que um atacante intercepte as chaves públicas, sem conhecer os números privados não é possível derivar o segredo compartilhado.

O protocolo Diffie-Hellman é fundamental em muitas aplicações de segurança, embora seja importante aplicar autenticação adicional para prevenir ataques de intermediário.

O algoritmo Diffie-Hellman é vulnerável a ataques MITM (Man-in-the-Middle), mas seguro contra adversários passivos.

4.2 Criptografia da chave pública

Chave pública para encriptar, chave privada para decriptar. Garante **confidencialidade e autenticidade (dependendo do algoritmo).**

Não garante autenticidade das chaves públicas

Algoritmos:

- **RSA (Rivest-Shamir-Adleman):** um dos algoritmos mais conhecidos e utilizados. Baseia-se na dificuldade de fatoração de números grandes. Utiliza chaves de 1024, 2048 ou 4096 bits. É amplamente utilizado em protocolos de segurança, como TLS e SSH. **Garante confidencialidade e autenticidade**.
- **DSA (Digital Signature Algorithm):** utilizado para assinaturas digitais. Baseia-se na dificuldade do problema do logaritmo discreto. Utiliza chaves de 1024, 2048 ou 3072 bits. É utilizado em protocolos de segurança, como TLS e SSH. **Garante autenticidade, mas não confidencialidade**. Mais eficiente que o RSA para assinaturas digitais, mas menos eficiente para cifragem.
- **ECDSA (Elliptic Curve Digital Signature Algorithm):** uma variante do DSA que utiliza curvas elípticas para gerar assinaturas digitais. É mais eficiente que o DSA e o RSA, pois utiliza chaves menores para o mesmo nível de segurança. É amplamente utilizado em protocolos de segurança, como TLS e SSH. **Garante autenticidade, mas não confidencialidade**.

- **ElGamal:** utilizado para cifragem e assinaturas digitais. Baseia-se na dificuldade do problema do logaritmo discreto. É menos utilizado que o RSA e o DSA, mas ainda é relevante em algumas aplicações de segurança. **Garante confidencialidade e autenticidade**.
- **ECC (Elliptic Curve Cryptography):** utiliza curvas elípticas para cifragem e assinaturas digitais. É mais eficiente que o RSA e o DSA, pois utiliza chaves menores para o mesmo nível de segurança. É amplamente utilizado em protocolos de segurança, como TLS e SSH. **Garante confidencialidade e autenticidade**.

4.3 Paradigma KEM/DEM

O paradigma KEM/DEM (Key Encapsulation Mechanism/Data Encapsulation Mechanism) é uma abordagem híbrida que combina criptografia simétrica e assimétrica para garantir segurança e eficiência.

- **KEM (Key Encapsulation Mechanism)**: utiliza criptografia assimétrica para encapsular uma chave simétrica. A chave simétrica é gerada aleatoriamente e encapsulada com a chave pública do destinatário. O destinatário utiliza a sua chave privada para decapsular a chave simétrica.
- **DEM (Data Encapsulation Mechanism)**: utiliza a chave simétrica para cifrar os dados. A chave simétrica é utilizada para cifrar a mensagem, garantindo eficiência e velocidade na cifragem e decifragem dos dados. Este paradigma é amplamente utilizado em protocolos de segurança, como TLS e SSH, para garantir a confidencialidade e autenticidade das mensagens. A combinação de criptografia simétrica e assimétrica permite que as partes estabeleçam um segredo compartilhado de forma segura e eficiente, aproveitando as vantagens de ambos os tipos de criptografia.

4.4 Infraestrutura de chave pública (PKI)

- Uma PKI é um sistema de hardware, software, políticas e procedimentos que gere, distribui, usa e amrazena chaves públicas e certificados digitais.

X509

- Apenas para autenticar chaves publicas
- Pressupoe uma relação de confiança na entidade responsável pela emissão do certificado (Autoridade de Certificação - CA).
- Permite associar uma chave pública à entidade detentora da chave privada correspondente, garantindo a autenticidade e integridade da chave pública.

Este certificado inclui:

- Nome do proprietário (quem possui a chave privada associada à chave publica)
- A chave publica
- Nome da entidade que emitiu o certificado (CA)
- Assinatura digital da CA

Esta informação é representada como atributo/valor (dicionário) e standarizada:

- DER (ASN.1 notation): formato binário;
- PEM: formato ASCII (base64) com cabeçalho e rodapé.

Atualmente os certificados utilizados sao X.509v3

Esta versão introduziu novos campos:

- Extensões: permitem adicionar novos campos ao certificado, como por exemplo:
    - Uso da chave: para que é que a chave publica pode ser usada (ex: assinatura digital, autenticação, etc);
    - Nome alternativo do sujeito: permite adicionar mais nomes ao certificado (ex: www.exemplo.com, mail.exemplo.com);
    - Autoridade de certificação: quem emitiu o certificado;
    - Lista de revogação de certificados (CRL): lista de certificados revogados.
    - Possui informação sobre o estado (crítica ou não crítica).

PKIX

Muitas entidades envolvidas numa PKI:

- Proprietários de certificados: quem requere um certificado para a sua chave publica;
- Clientes: quem usa a chave publica que contem o certificado;
- Autoridades de certificação (CAs): quem emite os certificados;
- Autoridades de registo (RAs): quem verifica a identidade do proprietário do certificado antes de o emitir;
- Repositórios: onde os certificados são armazenados e disponibilizados;

PKIX definiu um conjunto de especificações e protocolos permitindo:

- Sistemas/aplicação que efetivamente confiem em certificados X.509 (procotolos operacionais);
- gestão de PKI - operações de emissão, revogação e renovação de certificados;

4.5 Certificados

Estrutura:

- versão
- número de série
- sujeito
- emissor (CA)
- chave pública do sujeito
- assinatura da CA
- validade
- extensões
    - uso da chave: restringe o uso da chave publica
    - basic contraints: define se a chave publica é uma CA ou não

Criação de certificados: Um certificado é criado por uma autoridade de certificação (CA) que assina digitalmente a chave pública do proprietário do certificado. A assinatura digital é feita utilizando a chave privada da CA, garantindo que o certificado é autêntico e não foi alterado. O certificado contém a chave pública do proprietário, o nome do proprietário, o nome da CA, a validade do certificado e outras informações relevantes.

Utilização de certificados: Quando a Alice necessita de enviar a sua chave pública para o Bob, ela envia o seu certificado, que contém a sua chave pública e é assinado pela CA. O Bob verifica a assinatura do certificado utilizando a chave pública da CA, garantindo que o certificado é autêntico e não foi alterado. Se a assinatura for válida, o Bob pode confiar na chave pública da Alice contida no certificado e utilizá-la para

cifrar mensagens destinadas à Alice. Os certificados são utilizados para garantir a autenticidade e integridade das chaves públicas, permitindo que as partes confiem nas chaves públicas umas das outras sem a necessidade de um canal seguro pré-estabelecido.

Os certificados sao sempre utilizados aquando da partilha da chave publica. A chave publica é sempre acompanhada de um certificado. Cerficados sem CA são auto-assinados (self-signed). A chave publica é assinada pela própria entidade que a possui. Este tipo de certificado não é confiável, pois não existe uma terceira parte que o valide.

Validação Para cada certificado, o cliente tem de verificar:

- Assinatura: verificar se a assinatura do certificado é válida (assinar com a chave publica da CA);
- A aplicação do certificado: verificar se o certificado é aplicável à chave publica (verificar se o subject corresponde ao nome do servidor);

Trust Anchors O utilizador sabe um numero limitado de chaves publicas que pertençam a entidades de confiança (CA) que atuam como as raízes de confiança. Normalmente é o sistema operativo que gere a lista de CA de confiança. A compilação e atualização é feita pelo SO.

Não garante a confidencialidade da chave pública do titular.

4.6 Protocolo Station-to-Station (STS)

O protocolo Station-to-Station (STS) é um protocolo de troca de chaves que permite que duas partes estabeleçam um segredo compartilhado de forma segura, garantindo a autenticidade, integridade e não-repúdio das mensagens. O protocolo é baseado na troca de mensagens cifradas utilizando criptografia assimétrica e criptografia simétrica.

---

5. Criptação Hibrida

Combina a utilização de criptografia simétrica com a utilização de criptografia assimétrica. Também pode ser chamado de envelope digital.

Funcionamento

- Chave simétrica é gerada de forma aleatória.
- Mensagem é cifrada com essa chave.
- Chave gerada é cifrada com a chave pública do destinatário.
- No destino, é decifrada, com recurso à chave privada, a chave gerada aleatoriamente.
- Com recurso a essa, é decifrada a mensagem propriamente dita.

Envelope Digital

Um envelope digital é uma técnica de cifra híbrida onde a mensagem é cifrada com criptografia simétrica (rápida), e a chave simétrica é cifrada com criptografia assimétrica (segura para troca de chaves). Isso garante confidencialidade e desempenho. Exemplo: Ao enviar um e-mail seguro, o conteúdo é cifrado com AES e a chave AES é cifrada com a chave pública do destinatário (RSA).

6. Access Control Policies

Políticas de controlo de acesso definem quem pode aceder a que recursos e em que condições. As políticas podem ser baseadas em papéis (RBAC), atributos (ABAC) ou listas de controlo de acesso (ACL).

No Linux clássico, o modelo de controlo de acesso é do tipo discricionário (DAC - Discretionary Access Control), o que significa que os proprietários dos ficheiros podem decidir quem tem permissões para ler, escrever ou executar seus ficheiros. O sistema utiliza uma combinação de permissões básicas (leitura, escrita, execução) atribuídas ao dono, grupo e outros.

O arquivo /etc/passwd contém informações como nome de usuário, UID, GID e shell, mas o campo de senha foi substituído por um "x", indicando que a senha está em /etc/shadow. Modernamente, as senhas criptografadas são movidas para /etc/shadow, que é acessível apenas pelo superusuário (root), com permissões típicas de rw------- ou 600. O binário do comando **passwd** (geralmente localizado em **/usr/bin/passwd**) é configurado com o atributo **setuid**, que permite que ele seja executado com os privilégios do proprietário do arquivo, neste caso, root. Isso significa que, quando um usuário comum executa passwd, o comando roda como se fosse o root, permitindo que ele modifique o arquivo /etc/shadow, que de outra forma seria inacessível.

Existem três propriedades que se pretende alcançar

- **Autenticação** - verificação que as credenciais de um utilizador ou outra entidade são válidas
- **Autorização** - controlar o acesso (definindo quem é ou não confiável) a um determinado recurso
- **Auditoria** - capacidade de analisar a informação de logging que o sistema nos dá para determinar se as políticas estão a ser cumpridas ou para detetar determinadas falhas de segurança.

6.1 Role-Based Access Control (RBAC)

Um modelo de controlo de acesso baseado em papéis, onde os utilizadores são atribuídos a papéis e os papéis têm permissões associadas. As permissões são concedidas com base no papel do utilizador, permitindo uma gestão mais eficiente dos direitos de acesso.

6.2 Discretionary Access Control (DAC)

Um modelo de controlo de acesso onde o proprietário do recurso decide quem pode aceder ao recurso e em que condições. **O proprietário pode conceder ou revogar permissões a outros utilizadores**.

Existe uma matriz de controlo de acesso (**ACL**) que **define as permissões de cada utilizador em relação a cada recurso**. As permissões podem ser concedidas ou revogadas pelo proprietário do recurso, permitindo uma gestão flexível dos direitos de acesso. A matriz de controlo de acesso é uma tabela que relaciona os utilizadores com os recursos e as permissões associadas. Cada célula da tabela indica se o utilizador tem ou não permissão para aceder ao recurso. As permissões podem ser concedidas ou revogadas pelo proprietário do recurso, permitindo uma gestão flexível dos direitos de acesso. A matriz de controlo de acesso pode ser implementada através de listas de controlo de acesso (ACLs), que são **listas associadas a cada recurso que definem quais utilizadores têm permissão para aceder ao recurso** e quais são as suas permissões. As ACLs podem ser utilizadas para definir permissões de leitura, escrita, execução e outras ações em relação aos recursos. As ACLs são uma forma comum de implementar o modelo DAC e permitem uma gestão flexível dos direitos de acesso

6.3 Mandatory Access Control (MAC)

Um modelo de controlo de acesso onde o sistema define as regras de acesso e os utilizadores não podem alterar essas regras. As permissões são baseadas em etiquetas de segurança associadas aos recursos e aos utilizadores, garantindo que apenas utilizadores autorizados podem aceder a recursos sensíveis.

6.4 Attribute-Based Access Control (ABAC)

Um modelo de controlo de acesso baseado em atributos, onde as permissões são concedidas com base em atributos do utilizador, do recurso e do ambiente. As políticas de acesso são definidas com base em expressões lógicas que avaliam os atributos dos utilizadores e dos recursos, permitindo uma gestão mais flexível e dinâmica dos direitos de acesso.

6.5 Classes de sujeitos

- **Owner**: o proprietário do recurso, que tem controlo total sobre o recurso e pode conceder ou revogar permissões a outros utilizadores.
- **Group**: um conjunto de utilizadores que partilham permissões comuns. As permissões são concedidas ao grupo e todos os membros do grupo herdam essas permissões.
- **Other/world** : utilizadores que não são proprietários do recurso e não pertencem a nenhum grupo. As permissões concedidas a "outros" são geralmente mais restritivas e limitadas

6.6 Tipos de ações

- **Read**: Permissão para ler o conteúdo de um recurso.
- **Write**: Permissão para modificar o conteúdo de um recurso.
- **Execute**: Permissão para executar um recurso, como um programa ou script.
- **Delete**: Permissão para remover um recurso.
- **Create**: Permissão para criar novos recursos.
- **Search**: Permissão para procurar recursos dentro de um diretório ou sistema de arquivos.

Permissões de diretorias é diferente de ficheiros, ou seja, Permissões da diretoria podem impedir o acesso a arquivos, mesmo com permissões definidas. Criar grupos e adicionar usuários requer permissões de root. Taxa de falsos positivos e negativos têm relação inversa em autenticação biométrica.

No Linux, um utilizador pode conseguir realizar uma operação (leitura, escrita ou execução) num seu ficheiro dentro da diretoria em que se encontra mesmo que a respetiva permissão esteja definida para todos os utilizadores do sistema.

6.7 ACL vs Capability tickets

- **ACL (Access Control List)**: lista associada a cada recurso que define quais utilizadores têm permissão para aceder ao recurso e quais são as suas permissões. As ACLs implementam o modelo DAC e permitem uma gestão flexível dos direitos de acesso.
- **Capability tickets**: bilhetes que permitem a um utilizador aceder a um recurso específico. Cada bilhete contém informações sobre o recurso, as permissões concedidas e a identidade do utilizador. Implementam o modelo MAC e permitem uma gestão mais restritiva dos direitos de acesso.

---

7. Assinaturas Digitais

Onde uma chave privada é usada para assinar um documento, e a chave pública é usada para verificar a assinatura. As assinaturas digitais garantem a **autenticidade**, **integridade** e **não-repúdio** do documento.

Exploração aprofundada das assinaturas digitais:

Uma assinatura digital é criada aplicando uma função de hash ao documento (obtendo um resumo único) e cifrando esse hash com a chave privada do remetente. O destinatário pode verificar a assinatura decifrando-a com a chave pública do remetente e comparando o hash obtido com o hash do documento recebido.

Processo detalhado:

1. O remetente calcula o hash do documento.
2. O hash é cifrado com a chave privada do remetente, formando a assinatura digital.
3. O documento e a assinatura são enviados ao destinatário.
4. O destinatário calcula o hash do documento recebido.
5. O destinatário descifra a assinatura usando a chave pública do remetente, obtendo o hash original.
6. Se os hashes coincidirem, a assinatura é válida.

Vantagens:

- **Integridade:** qualquer alteração ao documento altera o hash, tornando a assinatura inválida.
- **Autenticidade:** apenas quem possui a chave privada pode criar uma assinatura válida.
- **Não-repúdio:** o remetente não pode negar a autoria, pois só ele possui a chave privada.

Utilizações comuns:

- Assinatura de emails (ex: S/MIME, PGP)
- Assinatura de software/distribuições
- Certificados digitais (X.509)
- Blockchain e contratos inteligentes

Limitações:

- A segurança depende da proteção da chave privada.
- Não garante confidencialidade (o conteúdo do documento pode ser lido por terceiros, a menos que seja cifrado).

---

8. Permissões Octais (Unix/Linux)

Criar grupos e adicionar usuários requer permissões de superusuário (root), pois envolve modificar /etc/group.

A flag set-group-id (SGID) faz um programa ser executado com o grupo efetivo igual ao grupo dono do arquivo, não o grupo real do usuário. => efetivo != real

Valores das permissões:

- r (read) - 4
- w (write) - 2
- x (execute) - 1

Exemplos comuns:

- **000**
  - Forma simbólica: ----------
  - Significado: Nenhuma permissão para ninguém. O proprietário, o grupo e outros usuários não podem ler, escrever ou executar o arquivo.
- **100**
  - Forma simbólica: ---x------
  - Significado: Apenas o proprietário tem permissão de execução. Ninguém pode ler ou escrever o arquivo, e o grupo e outros não têm permissões.
- **200**
  - Forma simbólica: --w-------
  - Significado: Apenas o proprietário tem permissão de escrita. Ninguém pode ler ou executar o arquivo, e o grupo e outros não têm permissões.
- **400**
  - Forma simbólica: --r-------
  - Significado: Apenas o proprietário tem permissão de leitura. Ninguém pode escrever ou executar o arquivo, e o grupo e outros não têm permissões.
- **644**
  - Forma simbólica: -rw-r--r--
  - Significado: O proprietário tem permissão de leitura e escrita (4 + 2 = 6), enquanto o grupo e outros têm apenas permissão de leitura (4). Comum em arquivos de texto ou configuração.
- **755**
  - Forma simbólica: -rwxr-xr-x
  - Significado: O proprietário tem todas as permissões: leitura, escrita e execução (4 + 2 + 1 = 7). O grupo e outros têm permissão de leitura e execução (4 + 1 = 5). Usado frequentemente em scripts ou programas executáveis.
- **777**
  - Forma simbólica: -rwxrwxrwx
  - Significado: Todos (proprietário, grupo e outros) têm permissão total: leitura, escrita e execução (4 + 2 + 1 = 7). Raramente usado por questões de segurança.

Solução que garante confidencialidade, integridade e autenticidade

Essa solução utiliza cifração simétrica (AES) para confidencialidade, combinada com cifração assimétrica (RSA/ECC) para distribuir a chave de sessão, assinaturas digitais para autenticidade de origem, e hash functions para integridade. As chaves públicas de todos os destinatários são previamente compartilhadas com o remetente de forma segura (por exemplo, via um certificado confiável). O remetente gera uma chave simétrica aleatória (chave de sessão) para cifrar a mensagem usando um algoritmo simétrico eficiente,

como AES (Advanced Encryption Standard). A autenticidade dos destinatários é assegurada por uma troca inicial segura. As assumpções baseiam-se em uma infraestrutura de confiança (PKI) e na segurança das chaves privadas, garantindo que os requisitos sejam cumpridos de forma robusta. O remetente calcula um hash da mensagem original (por exemplo, usando SHA-256). Esse hash é assinado com a chave privada do remetente usando um algoritmo de assinatura digital (como RSA ou DSA).