# Administração de Sistemas (ASIST)

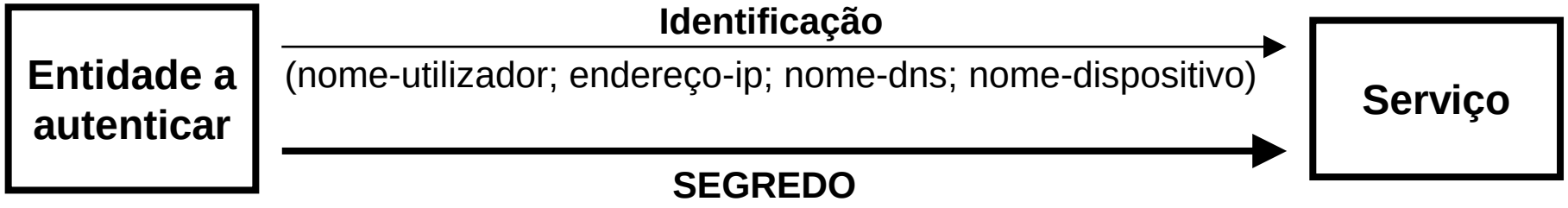
## Autenticação.
## Sistemas AAA.

Outubro de 2014

# Autenticação

O objectivo da autenticação é validar a identidade de um utilizador ou outra entidade, normalmente quando se apresenta perante um sistema ou serviço.

A forma mais simples de implementar é através de um segredo que seja apenas do conhecimento das duas entidades (segredo pré-partilhado), no caso dos utilizadores é normalmente conhecido por palavra-chave ("password").




A aplicação simples desta técnica tem alguns inconvenientes:

- não autentica o serviço (o segredo pode até estar a ser entregue a um intruso interposto)
- o segredo é transmitido (pode ser capturado por um intruso à escuta)
- há necessidade de um processo prévio de distribuição (partilha) do segredo

Esta técnica é usada no protocolo PAP ("Password Authentication Protocol"), obrigatoriamente através de uma ligação já segura.

# Autenticação com segredo pré-partilhado

O simples envio do segredo (PAP) apenas é admissível no caso de ser efetuado sobre uma ligação segura, mas essa ligação segura pressupõe que existiu uma autenticação anterior o que torna o PAP interessante apenas para situações muito específicas.

O SSH ("Secure SHell") usa frequentemente o PAP sobre uma ligação SSL/TLS já garantida através da chave pública do servidor (autenticação do servidor).

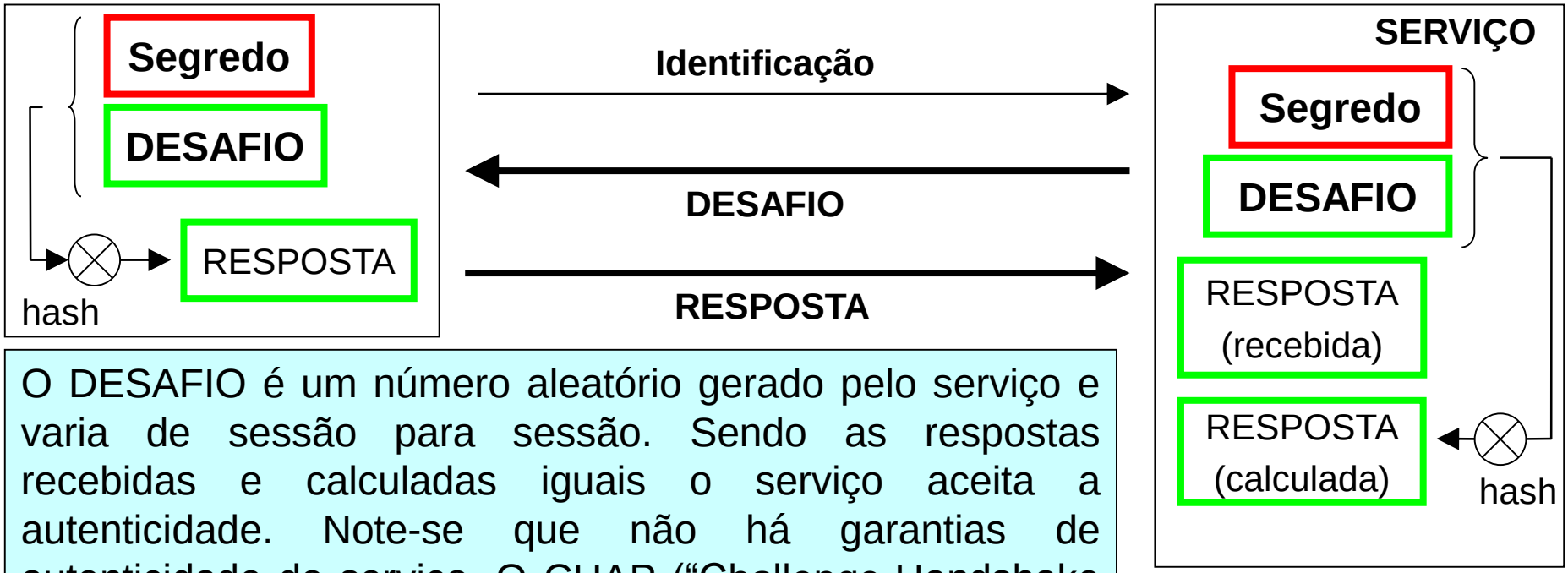
No entanto, se existe um segredo pré-partilhado o problema da autenticação pode ser resolvido de forma segura recorrendo a uma cifra simétrica.

Uma das formas propostas para distribuir de forma segura uma chave secreta de cifra simétrica entre duas entidades é através da utilização de um segredo anteriormente pré-partilhado. As cifras simétricas garantem a autenticação.

# Autenticação com segredo pré-partilhado

A utilização de uma cifra simétrica com geração da chave a partir de um segredo pré-partilhado garante a autenticidade dos dois intervenientes e a confidencialidade.

Mesmo sem recorrer a uma cifra simétrica é possível garantir a autenticação de forma segura através de um segredo pré-partilhado num processo em 3 passos.




O DESAFIO é um número aleatório gerado pelo serviço e varia de sessão para sessão. Sendo as respostas recebidas e calculadas iguais o serviço aceita a autenticidade. Note-se que não há garantias de autenticidade do serviço. O CHAP ("Challenge-Handshake Authentication Protocol") usa este mecanismo.

# Segredos pré-partilhados

Os segredos pré-partilhados permitem a autenticação segura, mas têm alguns inconvenientes:

Se o segredo não for sólido, compromete a segurança. Muitas vezes o segredo é a "password" do utilizador o que pode ser problemático.

O segredo tem de ser exclusivo de um par de entidades. Num universo de N entidades independentes cada uma delas tem de manter ( N – 1 ) segredos, num total de $C_{n,2}$ (combinações de N dois a dois) segredos que terão de existir nesse universo. Por exemplo para 100 entidades serão necessários quase 5000 segredos.

Este problema pode ser resolvido através de um KDC ("Key Distribution Center") como o do sistema KERBEROS, cada entidade necessita de apenas uma chave pré-partilhada com o KDC. O KDC cria chaves temporárias para sessões entre entidades.

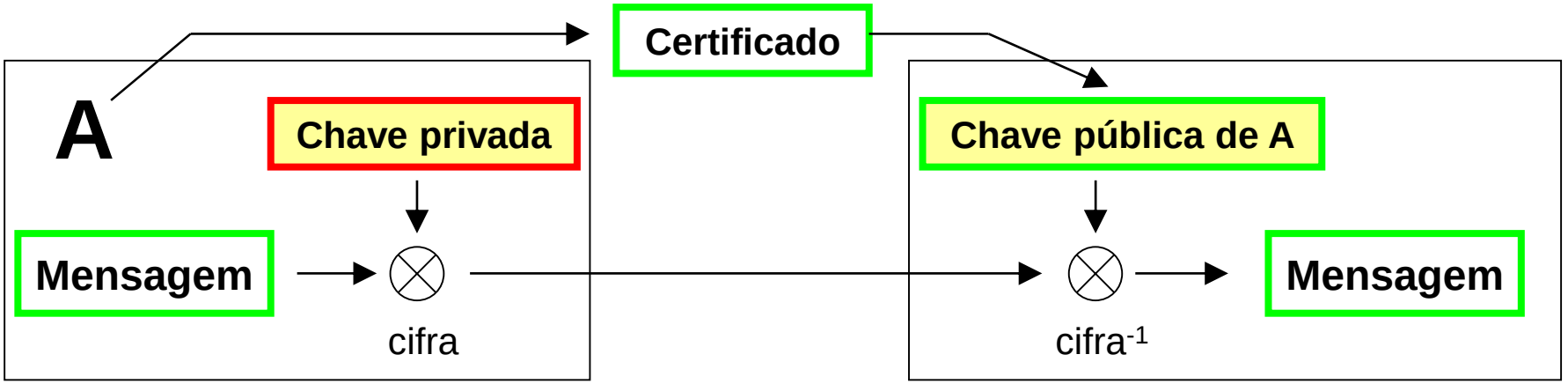
A existência de um segredo pré-partilhado pressupõe algum tipo de configuração prévia (por exemplo criar a conta de utilizador), uma entidade anteriormente desconhecida nunca se poderá autenticar.

# Autenticação com chave pública

A criptografia assimétrica resolve alguns dos problemas apontados à autenticação com segredo pré-partilhado. A chave privada é do conhecimento exclusivo de uma entidade, por isso num universo de N entidades, apenas vão ser necessárias N chaves públicas.

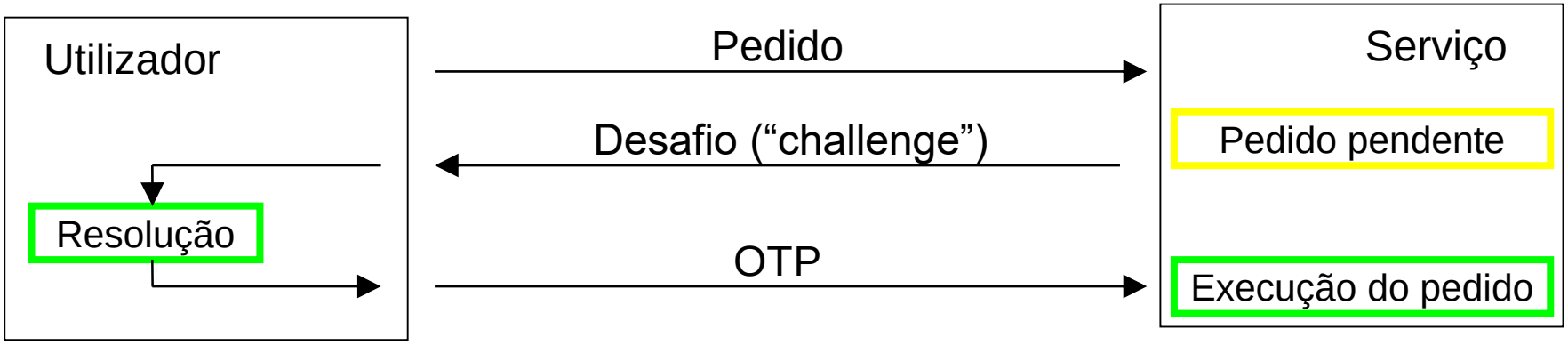
Não existem segredos pré-partilhados, a autenticidade das entidades é comprovada através dos respetivos certificados de chave pública.

A autenticidade de uma entidade A é comprovada através do envio de uma mensagem encriptada com a chave privada de A, o recetor usa a chave pública de A (obtida do certificado) para desencriptar a mensagem.



# “One-time password” (OTP)

Uma OTP é um segredo que apenas é válido uma única vez, muitas vezes para um único pedido. Este método é usado na autenticação de utilizadores quando se pretende maior segurança, muitas vezes em complemento a outros métodos.



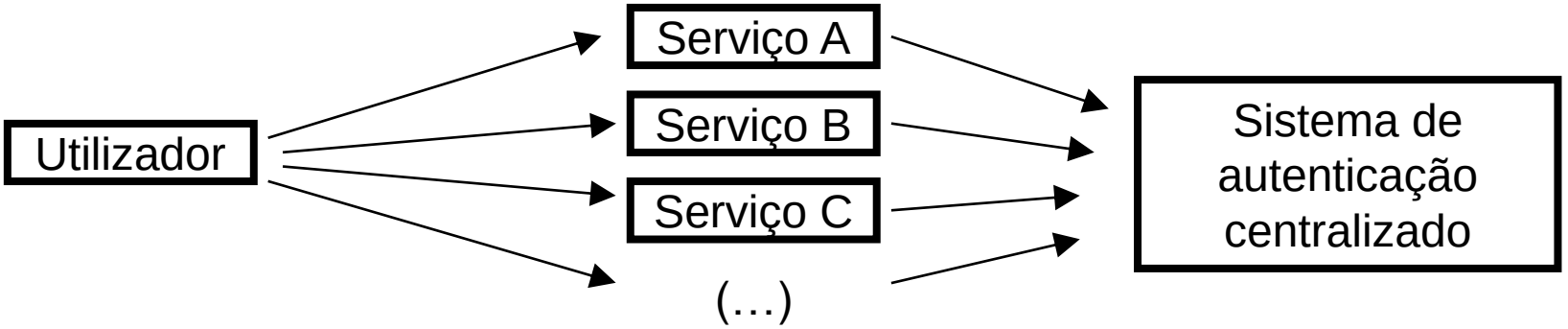

A resolução apenas é possível pelo utilizador que possui o meio adequado secreto, uma função matemática ou uma tabela eventualmente sob a forma de um “chip”.

Se for usado um meio alternativo e seguro para a transmissão do desafio, este poderá ser directamente a OTP, por exemplo através de mensagem pela rede de telemóvel. O período de validade da OTP deve ser apenas o estritamente necessário para que o utilizador complete a autenticação.

# Autenticação e autorização

Autenticação (“authentication”) refere-se a comprovar a identidade, autorização (“authorization”) refere-se a verificar se a entidade já autenticada tem permissão para aceder ao recurso ou serviço.

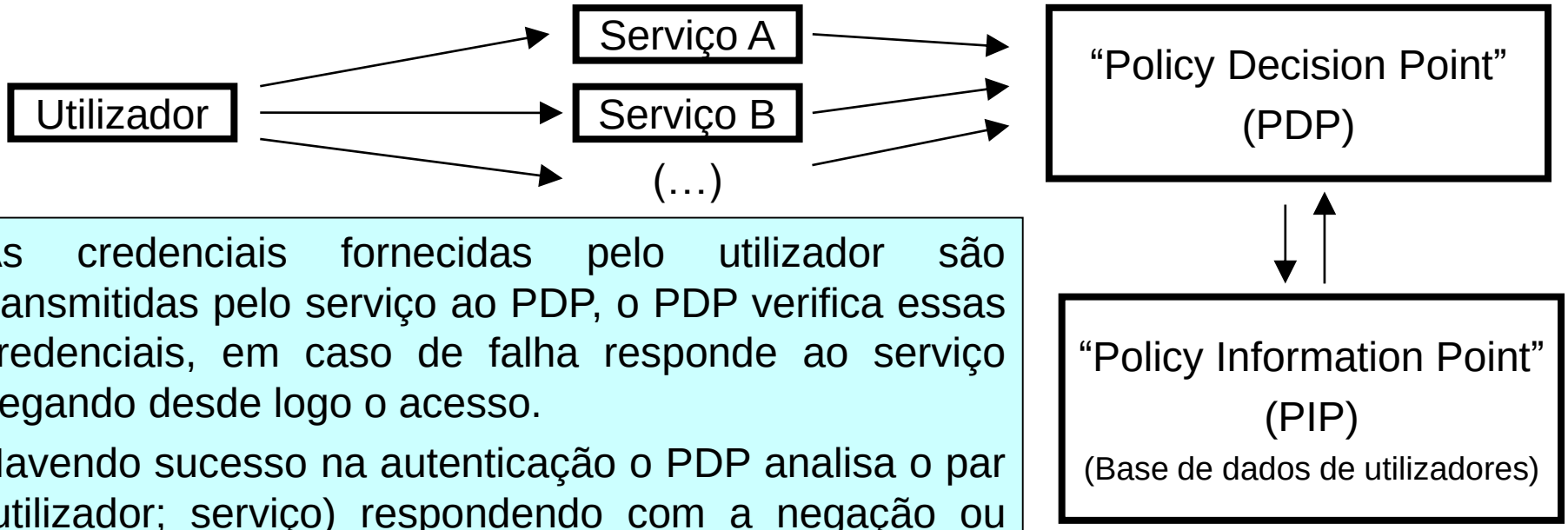
A distinção é pertinente porque é desejável que o sistema de autenticação seja comum ao maior número de serviços possível, minimizando o número de credenciais que cada utilizador necessita de manter.




O sistema de autenticação centralizado é desejável, garantindo que as credenciais são as mesmas seja qual for o tipo de serviço a que o utilizador acede dentro da organização. Contudo nem todos os serviços deverão estar acessíveis a todos os utilizadores, por isso depois da autenticação ocorre a verificação de autorização.

# Autenticação e autorização centralizados

O sistema de autenticação e autorização deve ser centralizado, recorrendo a uma base de dados central que contém a informação necessária, nomeadamente nomes de utilizadores, respectivas credenciais de acesso e lista de permissões.




As credenciais fornecidas pelo utilizador são transmitidas pelo serviço ao PDP, o PDP verifica essas credenciais, em caso de falha responde ao serviço negando desde logo o acesso.

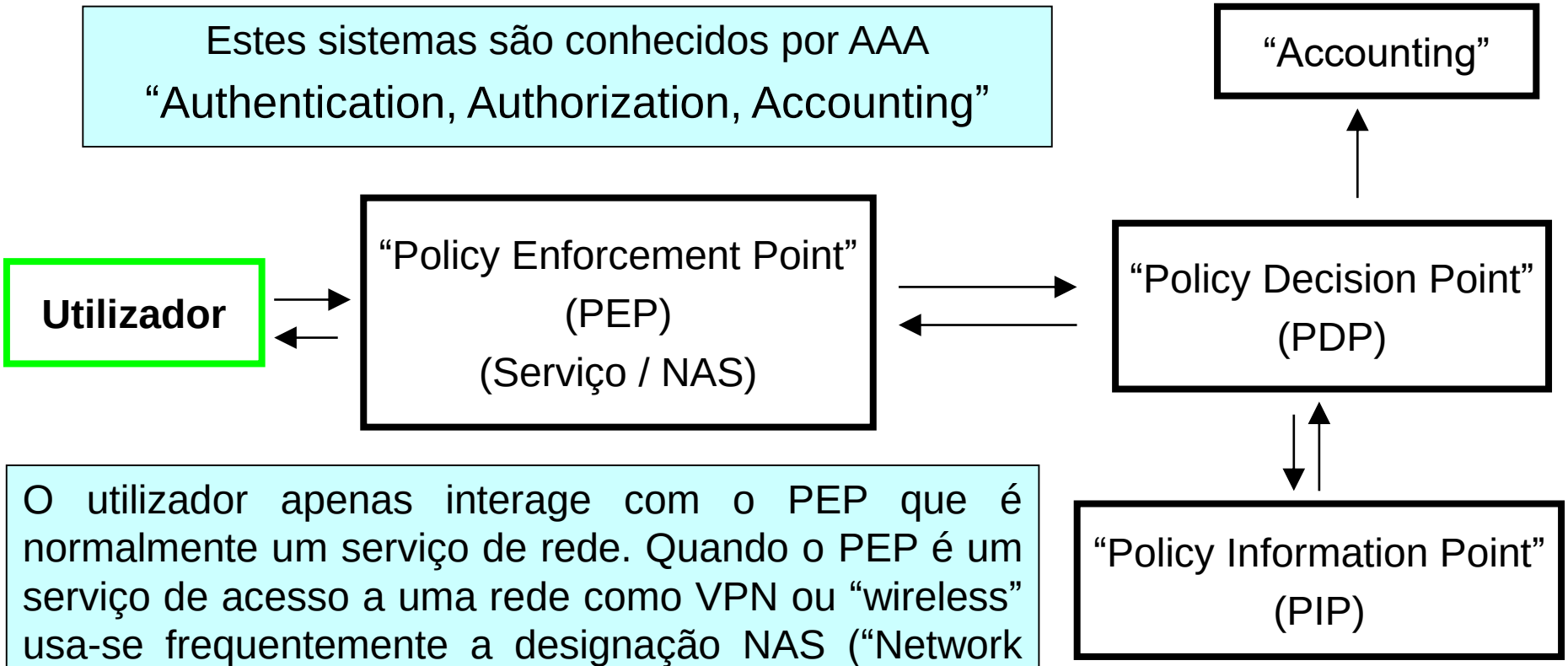
Havendo sucesso na autenticação o PDP analisa o par (utilizador; serviço) respondendo com a negação ou autorização de acordo com as listas de permissões existentes no PIP.

Posteriormente o serviço comunica a decisão ao utilizador e age em conformidade.

# Sistemas AAA

O sistema anterior é complementado com o registo (“accounting”) de acessos concedidos e negados, comunicados pelo PEP e realizados pelo PDP, garantindo o registo de atividades nesta vertente.

Estes sistemas são conhecidos por AAA
“Authentication, Authorization, Accounting”

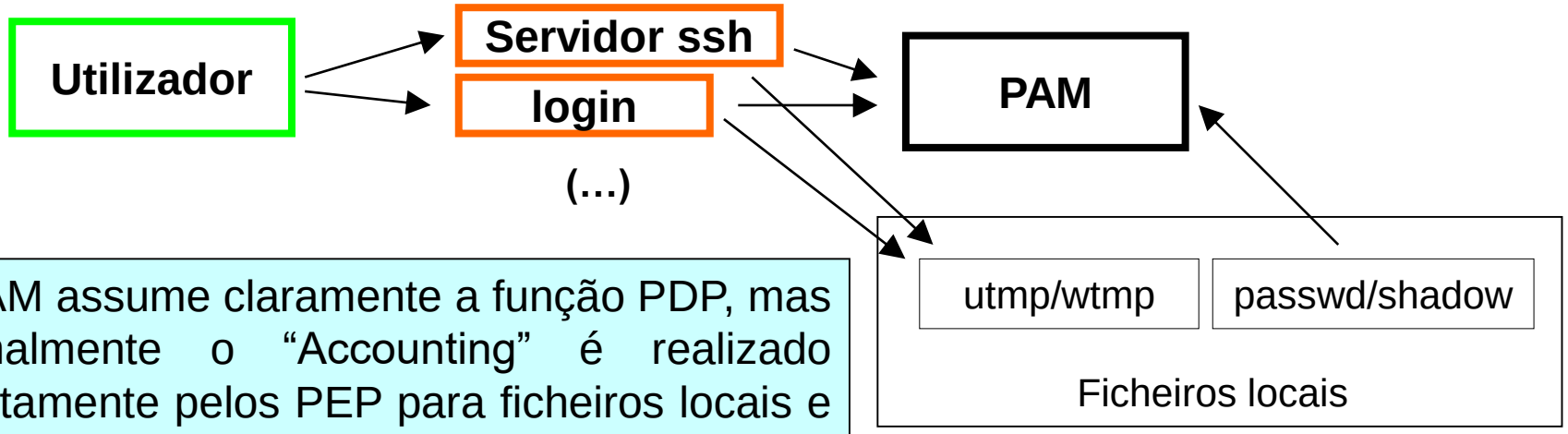



O utilizador apenas interage com o PEP que é normalmente um serviço de rede. Quando o PEP é um serviço de acesso a uma rede como VPN ou “wireless” usa-se frequentemente a designação NAS (“Network Access Service”).

# Implementações AAA – Linux + PAM

Os 4 componentes lógicos do sistema AAA (PEP ; PDP ; PIP ; Accounting) podem não corresponder a serviços independentes, ou podem ocorrer alguns desvios relativamente à arquitectura proposta.

Sistema Linux com PAM em modo de local

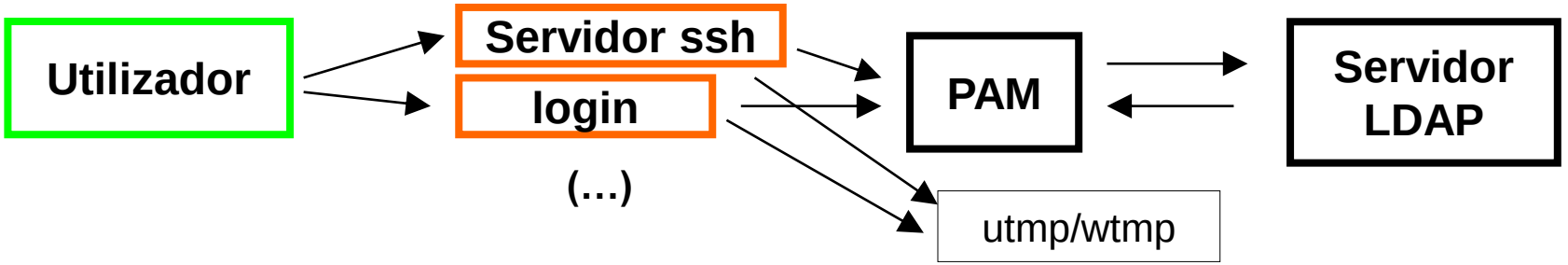



O PAM assume claramente a função PDP, mas normalmente o "Accounting" é realizado directamente pelos PEP para ficheiros locais e não pelo PDP como estava previsto.

O PIP é constituído pelos ficheiros locais "passwd/shadow".

# Implementações AAA – Linux + PAM + LDAP

Sistema Linux com PAM com recurso a um directório LDAP




O PAM continua a assumir a função PDP, o servidor LDAP parece assumir claramente a função PIP, mas a realidade pode não ser tão clara.

As trocas de informação entre o PAM e o LDAP merecem ser analisadas, normalmente o servidor LDAP não fornece credenciais para serem verificadas pelo PAM como seria de esperar na arquitectura original.

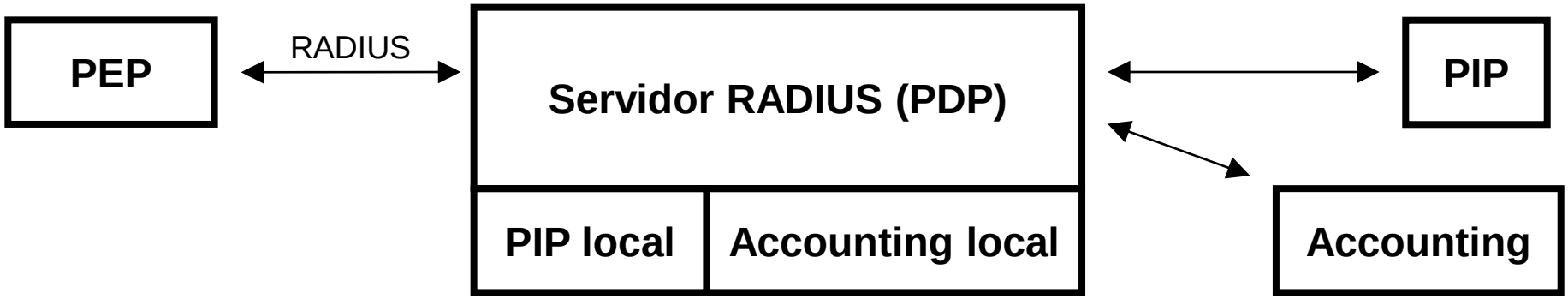
É o PAM que (através de um “bind”) envia as credenciais para serem validadas pelo servidor LDAP. O servidor LDAP está por isso a assumir a função PDP para além da função PIP.

Arquitecturas com vários PDP encadeados são perfeitamente possíveis. Nestes casos se um dos PDP nega o acesso, o resultado final será a negação de acesso.

# Protocolos AAA - RADIUS

O protocolo RADIUS (“Remote Authentication Dial In User Service”) foi concebido tendo em vista a implementação de sistemas AAA. Dá suporte directo às comunicações PEP/PDP.

Os servidores RADIUS podem usar um PIP local ou, através de módulos adicionais, podem usar protocolos apropriados para comunicar com implementações PIP externas. O mesmo se passa relativamente ao “accounting”.




O protocolo RADIUS assegura a ligação PEP/PDP através de “datagramas” UDP, existe uma chave pré-partilhada pelo administrador nas duas extremidades (PEP e PDP) que garante a autenticidade do PEP e PDP e é ainda usada para proteger as “passwords” através de uma aplicação do algoritmo MD5.

# Protocolos AAA – TACACS+

O TACACS+ (“Terminal Access Controller Access-Control System Plus”) é semelhante ao RADIUS, mas é mais usado para administradores, por exemplo no contexto de um conjunto de dispositivos de rede CISCO.

O RADIUS apenas permite verificar o acesso ao serviço através do pedido “access-request”, a autenticação e autorização são verificadas conjuntamente em resposta a esse pedido.

O TACACS+ efectua as duas verificações separadamente, primeiro “authentication-request” e depois “authorization-request”. Adicionalmente serão desencadeados novos “authorization-request” sempre que o utilizador solicita a execução de um comando.

Enquanto o RADIUS emite pedidos “accounting” apenas para o inicio e fim da sessão, o TACACS+ permite a emissão desses pedidos para cada comando executado.

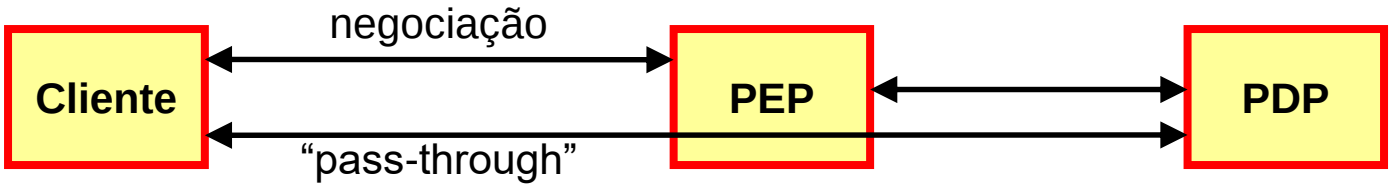
Enquanto o RADIUS usa UDP e não dá garantias de confidencialidade totais, o TACACS+ usa uma ligação TCP totalmente protegida por uma chave secreta.

# EAP (“Extensible Authentication Protocol”)

Num sistema AAA a zona de comunicações mais insegura situa-se normalmente entre os clientes e o PEP, por exemplo quando o PEP é um “access-point” de uma rede “wireless”.

O EAP permite um reforço da segurança nessa zona de uma forma flexível que não interfere com os sistemas de autenticação tradicionais. Como parte do EAP existe um procedimento de negociação entre o cliente e o PEP que permite a escolha do mecanismo de autenticação a usar.

O EAP suporta diversos mecanismos de autenticação e confidencialidade já existentes. Com EAP, o PEP pode funcionar em modo “pass-through” permitindo que alguns desses mecanismos sejam usados directamente entre o cliente e o PDP. Novos mecanismos podem ser por isso ser implementados no cliente e no PDP sem que sejam suportados no PEP.



# Autenticações EAP

O EAP suporta diversos mecanismos de autenticação, entre eles:

LEAP (“Lightweight Extensible Authentication Protocol”): usa uma implementação CHAP.

EAP-MD5: algoritmo CHAP onde é aplicado o MD5 a um segredo e um desafio

EAP-PSK: utiliza uma cifra simétrica com uma chave pré-partilhada, ou derivada de um segredo pré-partilhado como uma “password”.

EAP-TLS: utiliza certificados de chave pública (cifra de chaves assimétricas), exige que o cliente possua um certificado de chave pública.

EAP-TTLS: utiliza um “tunel” TLS entre o cliente e o PDP (o PEP funciona em “pass-through”), relativamente ao anterior tem a vantagem de não exigir um certificado de chave pública ao cliente, a respectiva autenticação é realizada posteriormente por outra via.

O PEAP (“Protected Extensible Authentication Protocol”) usa um túnel TLS de forma semelhante ao EAP-TTLS para garantir maior segurança. Difere por ser uma proteção exterior que pode ser combinada com as implementações EAP existentes.

As normas WPA (“Wi-Fi Protected Access”) incluem PEAP e EAP-TLS entre outros.

# Implementações PIP - LDAP

Na arquitectura AAA o PIP é um repositório que contém os elementos necessários à autenticação (credenciais) e definições de autorização (permissões). As implementações PDP são capazes de usar diversos tipos de PIP, sejam eles locais ou remotos, é o caso do sistema PAM (Linux) e dos servidores RADIUS.

Um tipo de repositório bastante usado para este efeito são os servidores LDAP ("Lightweight Directory Access Protocol"), caracterizam-se pela grande flexibilidade nas suas estruturas de dados e por serem optimizados para operações de consulta.

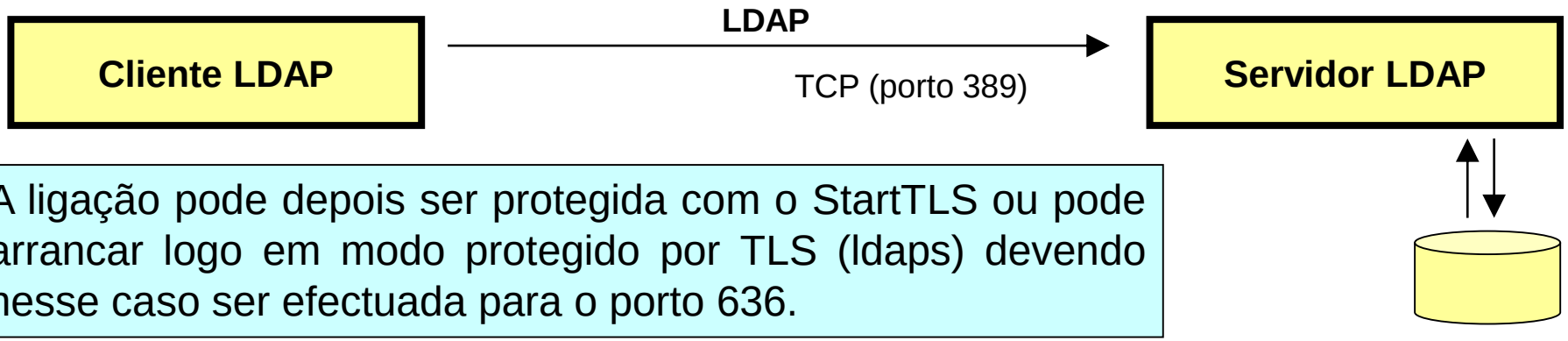
Outra caraterística interessante dos servidores LDAP é a disponibilidade de mecanismos de "mirroring" entre si.

Os servidores LDAP disponibilizam uma base de dados não relacional com estrutura em árvore em que cada ramo é um objeto, cada objeto (registo) possui um determinado número de atributos (campos) flexível. Os objetos são colocados em ramos constituindo assim uma estrutura em árvore.

Com este tipo de estrutura as operações de pesquisa seriam lentas, mas os servidores implementam a indexação dos atributos resultando em acessos muito rápidos. O "Active Directory" da "MicroSoft" é uma implementação LDAP.

# Cliente-servidor LDAP

O servidor LDAP, também designado DSA (“Directory System Agent”) é a aplicação que disponibiliza o acesso à base de dados segundo o protocolo LDAP, tipicamente através de uma ligação de rede TCP.




A ligação pode depois ser protegida com o StartTLS ou pode arrancar logo em modo protegido por TLS (ldaps) devendo nesse caso ser efectuada para o porto 636.

O acesso ao servidor é condicionado por um processo de autenticação (“bind”) com recurso às credenciais armazenadas, mas na actual versão 3 é suportado o “bind” como “anonymous” apenas para leitura de alguns dados.

O servidor tem como missão fornecer aos clientes uma interface de acesso segundo o protocolo LDAP. A base de dados “escondida” atrás do servidor é implementada com recurso a diversas tecnologias, eventualmente relacionais.

# LDAP – Objetos e Classes

O armazenamento de dados no servidor LDAP sustenta-se nos objetos que possuem atributos onde os dados são armazenados. A definição de um objeto é realizada com base em classes definidas no “shema”.

Os “shema” e as respectivas classes estão normalizados e não podem ser criados livremente.

Cada classe define quais os atributos válidos para essa classe e qual o tipo de utilização que cada atributo pode ter (tipo de dados que pode armazenar), alguns atributos podem ser obrigatórios outros opcionais. As definições de classes suportam herança, uma classe pode ser definida com base numa já existente, acrescentando-lhe mais atributos.

Existem classes estruturais e classes auxiliares, cada objeto pertence obrigatoriamente a uma única classe estrutural, mas adicionalmente pode pertencer a várias classes auxiliares. Isto permite criar objetos com atributos mais ajustados aos objetivos específicos.

Define-se o objeto com uma dada classe estrutural e depois acrescentam-se as classes auxiliares para proporcionar os atributos adicionais pretendidos.

# DIT – “Directory Information Tree”

Os vários objetos existentes na árvore são identificados através do valor de um dos seus atributos, começando pela raiz. Embora o protocolo original (X.500) proponha uma nomeação da raiz baseada nas letras que identificam os países (ISO3166), com a expansão da internet, usa-se agora mais frequentemente uma nomeação baseada em nomes de domínio DNS (RFC2247).

A RFC2247 propõe a classe “dcObject” que tem um único atributo obrigatório o “dc” (“domain component”), esta classe é auxiliar, por isso terá de ser conjugada com outra, tipicamente a classe “organization”. Em alternativa pode ser usada a classe estrutural “domain” que também tem o “dc” como único atributo obrigatório.

Segundo a RFC2247, o atributo “dc” deverá então ser usado para guardar o valor de um elemento do nome DNS da organização, por exemplo para o nome DNS “dei.isep.ipp.pt”, o nome do objeto raiz da árvore deverá ser:

dc=dei,dc=isep,dc=ipp,dc=pt

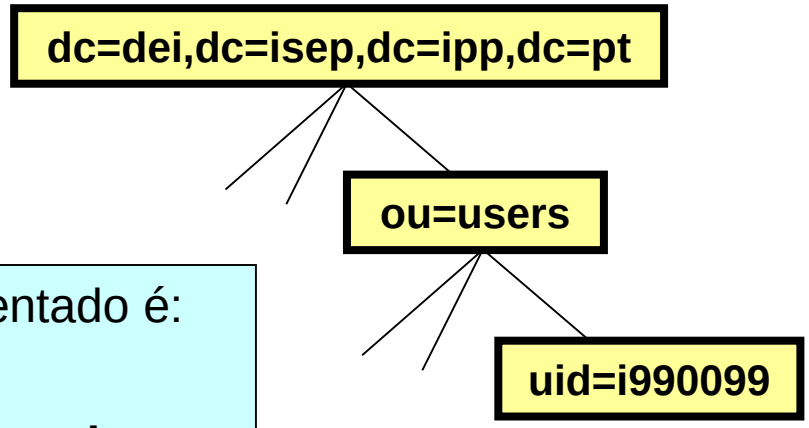
O nome que identifica o objeto na árvore designa-se “distinguished name” (dn), neste caso, tratando-se do nome do objeto raiz, designa-se “base dn” ou “root dn”.

# LDAP - “distinguished name” (dn)

Cada objeto existente na árvore é identificado através do valor de um dos seus atributos, esse par “atributo/valor” é obrigatoriamente único no ramo em que o objeto se encontra e é conhecido por RDN (“Relative Distinguished Name”), mas podem existir noutros ramos objetos com o mesmo par “atributo/valor”.

A identificação inequívoca de um objeto na árvore exige o respectivo DN que consiste no RDN concatenado por virgulas aos nomes dos objetos (ramos) desde a sua posição até à raiz.

Exemplo:




O DN do objeto com RDN “uid=i990099” apresentado é:

**uid=i990099,ou=users,dc=dei,dc=isep,dc=ipp,dc=pt**

# LDIF (“LDAP Data Interchange Format”)

O formato LDIF (RFC2849) define uma forma de expressar em texto o conteúdo de uma base de dados LDAP, permitindo importar e exportar a informação diretamente de e para servidores LDAP. A adicionalmente permite importar ordens de alteração à informação.

Exemplo (RFC2849):

```
version: 1
dn: cn=Barbara Jensen, ou=Product Development, dc=airius, dc=com
objectclass: top
objectclass: person
objectclass: organizationalPerson
cn: Barbara Jensen
cn: Barbara J Jensen
cn: Babs Jensen
sn: Jensen
uid: bjensen
telephonenumber: +1 408 555 1212
```

# LDIF – alterações de dados

Exemplo (RFC2849):

```
version: 1
# Delete an existing entry
dn: cn=Robert Jensen, ou=Marketing, dc=airius, dc=com
changetype: delete

# Modify an entry's relative distinguished name
dn: cn=Paul Jensen, ou=Product Development, dc=airius, dc=com
changetype: modrdn
newrdn: cn=Paula Jensen
deleteoldrdn: 1

# Rename an entry and move all of its children to a new location in
# the directory tree (only implemented by LDAPv3 servers).
dn: ou=PD Accountants, ou=Product Development, dc=airius, dc=com
changetype: modrdn
newrdn: ou=Product Development Accountants
deleteoldrdn: 0
newsuperior: ou=Accounting, dc=airius, dc=com
```

# LDAP – exemplo: conta de utilizador

Uma vantagem do LDAP é a possibilidade de um objeto incorporar atributos de várias classes. Por exemplo para construir um objeto adequado a guardar dados de um utilizador começamos por procurar uma classe estrutural adequada, uma opção possível é “inetOrgPerson”, derivada da classe “organizationalPerson”:

```
objectclass ( 2.16.840.1.113730.3.2.2 NAME 'inetOrgPerson‘ SUP organizationalPerson
STRUCTURAL
   MAY ( audio $ businessCategory $ carLicense $ departmentNumber $
      displayName $ employeeNumber $ employeeType $ givenName $
      homePhone $ homePostalAddress $ initials $ jpegPhoto $
      labeledURI $ mail $ manager $ mobile $ o $ pager $
      photo $ roomNumber $ secretary $ uid $ userCertificate $
      x500uniqueIdentifier $ preferredLanguage $
      userSMIMECertificate $ userPKCS12 ) )
```

# LDAP – exemplo: conta de utilizador

Se pretender usar o objeto para fornecer dados a um sistema UNIX, a classe "inetOrgPerson" não é suficiente, mas existe uma classe auxiliar que permite completar o objeto com os atributos necessários:

```
objectclass ( 1.3.6.1.1.1.2.0 NAME 'posixAccount' SUP top AUXILIARY
        DESC 'Abstraction of an account with POSIX attributes'
        MUST ( cn $ uid $ uidNumber $ gidNumber $ homeDirectory )
        MAY ( userPassword $ loginShell $ gecos $ description ) )
```

Adicionalmente se pretender que o objeto também possa ser usado por sistemas que realizem autenticação Windows, pode recorrer-se a mais uma classe:

```
objectClass (1.3.6.1.4.1.7165.2.2.6 NAME 'sambaSamAccount' SUP top AUXILIARY
  MUST ( uid $ sambaSID )
   MAY  ( cn $ sambaLMPassword $ sambaNTPassword $ sambaPwdLastSet $
       sambaLogonTime $ sambaLogoffTime $ sambaKickoffTime $
       sambaPwdCanChange $ sambaPwdMustChange $ sambaAcctFlags $
       displayName $ sambaHomePath $ sambaHomeDrive $ sambaLogonScript $
       sambaProfilePath $ description $ sambaUserWorkstations $
       sambaPrimaryGroupSID $ sambaDomainName ))
```

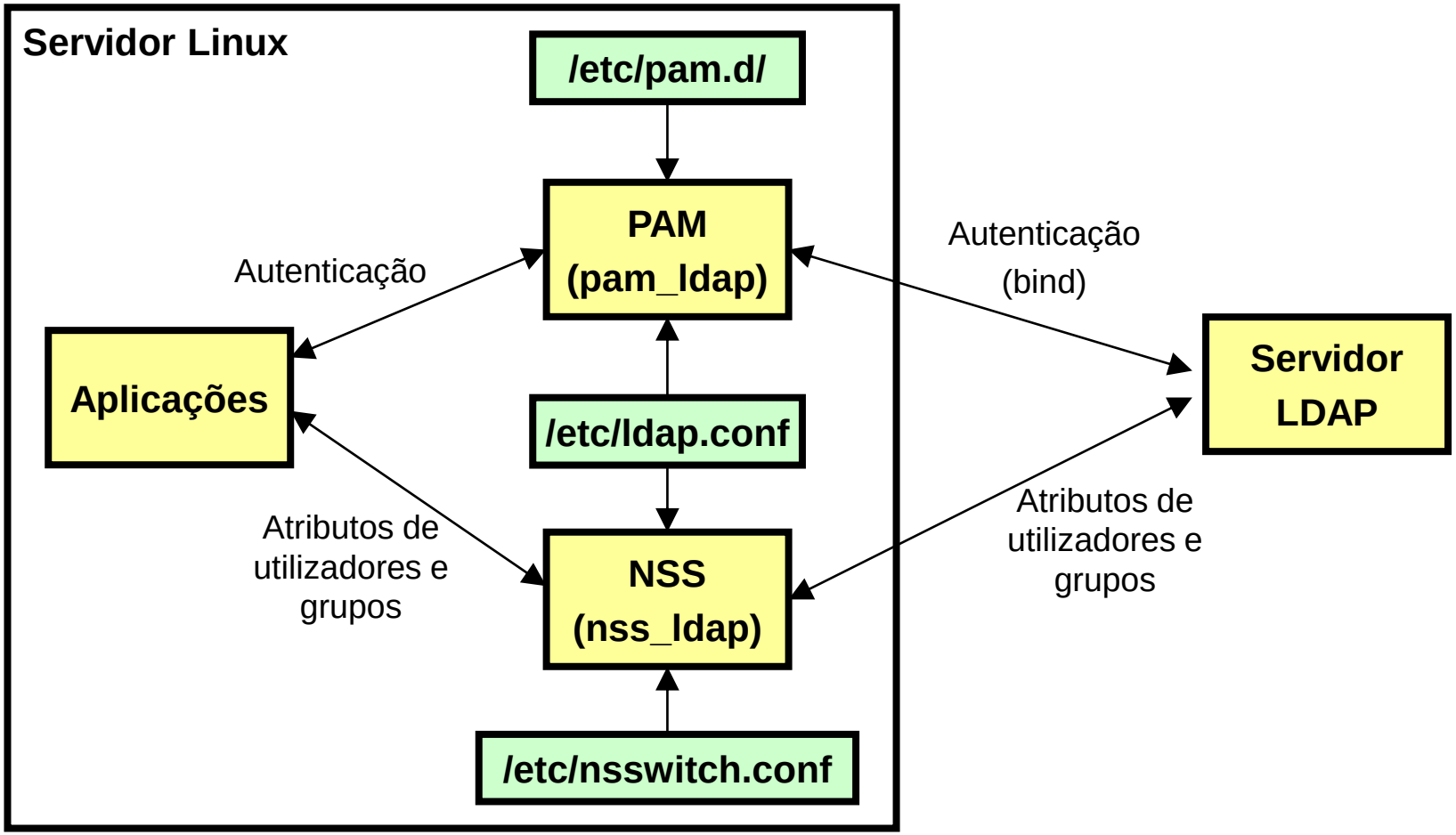
# LDAP – exemplo: conta de utilizador

Um exemplo de objeto que usa as classes anteriores seria:

```
dn: uid=i990099,ou=utilizadores,dc=exemplo,dc=pt
o: Exemplo
ou: Exemplo
cn: i990099
objectClass: inetOrgPerson
objectClass: sambaSamAccount
objectClass: posixAccount
gidNumber: 102
uid: i990099
uidNumber: 3858
loginShell: /bin/bash
displayName: Jorge
sambaSID: S-1-5-21-1309418901-1446189452-2364937762-4776
sambaPrimaryGroupSID: S-1-5-21-1309418901-1446189452-2364937762-21007
homeDirectory: /home/i990099
userPassword: {SMD5}LtvoVZZ6mgstrtaEqvDeiNrjhGXN9re4=
sambaLMPassword: EB4AAAAAAAAA234158759F68C114883
sambaNTPassword: 1CBBBBBBBBBBBBBBBB5BF27F5AF547
```

# NSS/PAM (Linux) - LDAP

A utilização da classe “posixAccount” permite que o objeto seja usado como conta de utilizador pelos sistema Linix através do NSS e do PAM:



# Referências bibliográficas

**[1] Lhamas A. (2010-2011). Aulas Teóricas de ASIST.**

**(…)**