Sistemas de Autenticação

Jaime Dias
FEUP > DEEC > MRSC > **Segurança em Sistemas e Redes**
v3.1

Sistemas de autenticação

Quem está do outro lado é mesmo quem diz ser?

- Sistema que permite verificar a identidade de uma entidade.
- Constituído por:
  - Entidade(s) autenticada(s)
  - Autenticador(es)
  - Protocolo(s) de autenticação
  - Entidade(s) de suporte ao protocolo de autenticação

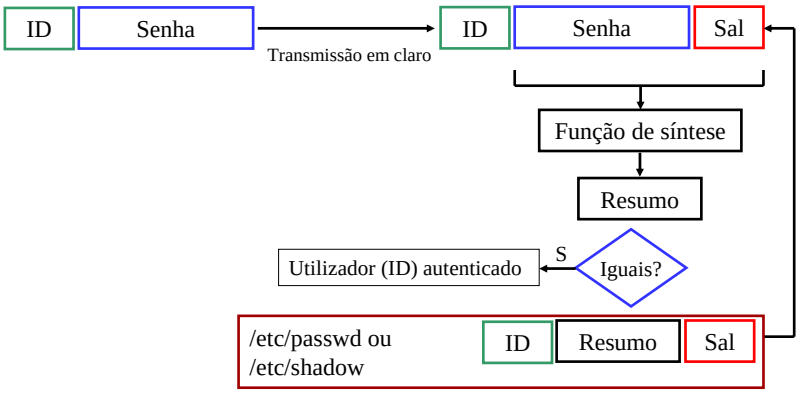
Sistemas de autenticação

- A verificação da identidade baseia-se no pressuposto que a entidade é a única capaz de enviar uma dada informação, a qual é o resultado de algo que a entidade
  - **Sabe**,
    - *Senha (= palavra chave = palavra passe = password)*
      ➔ *ex: Senhas Unix, PAP, CHAP, Kerberos*
  - **tem,**
    - *Testemunhos (Tokens)*
      ➔ *ex: cartão multibanco, S/Key, SecurID, SmartCard*
  - **é**
    - *Biometria*
      ➔ *ex: íris, impressão digital, timbre vocal*
- Para maior comodidade dos utilizadores:
  - *Single Sign on* (autentica-se uma vez no início) ➔ Ex: Kerberos
  - Biometria (não tem que se lembrar de senhas)

Senhas UNIX

- Senha enviada em claro
  - Telnet, rcp, FTP desaconselhados ➔ usar SSH, scp, sftp, …
- Autenticação
  - Utilizador envia *UserID* + senha
  - Servidor calcula resumo com base na **senha** recebida + **sal** guardado em ficheiro local
  - Se resumo igual ao guardado em ficheiro local então utilizador é autenticado
- Sal (*salt*)
  - Valor conhecido “apenas” pelo servidor, guardado juntamente com palavra chave codificada em ficheiro local (`/etc/passwd` ou `/etc/shadow`).
  - Permite obter diferentes resumos para a mesma senha

Senhas UNIX



Senhas UNIX

Implementações

- Várias versões. Uma delas:
  - Resumo é calculado através do algoritmo DES (25 iterações)
  - Mensagem (*input* do algoritmo DES na 1ª iteração): 64 bits a 0 ou 1
  - Chave simétrica: senha (8 bytes) e sal (12 bits)
    - *Senha é truncada. Apenas os 8 primeiros caracteres são utilizados!*
  - Guardado localmente: 13 bytes = 2 do sal + 11 do resumo

Senhas UNIX

Ataques

- Dedução de senhas
  - Força bruta ➔ todas as combinações têm a mesma probabilidade (exaustivo, o mais demorado)
  - Ataque baseado em dicionários ➔ testa primeiro combinações mais prováveis (derivadas de palavras de dicionários)
  - Engenharia social ➔ nome da esposa, data de nascimento, matrícula do carro…
- Para uma senha de 8 caracteres
  - Cerca de $6x10^{15}$ combinações possíveis (força bruta)
  - Pessoas tendem a escolher senhas baseadas em palavras de dicionário ➔ ≈ 1 milhão de combinações

Senhas UNIX

Ataques

- Ataque de dicionário *online*.
  - 15 tentativas por segundo
  - 18,5 horas para testar 1 milhão de combinações
  - 9,25 horas em média para descobrir a senha
- Solução: limitar o nº de autenticações falhadas

Senhas UNIX

Ataques

- Um atacante com conta no servidor poderá saber o valor do resumo+sal (`/etc/passwd`) →Ataque de diccionário *offline*.
  - Para um senha de 8 caracteres a um ritmo de 1000 tentativas por segundo
  - 17 minutos no máximo e 8,3 minutos em média para descobrir a senha
- Ataque de dicionário *offline* baseado em dados pré-calculados.
  - dados pré-calculados: tabela com possíveis resumos e uma senha por resumo
  - Duração do ataque na ordem dos segundos
  - Com um sal de 12 bits, existem 2^{12} possíveis resumos para a mesma senha. Atacante tem de pré-calcular todas as combinações de senhas para cada resumo.
  - Sem sal seriam necessários apenas 11 Mbytes de dados.
  - Com sal são necessários ≈ 50 Gbytes → actualmente já não é um impedimento
- Solução: mover senhas codificadas de `/etc/passwd` para `/etc/shadow` onde só o *root* tem acesso

PAP, CHAP, EAP

Enquadramento

- PPP – (*Point-to-Point Protocol*)
  - ligação (lógica) ponto-a-ponto.
  - permite a abstracção dos protocolos inferiores perante os protocolos de rede
- Suporta diversos protocolos de rede (IP, IPX…)
- Adapta-se a diferentes protocolos da camada inferior
  - POTS
  - RDIS
  - IP (PPTP)
  - Ethernet (PPPoE)
  - ATM (PPPoA)
- Dependendo do protocolo de autenticação utilizado, permite autenticação do cliente e eventualmente do servidor
  - PAP, CHAP (e variantes) ou EAP

PAP, CHAP

- Segredo partilhado por autenticador e entidade a autenticar
  - Unidireccional ➔ autenticador não é autenticado perante entidade a autenticar (tipicamente um utilizador)
- PAP (*PPP Authentication Protocol*)
  - Descrição do protocolo (convenção: **U**tilizador, **A**utenticador)
    - *U ➔ A: UID, password*
    - *U ➔ A: OK/notOK*
  - Senha passa em claro

PAP, CHAP

- CHAP (*CHallenge-response Authentication Protocol*)
  - Descrição do protocolo
    - *U ➔ A : authID, challenge*
    - *U ➔ A: MD5(AuthID, password, challenge)*
    - *U ➔ A : OK/notOK*
  - *Challenge* = desafio: número que deve ser usado apenas uma vez (prevenção de ataques de repetição – *replay attack*)
  - *Response* = resposta: resumo de informação que também contém desafio recebido
  - CHAP ➔ senhas guardadas em claro no servidor de autenticação
  - MSCHAPv1➔ servidor de autenticação guarda resumos
  - MSCHAPv2 ➔ servidor de autenticação também é autenticado perante utilizador

EAP

Enquadramento

- PPP foi confrontado com um problema: diversos implementadores ➔ diversos protocolos de autenticação
- Necessidade de um protocolo intermédio que suporte diversos protocolos de autenticação sem ter de alterar a norma PPP
- Solução: *Extensible Authentication Protocol*
- EAP não é um protocolo de autenticação, é um suporte a outros protocolos de autenticação. Ex.:
  - EAP-MD5
  - EAP-TLS
  - PEAP
  - EAP-TTLS
  - RSA Security SecurID EAP
  - …

Servidor de autenticação

- Permite autenticar utilizadores/máquinas perante serviços de rede e/ou aplicação.
- Para além da autenticação, um servidor pode ainda implementar mecanismos de Autorização e Contabilidade (*Authentication, Autorization and Accounting* – AAA)
  - Autorização. Controlar a que recursos o cliente pode aceder.
  - Contabilidade. Registo de acontecimentos envolvendo acessos.
    - *ex: recursos utilizados durante uma sessão: tempo, pacotes, bytes, etc.*
- Diversos servidores de autenticação
  - RADIUS, TACACS+, DIAMETER…
- O RADIUS é o *de facto* servidor de autenticação

RADIUS

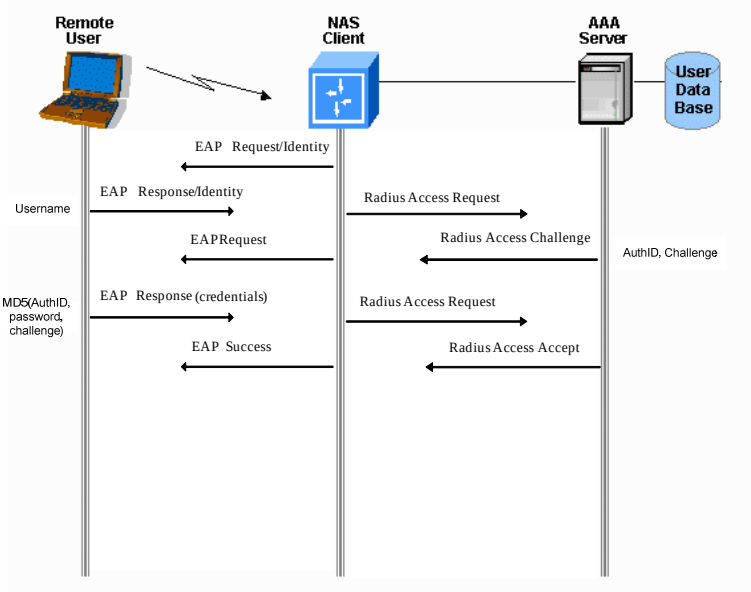
Remote Authentication Dial In User Service

- Serviço de AAA
- O seu objectivo inicial foi o de servir ISPs
  - Vários *Network Access Server* (NAS) a oferecerem ligações PPP ou SLIP
  - Em vez de cada NAS (*Network Access Server*) ter uma lista dos utilizadores/senhas autorizados, centraliza-se tudo num único servidor de *back-end* AAA (RADIUS)
- Actualmente já é usado em redes privadas com necessidade de controlo de acesso
- Suporta diversos protocolos de autenticação, (PAP, CHAP, EAP, …) e diversas bases de dados
- As mensagens trocadas entre o NAS e o servidor RADIUS são cifradas recorrendo a criptografia simétrica. A chave é previamente partilhada.

RADIUS

- Modelo cliente/servidor
- RFC 2138 ➔ protocolo de autenticação/autorização
- RFC 2059 ➔ protocolo para a contabilidade
- Portos: 1812 (UDP) autenticação/autorização, 1813 (UDP) contabilidade

RADIUS



RADIUS – Implementações

- Cisco Access Control Server 3.2
- Microsoft Internet Access Server
- Radiator
- GNU/GPL freeRADIUS

Kerberos

Enquadramento

- Uma rede suporta vários:
  - utilizadores
  - postos de trabalho
  - Servidores
  - Serviços
- Se cada serviço tiver o seu sistema de autenticação
  - gestão complexa
  - maior custo
  - maior vulnerabilidade a falhas de segurança
  - Clientes: uma autenticação por serviço
- Solução ➔ centralizar a autenticação

Kerberos

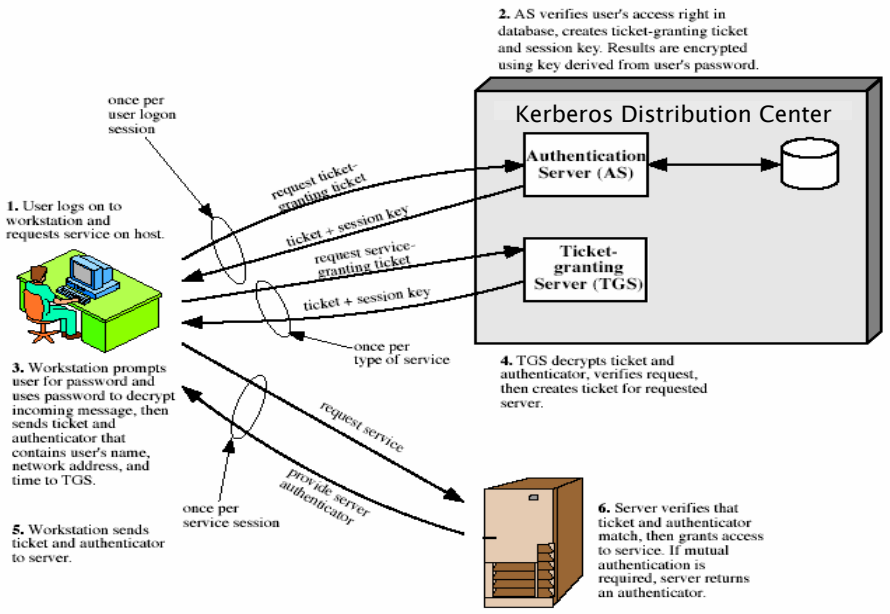
Introdução

- Serviço de autenticação central desenvolvido no MIT, no projecto Athena
- Baseado em criptografia simétrica
- *Single-sign-on:* cliente autentica-se uma vez, restantes autenticações são transparentes
- Actualmente, duas versões: 4 e 5
- Orientado aos sistemas UNIX
- Sistema de autenticação/controlo de acesso do Windows 2000/XP/2003 baseado no kerberos v5 (RFC 1510)

Kerberos

- Organizado por domínios (realm)
- Um domínio consiste:
  - num servidor Kerberos (*Kerberos Distribution Center* - KDC)
    - *composto pelo servidor de autenticação (*Authentication Server - *AS) e pelo* Ticket Granting Server *(TGS)*
  - vários clientes (*principals*) registados no KDC
    - Um cliente pode ser um utilizador, uma aplicação ou um *host*
  - Vários serviços (servidores)
  - Cada cliente e cada serviço (servidor) partilha uma chaves privada com o KDC
- Chaves dos utilizadores geradas a partir de senha
- É possível ligar múltiplos domínios ➔ os KDC têm de partilhar chaves privadas "e confiança"

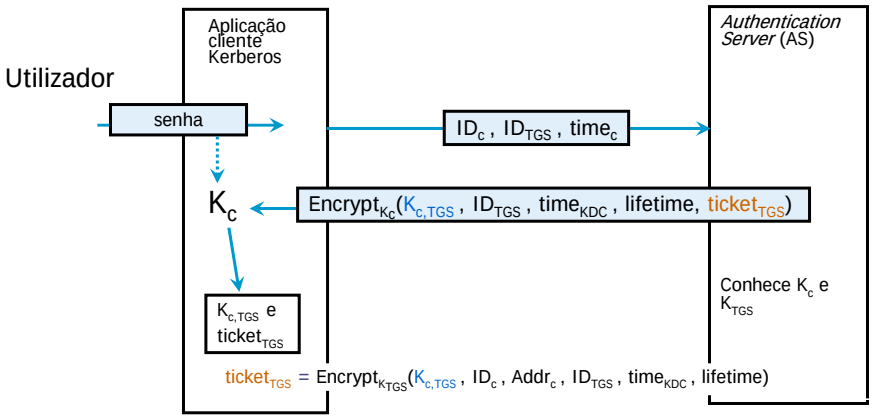
Kerberos



Chaves

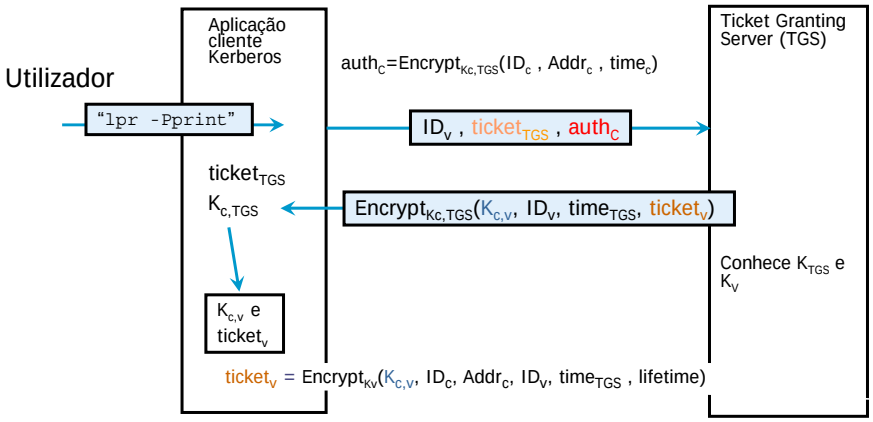
- $\mathbf{K_c}$ é a chave de longa duração do cliente C
  - Derivada da senha do utilizador
  - Conhecida pelo cliente C e pelo AS
- $\mathbf{K_{TGS}}$ é a chave de longa duração do TGS
  - Gonhecida pelo TGS e pelo AS
- $\mathbf{K_v}$ é a chave de longa duração do serviço V
  - Conhecida por V e pelo TGS
- $\mathbf{K_{c,TGS}}$ é uma chave de curta duração
  - Gerada pelo AS
  - Conhecida por C e pelo TGS (e AS)
- $\mathbf{K_{c,v}}$ é uma chave de curta duração
  - Gerada pelo TGS
  - Conhecida por C e por V (e TGS)

Fase 1 – "*Single-sign-on*"



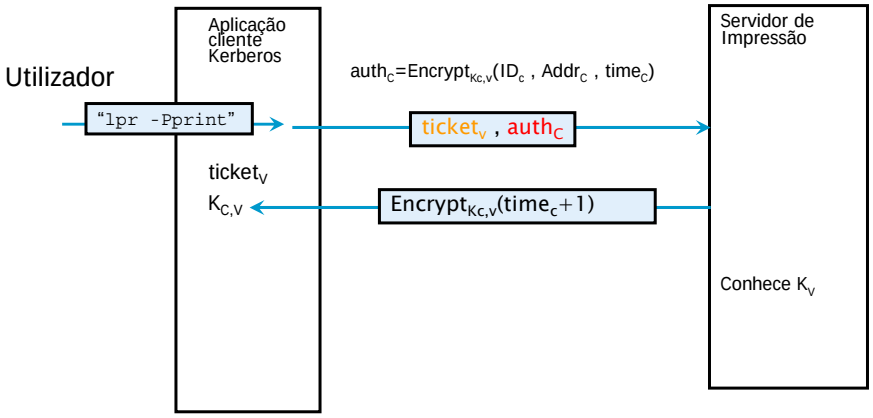
- O utilizador tem de se autenticar apenas uma vez
  - O bilhete TGS permite ao cliente obter outros bilhetes (para serviços) sem ter de se autenticar novamente.
  - O bilhete está cifrado, pelo que o cliente não consegue forjar um

Fase 2 – Obtenção de um bilhete para um serviço



- O cliente usa o bilhete TGS para obter um bilhete e uma chave de curta duração para cada serviço
- O $auth_C$ prova que o cliente conhece a chave $K_{C,V}$ contida no bilhete TGS

Fase 3 – Acesso ao serviço



- O $auth_C$ prova que o cliente conhece a chave $K_{c,v}$ contida no bilhete
- O servidor autentica-se perante o cliente pela demonstração de que conhece K_V
  - O servidor só pode cifrar a mensagem se conhecer $K_{c,v}$
  - O servidor só pode conhecer $K_{c,v}$ se conhecer K_v para decifrar o bilhete $ticket_V$
  - Pelo facto de conhecer K_v prova ser o servidor correcto.

Kerberos v4 – Resumo das mensagens trocadas

(a) Authentication Service Exchange: to obtain ticket-granting ticket
(1) C → AS: $ID_c \| ID_{tgs} \| TS_1$
(2) AS → C: $E_{K_c}\left[K_{c,tgs} \| ID_{tgs} \| TS_2 \| Lifetime_2 \| Ticket_{tgs}\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$
(b) Ticket-Granting Service Exchange: to obtain service-granting ticket
(3) C → TGS: $ID_v \| Ticket_{tgs} \| Authenticator_c$
(4) TGS → C: $E_{K_{c,tgs}}\left[K_{c,v} \| ID_v \| TS_4 \| Ticket_v\right]$
$Ticket_{tgs} = E_{K_{tgs}}\left[K_{c,tgs} \| ID_C \| AD_C \| ID_{tgs} \| TS_2 \| Lifetime_2\right]$
$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$
$Authenticator_c = E_{K_{c,tgs}}\left[ID_C \| AD_C \| TS_3\right]$
(c) Client/Server Authentication Exchange: to obtain service
(5) C → V: $Ticket_v \| Authenticator_c$
(6) V → C: $E_{K_{c,v}}\left[TS_5 + 1\right]$ (for mutual authentication)
$Ticket_v = E_{K_v}\left[K_{c,v} \| ID_C \| AD_C \| ID_v \| TS_4 \| Lifetime_4\right]$
$Authenticator_c = E_{K_{c,v}}\left[ID_C \| AD_C \| TS_5\right]$

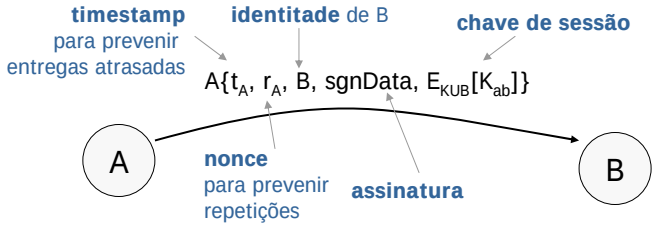
Kerberos v5

- Melhorias sobre v4
  - Algoritmos criptográficos → v4 só suporta DES
  - Protocolo de rede → v4 só suporta IPv4
  - Duração dos bilhetes maior → v4 duração máxima ≈ 21h
  - Reencaminhamento de autenticação → v4 não suporta
  - Autenticação inter-domínios → optimizada (tornada nativa) na v5
  - Derivação de sub-chaves de sessão → normalizada na v5
  - Codificação de mensagens → v5 usa ASN.1
  - Optimizações criptográficas
- Internet standard RFC 1510

Autenticação X.509

One-way authentication

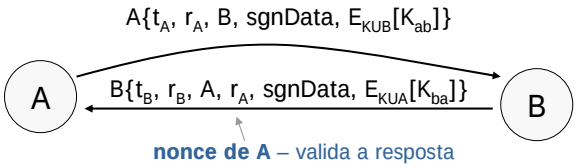
- Só A é autenticada
- 1 mensagem que
  - autentica A perante B,
  - autentica a origem (A) e o destino (B)
  - e que chegou integra
- Mensagem inclui um *timestamp*, *nonce*, identidade de B, assinatura de A e tipicamente uma chave de sessão



Autenticação X.509

Two-way authentication

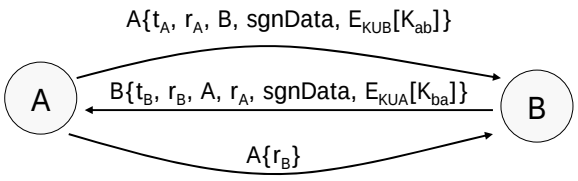
- Ambas as identidades são autenticadas
- Para além da 1ª mensagem, B devolve uma resposta que
  - confirma a recepção da 1ª mensagem por B
  - autentica B perante A
  - autentica a origem (B) e o destino (A)
  - e que chegou integra



Autenticação X.509

Three-way authentication

- Idêntico ao anterior.
- Utilizado quando os relógios de ambas as entidades não estão sincronizados



Sistemas de autenticação de senha única

One Time Password (OTP)

- Sistemas de autenticação baseados em testemunhos (hardware ou software) que geram senhas únicas a partir de chave simétrica (ou senha) e de:
  - Eventos
  - Tempo
  - Desafio-Resposta
- Só podem ser usadas uma vez → podem ser transmitidas em claro
- Exemplos:
  - S/Key
  - OPIE (idêntico ao S/Key)
  - SecurID

S/Key

- *One-time password*: gera uma senha única por cada autenticação.
- Utilizador partilha uma senha com autenticador (servidor de autenticação), a qual nunca atravessa a rede
- Cada vez que se autentica, o utilizador tem de enviar uma senha derivada da original a qual é sempre diferente (única)

S/Key

- Autenticador envia um desafio
  - um índice N que corresponde à senha única nº N
  - e uma semente (*seed*)
- Utilizador gera $\texttt{senha\_unica}_N$
  - `Senha_unica1` = MD4(`semente`, `senha`)
  - $\texttt{Senha\_unica2}$ = MD4($\texttt{Senha\_unica}_1$)
  - …
  - $\texttt{Senha\_unica}_N$ = MD4($\texttt{Senha\_unica}_{n-1}$)
- Envia $\texttt{Senha\_unica}_N$ ao autenticador
- O autenticador também calcula $\texttt{Senha\_unica}_N$ e compara com a que recebeu. Se iguais → autenticação positiva
- Em autenticações posteriores, o autenticador não pode pedir uma senha única de ordem superior (N+x) com a mesma semente.

RSA SecurID

- Desenvolvido pela RSA Security
- *One-time password*: gera uma senha diferente em cada minuto (*tokencode*)
- *Two-factor authentication* (como nos cartões multibanco)
  - Ter: o testemunho (token)
  - Saber: um PIN
- Um testemunho por utilizador
  - Pode ser um cartão, ou
  - um módulo de software (disponível para telemóveis, palmtops, PCs Windows…)

RSA SecurID

- O *tokencode* é o resumo gerado através do algoritmo de síntese *SecurID Hash* (proprietário) que tem como parâmetros de entrada:
  - Hora,
  - Chave simétrica de 64 bits (elevada entropia → dificulta ataques)
- O *tokencode* é apresentado periodicamente (1/60s) no visor do testemunho
- Sempre que o utilizador tiver que se autenticar num dado serviço introduz o UserID e uma senha: *passcode* = PIN + *tokencode* (concatenação)
- Servidor de autenticação (RSA ACE/Server) conhece o *passcode*. Pois conhece PIN, chave simétrica e hora
- Relógios sincronizados
- Aplicável a qualquer serviço de autenticação convencional do tipo utilizador/senha. Ex: webmail, telnet.

SmartCard

- Tipicamente para assinaturas digitais baseadas em PKI X.509
- Guarda certificado X.509 + chave assimétrica privada (nunca sai do *smartcard*)
- Implementa operações de assinatura digital e leitura do certificado X.509
- No caso de acesso biométrico ao smartcard → tem de guardar registos biométricos (*Match On Card* - MOC)