1.4 Segurança em sistemas distribuídos

A segurança num sistema distribuído não deve ser vista como a segurança de cada uma das suas componentes isoladas, mas sim como algo que tem uma abrangência maior em termos de políticas e mecanismos. Por exemplo, é vulgar um sistema distribuído ter apenas uma política homogénea de autenticação de utentes do sistema. Deste modo, a segurança de um sistema distribuído tem de levar em conta tanto o todo como as partes, ou seja, deverão existir políticas e mecanismos globais, de que a defesa em perímetro é um exemplo, e políticas e mecanismos locais, em cada computador, rede e equipamento de rede, que levem a cabo a defesa em profundidade.

No entanto, como a defesa dos sistemas computacionais é uma tarefa complexa e custosa, devem ser atribuídas prioridades máximas a técnicas de defesa de perímetro que criem uma primeira barreira aos potenciais atacantes, sejam os mesmos *hackers* determinados ou ciberpragas lançadas por "brincalhões".

Assim, numa primeira fase de conceção da segurança de um sistema distribuído, é preciso subdividir o mesmo em subgrupos de redes e máquinas e enquadrar esses subgrupos em domínios de segurança, definindo bem que sujeitos e em que circunstâncias os mesmos podem ter acesso a cada perímetro de segurança, tanto para efeitos de administração como para fins de exploração. Para cada domínio de segurança deve ainda ser definido claramente o conjunto de atividades autorizadas e proibidas, quer explícita quer implicitamente. Entre cada domínio de segurança, deverá haver um número mínimo de pontos de contacto que deverão ser estabelecidos por pontes de segurança de segurança (*security gateways*).

A Internet, entendida como a restante rede global sobre IP a que se liga uma rede local organizacional ou um simples computador pessoal, deverá ser considerada um domínio de segurança não controlado e, à partida, hostil.

Uma ponte de segurança é uma infraestrutura intrinsecamente segura por definição e que controla, monitoriza e limita as interações entre domínios de segurança, de forma a evitar interações ilegais ou inesperadas entre sujeitos, aplicações ou máquinas operando em domínios de segurança distintos. A nível das redes as pontes de segurança são instanciadas por *firewalls*, como veremos no Capítulo 6. Mas pode haver outros tipos de pontes de segurança, como, por exemplo, pontos de contacto controlados entre compartimentos fisicamente separados (por exemplo, entradas com duas portas nunca simultaneamente abertas, que são vulgares à entrada de instalações bancárias).

1.4.1 Riscos

Os riscos inerentes aos sistemas distribuídos são fundamentalmente os riscos relativos aos computadores e às redes que constituem o sistema. Assim, em relação aos computadores, temos os seguintes riscos:

Valores públicos	p g	número primo de grande dimensão elemento primitivo módulo p, ou de $\mathbb{Z}_p$
Chave privada Chave pública	x y	$0 \leq x < p$ $y = g^x \bmod p$
Geração da assinatura de M	γ, δ	valor aleatório secreto $k < p-1$, k e $p-1$ coprimos $k{\cdot}k^{-1} \equiv 1 \pmod{p-1}$ $\gamma = g^k \bmod p$ $\delta = (M - x\gamma)k^{-1} \bmod (p-1)$
Validação	p, g, y γ, δ	$y^\gamma \gamma^\delta \equiv g^M \pmod{p}$

Tabela 2.8: *Algoritmo de assinatura digital de ElGamal*

Valores públicos	p q g	número primo de grande dimensão (número de *bits* múltiplo de 64 e entre 512 e 1024) fator primo de $p-1$ com 160 *bits* ($2^{159} < q < 2^{160}$) q-ésima raiz de 1 módulo p ($\Rightarrow g^q \bmod p = 1$)
Chave privada Chave pública	x y	$x < p$ $y = g^x \bmod p$
Geração da assinatura do texto M	r, s	valor aleatório secreto $k \in [1, q-1]$ k^{-1}, inverso multiplicativo de k, $(k^{-1}{\cdot}k) \bmod q = 1$ $r = (g^k \bmod p) \bmod q$ $s = k^{-1} \cdot (\text{SHA-1}(M) + x{\cdot}r) \bmod q$
Validação	p, q, g, y r, s	$w = s^{-1}$ $u_1 = (w{\cdot}\text{SHA-1}(M)) \bmod q$ $u_2 = (w{\cdot}r) \bmod q$ $r \equiv ((g^{u_1}{\cdot}y^{u_2}) \bmod p) \bmod q$

Tabela 2.9: *Algoritmo de assinatura digital DSA*

1994 [11]. O DSS usa o DSA, derivado do algoritmo de assinatura de ElGamal, que funciona do modo indicado na Tabela 2.9. O DSA usa a função de síntese SHA-1 na geração e validação da síntese de um texto de dimensão variável M.

O DSA é bastante mais lento do que o RSA, nomeadamente na validação de assinaturas. No entanto, podem-se usar valores pré-computados para acelerar a geração de assinaturas. Em particular, podem-se gerar antecipadamente vários valores para k e guardar trios $\langle k, k^{-1}, r\rangle$, uma vez que o cálculo de r e k^{-1} não depende da mensagem a assinar.

Um aspeto crítico do DSA é a aleatoriedade de k. Primeiro, porque sabendo-se o k usado numa assinatura, é trivial obter a chave privada x usada na sua geração. Segundo, porque sabendo-se que o mesmo k foi usado em duas assinaturas diferentes, consegue-se obter a chave privada x mesmo sem saber k. Por isso, o uso do DSA pressupõe a existência e utilização de um bom gerador de números aleatórios.

3.5.2 Assinaturas digitais com o *smartcard* do Cartão de Cidadão

Como se viu na secção 2.10.2, as assinaturas digitais são uma forma não repudiável de autenticação de documentos. Elas permitem simultaneamente garantir a inalterabilidade de um documento e indicar a sua autoria, que não é mais do que a identidade de quem as produziu. As assinaturas digitais podem ser validadas usando a chave pública correspondente à privada que a gerou. A divulgação fidedigna das chaves públicas aos validadores de assinaturas digitais é feita através de certificados digitais dessas mesmas chaves públicas.

O *smartcard* do Cartão de Cidadão possui um par de chaves assimétricas de assinatura digital qualificada, as quais podem ser usadas por diversas aplicações para assinar documentos. O *smartcard* possui e disponibiliza um certificado X.509v3 com a chave pública de validação da assinatura digital qualificada do titular. Este certificado pode ser comunicado aos interlocutores do titular para que os mesmos possam verificar a correção e validade das suas assinaturas.

O PIN de assinatura digital do titular tem de ser enviado para o *smartcard* de cada vez que for necessário usar a chave privada do par de chaves assimétricas de assinatura digital do titular.

Por omissão, a funcionalidade de assinatura digital não está ativada quando o Cartão de Cidadão é entregue ao seu titular. Tal é feito através da publicação de um certificado de revogação, na CRL da sua EC, das credenciais de assinatura digital presentes no *smartcard*. Desta forma, o titular poderá produzir assinaturas digitais com o seu cartão, mas as mesmas não poderão ser validadas porque o validador receberá uma indicação que as credenciais usadas na geração da assinatura não estão válidas.

A ativação da funcionalidade de assinatura digital tem de ser requerida presencialmente pelo seu titular numa instituição autorizada para esse efeito. A ativação terá como efeito a remoção do certificado de revogação acima referido da CRL publicada pela sua PKI.

3.5.3 Hierarquias de certificação do Cartão de Cidadão

Os certificados de chave pública dos titulares do Cartão de Cidadão possuem as hierarquias de certificação ilustradas na Figura 3.8 e pormenorizadas nas Figuras 3.9 e 3.10. Nestas duas hierarquias os três certificados de topo são os mesmos.

Nem todos os certificados destas hierarquias estão guardados no *smartcard* de cada Cartão de Cidadão. Nomeadamente, os três do topo têm de ser obtidos por outros meios (por exemplo, no *software* disponibilizado para exploração do Cartão de Cidadão). Mais ainda, existem diversas EC emissoras de certificados pessoais (com nomes que terminam num número, como 0003 no exemplo apresentado na Figura 3.8) para além daquelas cujos certificados são disponibilizados em cada Cartão de Cidadão. Para autenticar cidadãos portugueses usando assinaturas digitais

4.2.3.1 Caso de estudo: OpenVAS e Nessus

OpenVAS (*Open Vulnerability Assessment System*[2]) é o herdeiro gratuito do Nessus[3]. As versões iniciais do Nessus eram gratuitas, mas a partir da sua versão 3 passou a ser um produto comercial. No entanto, a linha de desenvolvimento e disponibilização pública foi continuada a partir da versão 2.2, mas com o nome OpenVAS.

O OpenVAS é uma ferramenta pública de inventariação remota de deficiências de administração. Possui uma arquitetura cliente-servidor: o cliente requer o inventário sobre uma máquina ou conjunto de máquinas ao servidor e este realiza-o, enviando o resultado para o cliente. O resultado é um relatório que indica as vulnerabilidades detetadas, tanto em termos de portos de transporte acessíveis na(s) máquina(s) analisada(s) como em termos de deficiências de configuração dessa(s) máquina(s) ou dos seus serviços. O OpenVAS pode analisar simultaneamente várias máquinas, indicadas explicitamente ou através de uma máscara de rede.

Assim, o OpenVAS é uma ferramenta inestimável para automatizar a descoberta de vulnerabilidades numa rede local e orientar a sua eliminação, antes que as mesmas possam ser procuradas e exploradas por atacantes ou por código automatizado.

De modo a acomodar a evolução no reconhecimento de deficiências, o OpenVAS possui uma arquitetura baseada em *plugin*. Os testes de vulnerabilidades e deficiências de administração são feitos por blocos aplicacionais, os *plugins*, ou NVT (*Network Vulnerability Tests*), que são acrescentados dinamicamente ao motor de inventariação da aplicação. Os NVT são escritos na linguagem NASL (*Nessus Attack Scripting Language*), uma linguagem interpretada sintaticamente inspirada na linguagem C. Tal permite uma fácil atualização da lista de problemas que a ferramenta consegue identificar, o que é fundamental para garantir uma segurança efetiva dos sistemas analisados. Este funcionamento é muito semelhante ao da atualização de assinaturas de sistemas antivírus.

Hoje em dia o OpenVAS suporta igualmente interpretação de NVT escritos numa outra linguagem: OVAL (*Open Vulnerability and Assessment Language*[4]).

O projeto OpenVAS disponibiliza um sistema público de fornecimento dos NVT, o *OpenVAS NVT Feed*. Estes testes são ficheiros assinados. Contudo, qualquer pessoa pode desenvolver os seus próprios NVT para realizar testes específicos publicamente indisponíveis.

Atualmente o servidor OpenVAS executa apenas em sistemas Linux e FreeBSD. O servidor do OpenVAS usa vários outros programas públicos para realizar al-

[2] `http://www.openvas.org`
[3] `http://www.nessus.org`
[4] `http://oval.mitre.org`

Aparentemente, a certificação de nomes DNS usando assinaturas digitais e certificados de chaves públicas para validação das assinaturas poderia resolver o problema. Infelizmente, na maior parte dos casos os próprios certificados e as assinaturas são identificadas com nomes DNS, o que cria um problema recorrente.

5.3.2 Resolução errada de nomes DNS (DNS *spoofing*)

Uma outra forma de promover o uso de servidores falsos é interferir com o processo de resolução de nomes DNS. Este processo é conhecido como DNS *spoofing*. O caráter crítico da resolução de nomes DNS para endereços IP e a importância da correção nessa tradução para evitar a personificação de servidores são salientados em [92].

A especificação inicial do DNS [245] não contemplava quaisquer políticas ou mecanismos de segurança para evitar ataques à tradução de nomes. Pelo contrário, o seu desenho contemplou fundamentalmente aspetos de eficácia, eficiência e escalabilidade. Por esse facto, essa especificação possuía diversas vulnerabilidades de segurança que foram exploradas ao longo dos anos para induzir erros na resolução de nomes DNS. Como reação, apareceram diversas políticas e mecanismos que procuraram eliminar ou minimizar esses riscos.

Paralelamente, a maior parte dos servidores DNS usa uma versão do `BIND`, a realização mais popular das diversas componentes que lidam com a gestão e a resolução de nomes DNS. Esta homogeneidade tem como desvantagem o facto de tornar o sistema no seu todo mais vulnerável a erros de realização do `BIND`. Um exemplo dessa vulnerabilidade foi a propagação do *Lion Worm* em 2001, usando uma vulnerabilidade da versão 8 do `BIND`, que permitia um transbordamento de memória no código de manipulação de assinaturas TSIG (*Transaction SIGnature*), que serão explicadas mais adiante. Pior ainda, muitos servidores DNS anunciam a sua versão, o que facilita a localização de versões vulneráveis. Mais recentemente, com o advento dos servidores MS Windows 2000 e seguintes, passaram a ser também muito usados os servidores DNS dos domínios MS Windows, mas tal não melhorou significativamente o risco inerente à homogeneidade. De qualquer modo, neste texto não se irá abordar os problemas de realização do DNS, mas apenas de vulnerabilidades protocolares.

A técnica mais frequente de efetuar DNS *spoofing* passa pelo envenenamento de *caches* DNS (*DNS cache poisoning*). O princípio é simples: quando um servidor DNS pede a outro que lhe resolva um nome a resposta é autenticada de forma fraca. Qualquer um pode enviar respostas falsas para um servidor DNS e, se as mesmas forem aceites, introduzir traduções de nomes DNS erradas. Como os servidores DNS, por razões de eficiência, vão guardar a tradução em *cache* durante algum tempo, tempo esse que depende do servidor e de informação afeta ao nome, enquanto a tradução errada estiver em *cache* esta última está "envenenada" e permite efetuar DNS *spoofing*.

6.2.7.1 IP *masquerading*

A tradução de endereços IP, vulgarmente designada por IP *masquerading* ou DNAT (*Dynamic Network Address Translation*), visa esconder uma rede privada atrás de endereços públicos da sua *gateway*. Assim, quando um datagrama IP passa pela *gateway* para o exterior, o seu endereço IP de origem é mudado para um IP público da *gateway*. O porto de transporte de origem pode ser alterado para evitar colisões ao nível da *gateway*. Para isso a *gateway* guarda um mapa de traduções de endereços IP e portos de transporte interiores para portos relativos ao IP exterior da *gateway*.

A tabela de traduções é construída e mantida dinamicamente, tendo por base as solicitações de comunicação iniciadas no interior e a comunicação subsequente. Cada entrada desta tabela tem um tempo de vida próprio que é gerido com base no tipo de protocolo de transporte (com ou sem ligação) e nas trocas efetivas de datagramas que a ela dizem respeito. No caso de trocas de datagramas UDP, o tempo de vida deverá ser suficiente para permitir interações do tipo pergunta-resposta sobre UDP iniciados em máquinas interiores (um exemplo típico são pesquisas DNS). No caso de circuitos virtuais TCP, o tempo de vida deverá ser gerido como base na observação de mensagens explícitas de fim de ligação (segmentos FIN ou RST) ou ausência de trocas de datagramas.

A técnica de IP *masquerading* é particularmente útil para a segurança fornecida por uma *firewall*, porque permite que máquinas interiores à mesma iniciem contactos com o exterior mas impede o contrário. Teoricamente as máquinas exteriores não conseguem sequer ter a noção de que existem máquinas para além da *firewall*. De facto, há diversas maneiras de obter indícios relativos à sua existência (datagramas IP com *Source Route*, preenchimento diferente de cabeçalhos IP, UDP ou TCP, como vimos nos Capítulos 4 e 5, etc.). No entanto, o que interessa é que as máquinas interiores não estão visíveis nem contactáveis a partir do exterior, a menos que iniciem algum contacto com serviços localizados em máquinas exteriores.

6.2.7.2 *Port forwarding*

O mecanismo de *port forwarding*, ou *Static NAT* (SNAT), é complementar ao anterior: permite que acessos originados no exterior possam chegar até um serviço localizado numa máquina interior cujo IP não é público. Neste caso, a tabela de tradução de endereços da *gateway* não é totalmente dinâmica e mantém traduções fixas em relação a alguns portos. Nomeadamente, mantém traduções fixas entre pares ⟨IP, porto de transporte⟩ da *gateway* e pares semelhantes relativos a máquinas internas. Desta forma, a rede interna pode disponibilizar serviços públicos, uma vez que os mesmos podem ser acedidos do exterior através dos mapeamentos fixos feitos através de *port forwarding*.

Por exemplo, para a receção de *e-mail* numa máquina interior via SMTP a tabela de *port forwarding* pode indicar que um porto 25 TCP público (relativo a um dos IP

7.4 Limitações dos IDS

Os IDS possuem diversas limitações genéricas que são entraves à sua vulgarização. Nesta secção iremos indicar algumas dessas limitações, tanto operacionais como tecnológicas, e o que pode ou está a ser feito para as conseguir ultrapassar.

A primeira grande limitação operacional é a adaptação a ambientes de trabalho diversificados, adaptação essa que tem de ser feita obrigatoriamente para reduzir para um nível aceitável a taxa de falsos positivos. A diminuição desta taxa é crítica para que os administradores e operadores do IDS não percam confiança na sua utilidade, mas essa diminuição deverá ser feita de modo cuidadoso para não aumentar a taxa dos falsos negativos (que, ainda por cima, é difícil de calcular). A adaptação pode ainda ser muito complicada em ambientes de trabalho muito anárquicos, como universidades ou instituições de investigação ligadas às universidades. Resumindo, é difícil obter uma solução de IDS "chave na mão", cada instalação de IDS tem de ser afinada localmente e essa afinação não é trivial.

A segunda grande limitação operacional é a escalabilidade. A escalabilidade dos HIDS é naturalmente complexa porque cada máquina possui um sistema próprio ou uma configuração do sistema ou dos seus serviços própria, o que dificulta a aplicação do mesmo IDS e do mesmo tipo de análise a todas as máquinas. Os NIDS, que em teoria se adaptam melhor a este problema, não escalam facilmente em redes de muito alto débito ou comutadas por exigirem arquiteturas paralelas ou distribuídas de deteção e fusão de dados, o que ainda está em grande medida no domínio da investigação.

A terceira grande limitação operacional é a falta de uma taxionomia universal de caracterização de ataques e intrusões. Esta taxionomia é importante para fornecer informação útil aos operadores e administradores do sistema, mas também para permitir a interoperabilidade com outros IDS ou demais meios de segurança, como *firewalls*. Este aspeto está atualmente a ser estudado pelo grupo de trabalho IDWG[2] (*Intrusion Detection Workgroup*) da IETF (*Internet Engineering Task Force*), que tem por objetivo a normalização da linguagem de troca de dados relativos à operação dos IDS. A solução atualmente em estudo é baseada em XML (*Intrusion Detection Message Exchange Format Data Model*) e usa um protocolo próprio para a troca de dados (IDXP – *Intrusion Detection eXchange Protocol*). A classificação de vulnerabilidades exploradas nos ataques é feita usando identificadores de vários dicionários, nomeadamente a base de dados da SecurityFocus.com[3] ("Bugtraq") e a lista CVE[4] (*Common Vulnerabilities and Exposures*). Existem também esforços para definir interfaces entre IDS e outros sistemas, como a interface SAMP (*Suspicious Activity Monitoring Protocol*) proposta pela CheckPoint, no âmbito do OPSEC (*Open Platform for Secure Enterprise Connection*), que permite que um IDS consiga controlar

[2] `http://www.ietf.org/html.charters/idwg-charter.html`
[3] `http://www.securityfocus.com/archive`
[4] `http://www.cve.mitre.org`

O túnel PPTP permite, como anteriormente se viu, que entre uma máquina-cliente, ligada a um ISP ou a uma LAN, e um servidor de acesso PPTP (PPTP RAS (*Remote Access Server*)) exista um túnel, eventualmente seguro, que suporte uma interação de nível 2 sobre IP, ou seja, através da Internet. Tal permite que a máquina-cliente use a LAN a que o RAS está ligado como se entre as duas existisse uma ligação física direta. Assim, uma ligação PPTP permite, quando dotada de segurança, o que veremos já de seguida, construir uma VPN de nível 2.

Uma VPN de nível tão baixo tem vantagens e desvantagens operacionais. As vantagens são existir um comportamento em rede como se a máquina remota estivesse diretamente ligada à rede a que se liga via VPN. Por exemplo, a máquina remota consegue receber tramas em difusão ao nível 2, tráfego NetBEUI nativo (não encapsulado em IP), etc., o que não é trivial ou possível com as VPN de nível superior. Porém, esse tráfego pode sobrecarregar inutilmente a VPN e ser especialmente indesejável se a mesma for suportada por um meio de transmissão de baixo débito.

8.9.2 Segurança numa interação PPTP

Os túneis PPTP permitem a cifra do tráfego PPP que através deles flui, bem como a autenticação dos extremos dos túneis. Mas a cifra só é aplicada, se o for, após o sucesso do protocolo de autenticação escolhido. Ou seja, a cifra nunca protege o protocolo de autenticação nem as demais negociações de configuração do PPP. Tal é facilmente observável na captura de datagramas apresentada na Figura 8.7, onde só nas tramas PPP com dados os mesmos circulam cifrados (datagramas com o número de ordem 43 e 46, com a legenda `Compressed Data`).

O PPTP, sendo fundamentalmente uma extensão do PPP que permite a troca de tramas sobre redes IP, permite os protocolos-base de autenticação do PPP (PAP, CHAP), mais algumas extensões do metaprotocolo EAP (*Extensible Authentication Protocol*) [63], como o EAP-MD5-Challenge e permite uma interação fácil com servidores de autenticação de máquinas ou domínios Microsoft através dos protocolos MS-CHAPv1 [42] e MS-CHAPv2 [38].

Contudo, os dois primeiros, PAP e CHAP, não devem ser usados para instanciar uma VPN PPTP, porque não permitem ativar a cifra da comunicação PPTP (ver MPPE mais adiante). Mais ainda, o PAP não deve de forma alguma ser usado, porque troca a senha de acesso em claro (ver Figura 8.9).

O PAP, o CHAP e o MS-CHAPv1 também devem ser evitados, dado que não autenticam o servidor de VPN (ou o servidor de autenticação usado pelo servidor de VPN). Este facto permite que um cliente seja enganado por um atacante que personifique o servidor legítimo. Tais ataques de personificação podem ser usados para capturar senhas dos utentes enganados ou respostas legítimas desses utentes a desafios enviados pelo atacante. Por exemplo, o MS-CHAPv1 permite a alteração da senha do utente e um servidor falso pode forçar essa alteração indicando que a senha anterior expirou.

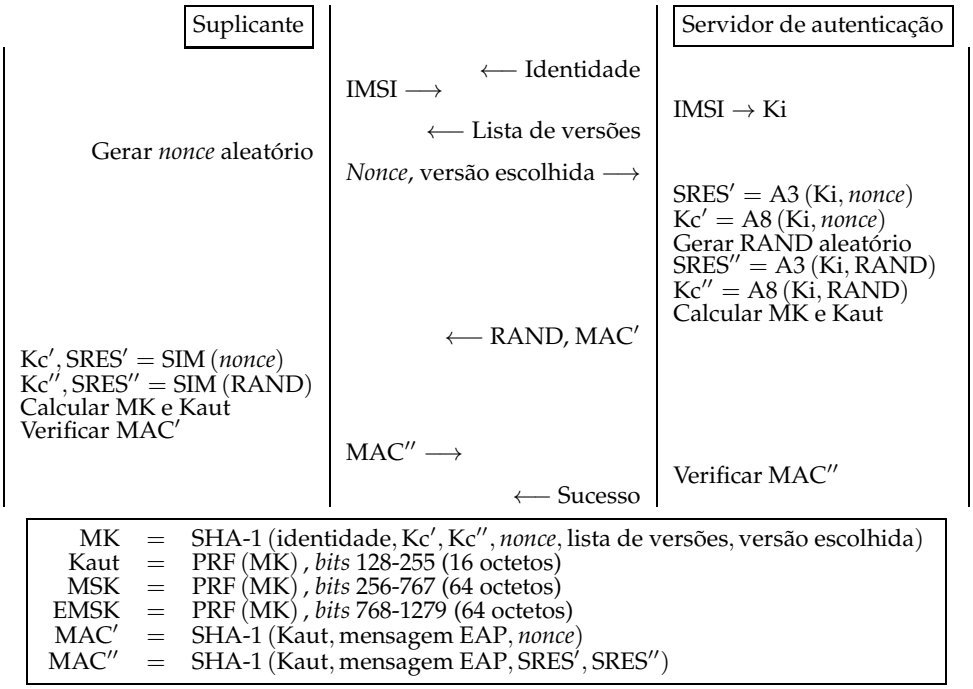

Figura 9.26: *Protocolo de autenticação EAP-SIM*

9.9.6 EAP-SIM e EAP-AKA

O EAP-SIM [229] consiste numa adaptação do paradigma de autenticação usado no GSM, descrito na secção 10.3.2.2, para a arquitetura 802.1X. O suplicante usa um *smartcard* com um módulo SIM e usa-o para gerar chaves como respostas a desafios, quer criados pelo servidor de autenticação e transmitidos pelo autenticador, como criados por si próprio.

O EAP-AKA é similar ao EAP-SIM mas orientado para a autenticação com módulos USIM, usados em UMTS.

A chave Kc derivada a partir de uma autenticação GSM é usada em GSM para cifrar diretamente as tramas trocadas, mas no caso do 802.1X tal não acontece: é apenas usada para derivar outras chaves, nunca é usada diretamente para cifrar dados trocados. Nomeadamente, são usadas duas chaves Kc para criar uma chave mestra (MK – *Master Key*), a qual será, então, usada para gerar outras chaves, em particular MSK (ver Figura 9.26). Para gerar as chaves a partir de MK é usada a seguinte função PRF, geradora de uma sequência pseudoaleatória:

10.3.6 Metaprotocolos de autenticação

Os metaprotocolos de autenticação não são mais do que protocolos que estabelecem as regras gerais de interação entre as partes envolvidas e mensagens genéricas capazes de encapsular mensagens pertencentes a instanciações do metaprotocolo. Os exemplos mais interessantes que importa referir são o 802.1X [4], o *SAML Web Browser SSO Profile* [134] e o OpenID [88]. O primeiro ganhou relevo com as redes sem fios 802.11 e foi descrito no Capítulo 9 (secção 9.8). Os dois últimos são usados sobretudo em aplicações *Web* para autenticar clientes perante servidores usando os serviços de um IdP.

10.3.6.1 Caso de estudo: *SAML Web Browser SSO Profile*

O *SAML Web Browser SSO Profile* é um perfil de autenticação criado pela OASIS (*Advancing Open Standards for the Information Society*) para autenticar utentes que usam um navegador (UA – *User-Agent*) perante um servidor HTTP, do qual pretendem obter um recurso (SP – *Service Provider*). A autenticação é efetivamente realizada por um fornecedor de identidade (IdP – *Identity Provider*), em que o SP confia e que, supostamente, será igualmente capaz de identificar e autenticar o utente. Este perfil de autenticação usa mecanismos próprios do HTTP para redirigir mensagens através do UA e usa asserções SAML para transmitir atributos de identidade do utente do IdP para um SP.

Para além disso, e para atingir a funcionalidade de SSO, se vários SP usarem o mesmo IdP, após uma primeira identificação e autenticação do utente no IdP, o mesmo não precisará, num futuro próximo (que depende de inúmeros fatores) de se autenticar novamente perante o IdP. Por outras palavras, existe uma sessão SSO que é mantida entre o UA e o IdP que, após a sua criação e até à sua terminação, poupa ao utente a realização de inúmeras operações de autenticação para aceder a vários SP que explorem o mesmo IdP.

O tempo durante o qual permanece ativa a sessão SSO depende do tempo de manutenção, no UA e no IdP, de um contexto relativo a essa sessão. Este é formado por *cookies* HTML, no caso do UA, e por um contexto que relaciona um *cookie* com um conjunto de atributos de identidade, no IdP. O primeiro que invalidar ou remover um destes contextos, seja por que razão for, terminará a sessão SSO.

Neste modelo todas as funcionalidades de identificação, autenticação e SSO do utente estão centralizadas no IdP. É esta componente que é responsável por obter, manter e fornecer atributos de identidade de utentes a um SP, sendo ela igualmente a responsável por realizar a autenticação dos utentes.

O protocolo de autenticação é iniciado pelo SP quando recebe um pedido de um UA relativo ao qual não conhece os necessários atributos de identidade do seu utilizador (mensagem 1 da Figura 10.26). Logo, o acesso ao recurso é adiado até que