



CAPÍTULO

17

Segurança na Web

- 17.1 Considerações sobre segurança na Web
- 17.2 Secure Socket Layer e Transport Layer Security
- 17.3 Secure Electronic Transaction

- 17.4 Leitura e sites Web recomendados
- 17.5 Principais termos, perguntas de revisão e problemas

*Use sua mente.
Acorde para a realidade.*

— Da canção “I've got you under my skin”, de Cole Porter

PONTOS PRINCIPAIS

- ◆ O Secure socket layer (SSL) oferece serviços de segurança entre TCP e aplicações que usam TCP. A versão-padrão da Internet é chamada de Transport Layer Service (TLS).
- ◆ O SSL/TLS oferece confidencialidade usando criptografia simétrica e integridade de mensagens usando um código de autenticação de mensagens.
- ◆ O SSL/TLS inclui mecanismos de protocolo para permitir que dois usuários TCP determinem os mecanismos e serviços de segurança que eles usarão.
- ◆ O Secure Electronic Transaction (SET) é uma especificação aberta de criptografia e segurança, projetada para proteger transações com cartão de crédito na Internet.

Praticamente todas as empresas, a maioria dos órgãos do governo e muitos indivíduos agora possuem sites Web. O número de indivíduos e empresas com acesso à Internet está aumentando rapidamente e todos eles têm navegadores Web gráficos. Como resultado, as empresas demonstram grande interesse em utilizar a Web para comércio eletrônico. Mas a realidade é que a Internet e a Web são extremamente vulneráveis a riscos de vários tipos. À medida que as empresas percebem essa realidade, a demanda por serviços Web seguros aumenta.

O assunto de segurança na Web é muito amplo e pode facilmente ocupar um livro inteiro (vários são recomendados no final deste capítulo). Neste capítulo, começamos com uma explicação sobre os requisitos gerais para segurança na Web e depois focalizamos dois esquemas padronizados que estão se tornando cada vez mais importantes como parte do comércio na Web: SSL/TLS e SET.

17.1 CONSIDERAÇÕES SOBRE SEGURANÇA NA WEB

A World Wide Web é fundamentalmente uma aplicação cliente/servidor trabalhando sobre a Internet e intranets TCP/IP. Dessa maneira, as ferramentas de segurança e as técnicas explicadas até aqui neste livro são relevantes para a questão de segurança na Web. Mas, conforme indicado em [GARF97], a Web apresenta novos desafios geralmente não apreciados no contexto da segurança de computador e de rede.

- A Internet tem duas direções. Ao contrário dos ambientes de publicação tradicionais, até mesmo sistemas de publicação eletrônica envolvendo teletexto, resposta de voz, ou fax back, a Web é vulnerável a ataques aos servidores Web pela Internet.
- A Web está cada vez mais servindo como uma vitrine com alta visibilidade para as informações corporativas e de produtos e como plataforma para transações comerciais. Reputações podem ser prejudicadas e dinheiro pode ser perdido se os servidores Web forem subvertidos.
- Embora os navegadores Web sejam muito fáceis de usar, os servidores Web sejam relativamente fáceis de configurar e gerenciar e o conteúdo da Web esteja cada vez mais fácil de desenvolver, o software que dá suporte a tudo isso é extraordinariamente complexo. Esse software complexo pode ocultar muitas falhas de segurança em potencial. A história recente da Web está repleta de exemplos de sistemas novos e atualizados, devidamente instalados, que são vulneráveis a uma série de ataques à segurança.
- Um servidor Web pode ser explorado como uma plataforma de lançamento para o sistema de computadores de uma empresa ou de um órgão governamental. Quando o servidor Web é contaminado, um atacante pode ser capaz de obter acesso a dados e sistemas que não fazem parte da Web, mas que estão em máquinas conectadas localmente ao servidor.
- Os usuários ocasionais e não treinados (em questões de segurança) são clientes comuns de serviços baseados na Web. Esses usuários não estão necessariamente cientes dos riscos de segurança que existem e não possuem as ferramentas ou o conhecimento para tomar contramedidas eficazes.

Ameaças à segurança na Web

A Tabela 17.1 oferece um resumo dos tipos de ameaças à segurança enfrentadas no uso da Web. Uma maneira de agrupar essas ameaças é em termos de ataques passivos e ativos. Ataques passivos incluem acesso não autorizado ao tráfego de rede entre navegador e servidor e obtenção de acesso a informações em um site Web que deveria ser restrito. Ataques ativos incluem simulação de outro usuário, alteração de mensagens em trânsito entre cliente e servidor e alteração de informações em um site Web.

Outra maneira de classificar as ameaças de segurança na Web é em termos do local da ameaça: servidor Web, navegador Web e tráfego de rede entre navegador e servidor. As questões de segurança de servidor e navegador entram na categoria de segurança de sistema de computador; a Parte Quatro deste livro focaliza a questão da segurança de sistema em geral, mas também se aplica à segurança de sistemas Web. As questões de segurança de tráfego entram na categoria de segurança da rede e são focalizadas neste capítulo.

Técnicas de segurança de tráfego na Web

Diversas técnicas para oferecer segurança na Web são possíveis. As diversas técnicas que têm sido consideradas são semelhantes nos serviços que oferecem e, até certo ponto, nos mecanismos que utilizam, mas diferem com relação ao seu escopo de aplicabilidade e seu local relativo dentro da pilha de protocolos TCP/IP.

A Figura 17.1 ilustra essa diferença. Um modo de oferecer a segurança na Web é usar IP Security (Figura 17.1a). A vantagem de usar IPSec é que ele é transparente para os usuários finais e aplicações e oferece uma solução de uso geral. Além disso, o IPSec inclui uma capacidade de filtragem pela qual somente o tráfego selecionado precisa ser submetido à etapa adicional do processamento IPSec.

Tabela 17.1 Comparação de ameaças na Web [RUBI97].

	Ameaças	Consequências	Contramedidas
Integridade	<ul style="list-style-type: none"> Modificação de dados do usuário Navegador cavalo-de-tróia Modificação de memória Modificação de mensagens em trânsito 	<ul style="list-style-type: none"> Perda de informações Comprometimento da máquina Vulnerabilidade a todas as outras ameaças <p>Somas de verificação (checksums) criptográficas</p>	
Confidencialidade	<ul style="list-style-type: none"> Registro não autorizado de tráfego na rede Roubo de informações do servidor Roubo de dados do cliente Informações sobre configurações de rede Informações sobre qual cliente fala com o servidor 	<ul style="list-style-type: none"> Perda de informações Perda de privacidade 	Criptografia, proxies Web
Negação de serviço	<ul style="list-style-type: none"> Encerramento de threads do usuário Inundação da máquina com solicitações falsas Preenchimento da capacidade total do disco ou da memória Isolamento da máquina por ataques de DNS 	<ul style="list-style-type: none"> Interrupção Incômodo Impede que o usuário realize o trabalho 	Difícil de impedir
Autenticação	<ul style="list-style-type: none"> Simulação de usuários legítimos Falsificação de dados 	<ul style="list-style-type: none"> Falsificação da identidade do usuário Crença de que informações falsas são válidas 	Técnicas criptográficas

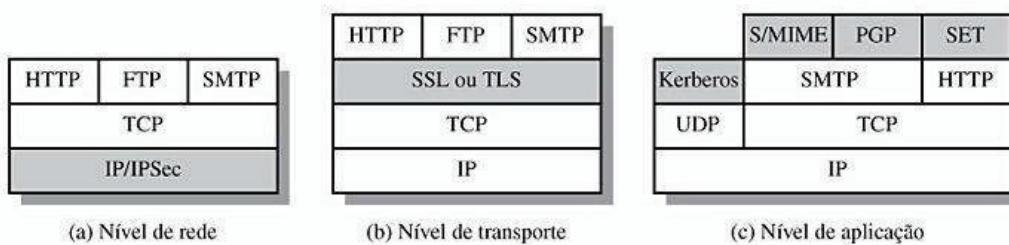


Figura 17.1 Local relativo das instalações de segurança na pilha de protocolos TCP/IP.

Outra solução de uso relativamente geral é implementar a segurança logo acima do TCP (Figura 17.1b). O exemplo principal dessa técnica é o Secure Sockets Layer (SSL) e seu padrão de Internet correspondente, conhecido como Transport Layer Security (TLS). Nesse nível, existem duas opções de implementação. Para a generalidade total, o SSL (ou TLS) pode ser fornecido como parte do conjunto de protocolos básico e, portanto, ser transparente às aplicações. Como alternativa, o SSL pode ser embutido em pacotes específicos. Por exemplo, os navegadores Netscape e Microsoft Explorer vêm equipados com SSL, e a maioria dos servidores Web tem implementado o protocolo.

Os serviços de segurança específicos de aplicação são embutidos na aplicação específica. A Figura 17.1c mostra exemplos dessa abordagem. A vantagem dessa abordagem é que o serviço pode ser ajustado às necessidades específicas de determinada aplicação. No contexto da segurança na Web, um exemplo importante dessa técnica é o Secure Electronic Transaction (SET).¹

O restante deste capítulo é dedicado a uma explicação sobre SSL/TLS e SET.

1. A Figura 17.1c mostra o SET em cima do HTTP; essa é uma implementação comum. Em algumas implementações, o SET utiliza o TCP diretamente.

17.2 SECURE SOCKET LAYER E TRANSPORT LAYER SECURITY

A Netscape criou o SSL. A versão 3 do protocolo foi projetada com revisão pública e do setor e foi publicada como um *draft* na Internet. Posteriormente, quando se chegou a um consenso para submeter o protocolo a padronização na Internet, um grupo de trabalho de TLS foi formado dentro do IETF para desenvolver um padrão comum. Essa primeira versão publicada do TLS pode ser vista como basicamente um SSLv3.1 e é muito próximo e compatível com o SSLv3.

A maior parte desta seção é dedicada a uma explicação do SSLv3. Ao final da seção, são descritas as principais diferenças entre SSLv3 e TLS.

Arquitetura SSL

O SSL é projetado para utilizar TCP para oferecer um serviço seguro confiável de ponta a ponta. O SSL não é um protocolo isolado, mas duas camadas de protocolo, conforme ilustra a Figura 17.2.



Figura 17.2 Pilha de protocolos SSL.

O Protocolo de Registro (Record Protocol) SSL oferece serviços básicos de segurança para vários protocolos de camada superior. Em particular, o Hypertext Transfer Protocol (HTTP), que oferece o serviço de transferência para interação cliente/servidor Web, pode operar em cima do SSL. Três protocolos de camada superior são definidos como parte do SSL: o Protocolo de Estabelecimento de Sessão (Handshake Protocol), o Protocolo de Mudança de Especificação de Cifra (Change Cipher Spec Protocol) e o Protocolo de Alerta (Alert Protocol). Esses protocolos específicos do SSL são usados no gerenciamento de trocas SSL e serão examinados mais adiante nesta seção.

Dois conceitos importantes do SSL são a sessão SSL e a conexão SSL, definidos na especificação da seguinte maneira:

- **Conexão:** Uma conexão é um transporte (na definição do modelo de camadas OSI) que oferece um tipo adequado de serviço. Para o SSL, essas conexões são relacionamentos peer-to-peer. As conexões são transientes. Cada conexão está associada a uma sessão.
- **Sessão:** Uma sessão SSL é uma associação entre um cliente e um servidor. As sessões são criadas pelo Protocolo de Estabelecimento de Sessão. As sessões definem um conjunto de parâmetros de segurança criptográficos que podem ser compartilhados entre várias conexões. As sessões são usadas para evitar a trabalhosa negociação de novos parâmetros de segurança para cada conexão.

Entre qualquer conjunto de duas partes (aplicações como HTTP no cliente e servidor), pode haver várias conexões seguras. Em teoria, também pode haver várias sessões simultâneas entre as partes, mas esse recurso não é usado na prática.

Na realidade, existem diversos estados associados a cada sessão. Quando uma sessão é estabelecida, existe um estado operacional para leitura e escrita (ou seja, recepção e envio). Além disso, durante o Protocolo de Estabelecimento de Sessão, são criados estados de leitura e escrita pendentes. Na conclusão bem-sucedida desse protocolo, os estados pendentes se tornam os estados atuais.

Um estado de sessão é definido pelos seguintes parâmetros (definições extraídas da especificação SSL):

- **Identificador de sessão (session identifier):** Uma seqüência arbitrária de bytes escolhida pelo servidor para identificar um estado de sessão ativo ou que pode ser retomado.
- **Certificado da parte (peer certificate):** Um certificado X509.v3 da parte. Esse elemento do estado pode ser nulo.

- **Método de compactação (compression method):** O algoritmo usado para compactar dados antes da criptografia.
- **Especificação de cifra (cipher spec):** Especifica o algoritmo de criptografia de dados em massa (sem criptografia, AES etc.) e um algoritmo de hash (como MD5 ou SHA-1) usado para o cálculo do MAC. Ela também define as propriedades criptográficas, como hash_size.
- **Segredo mestre (master secret):** Segredo de 48 bytes compartilhado entre o cliente e o servidor.
- **É retomável (is resumable):** Um flag indicando se a sessão pode ser usada para iniciar novas conexões.

Um estado de conexão é definido pelos parâmetros a seguir:

- **Aleatório de servidor e aleatório de cliente:** Seqüências de bytes escolhidas pelo servidor e cliente para cada conexão.
- **Segredo MAC de escrita do servidor:** A chave secreta usada nas operações MAC em dados enviados pelo servidor.
- **Segredo MAC de escrita do cliente:** A chave secreta usada em operações MAC em dados enviados pelo cliente.
- **Chave de escrita do servidor:** A chave de criptografia convencional para dados criptografados pelo servidor e descriptografados pelo cliente.
- **Chave de escrita do cliente:** A chave de criptografia convencional para dados criptografados pelo cliente e descriptografados pelo servidor.
- **Vetores de inicialização:** Quando uma cifra de bloco no modo CBC é usada, um vetor de inicialização (IV) é mantido para cada chave. Esse campo é inicializado primeiro pelo Protocolo de Estabelecimento de Sessão do SSL. Depois disso, o bloco de texto cifrado final de cada registro é preservado para uso como o IV com o registro seguinte.
- **Números de seqüência:** Cada parte mantém números de seqüência separados para mensagens transmitidas e recebidas para cada conexão. Quando uma parte envia ou recebe uma mensagem de mudança de especificação de cifra, o número de seqüência apropriado é definido como zero. Números de seqüência não podem exceder $2^{64} - 1$.

Protocolo de registro SSL

O Protocolo de Registro SSL oferece dois serviços para conexões SSL:

- **Confidencialidade:** O Protocolo de Estabelecimento de Sessão define uma chave secreta compartilhada usada para criptografia convencional dos dados a serem transportados pelo SSL.
- **Integridade da mensagem:** o Protocolo de Estabelecimento de Sessão também define uma chave secreta compartilhada usada para formar um código de autenticação de mensagem (MAC).

A Figura 17.3 indica a operação geral do protocolo de registro SSL. O protocolo de registro toma uma mensagem da aplicação a ser transmitida, fragmenta os dados em blocos gerenciáveis, comprime opcionalmente os dados, aplica um MAC, criptografa, acrescenta um cabeçalho e transmite a unidade resultante em um segmento TCP. Os dados recebidos são descriptografados, verificados, descompactados e remontados e, então, entregues a usuários de nível mais alto.

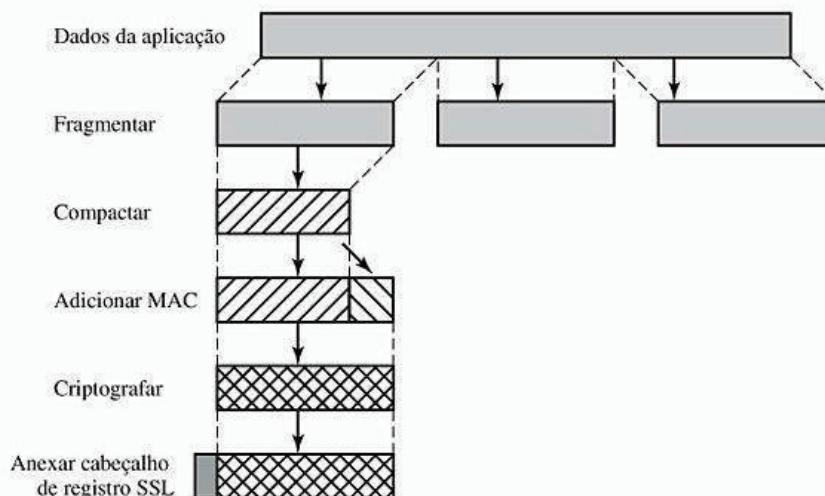


Figura 17.3 Operação do protocolo de registro SSL.

A primeira etapa é a **fragmentação**. Cada mensagem de camada superior é fragmentada em blocos de 2^{14} bytes (16384 bytes) ou menos. Em seguida, a **compactação** é aplicada opcionalmente. A compactação precisa ser sem perdas e não pode aumentar o tamanho do conteúdo em mais que 1.024 bytes.² No SSLv3 (além da versão atual do TLS), nenhum algoritmo de compactação é especificado, de modo que o algoritmo de compactação padrão é nulo (sem compactação).

A etapa seguinte no processamento é calcular um **código de autenticação de mensagens** sobre os dados compactados. Para essa finalidade, é usada uma chave secreta compartilhada. O cálculo é definido como

```
hash(MAC_write_secret || pad_2 ||
      hash(MAC_write_secret || pad_1 || seq_num ||
           SSLCompressed.type ||
           SSLCompressed.length || SSLCompressed.fragment))
```

onde

	= concatenação
MAC_write_secret	= chave secreta compartilhada
hash	= algoritmo de hash criptográfico; ou MD5 ou SHA-1
pad_1	= o byte 0x36 (0011 0110) repetido 48 vezes (384 bits) para MD5 e 40 vezes (320 bits) para SHA-1
pad_2	= o byte 0x5C (0101 1100) repetido 48 vezes para MD5 e 40 vezes para SHA-1
seq_num	= o número de sequência para essa mensagem
SSLCompressed.type	= o protocolo de nível mais alto usado para processar esse fragmento
SSLCompressed.length	= o tamanho do fragmento compactado
SSLCompressed.fragment	= o fragmento compactado (se a compactação não for usada, o fragmento de texto claro)

Observe que isso é muito semelhante ao algoritmo HMAC definido no Capítulo 12. A diferença é que os dois complementos são concatenados no SSLv3 e passam por um XOR no HMAC. O algoritmo MAC do SSLv3 é baseado no *draft* de Internet original para HMAC, que usava concatenação. A versão final do HMAC, definida na RFC 2104, utiliza o XOR.

Em seguida, a mensagem compactada mais o MAC são criptografados usando a criptografia simétrica. A criptografia não pode aumentar o tamanho do conteúdo em mais que 1.024 bytes, de modo que o tamanho total não pode exceder $2^{14} + 2.048$. Os algoritmos de criptografia a seguir são permitidos:

Cifra de bloco		Cifra de fluxo	
Algoritmo	Tamanho da chave	Algoritmo	Tamanho da chave
AES	128, 256	RC4-40	40
IDEA	128	RC4-128	128
RC2-40	40		
DES-40	40		
DES	56		
3DES	168		
Forteza	80		

Forteza pode ser usado em um esquema de criptografia de smartcard.

Para a criptografia de fluxo, a mensagem compactada mais o MAC são criptografados. Observe que o MAC é calculado antes que a criptografia ocorra e que o MAC é então criptografado juntamente com o texto claro ou texto claro compactado.

Para a criptografia de bloco, a complementação pode ser acrescentada após o MAC antes da criptografia. A complementação está na forma de uma quantidade de bytes de complementação seguidos por uma indicação do comprimento da complementação de um byte. A quantidade total de complementação é a menor quantidade de modo que o tamanho total dos dados a serem criptografados (texto claro mais MAC mais preenchimento) seja um múltiplo do tamanho de bloco da cifra. Um exemplo é um texto claro (ou texto comprimido, se a compressão for usada) de 58

2. Naturalmente, espera-se que a compactação reduza ao invés de expandir os dados. Porém, para blocos muito curtos, é possível, devido a convenções de formatação, que o algoritmo de compactação resulte em uma saída que seja maior que a entrada.

bytes, com um MAC de 20 bytes (usando SHA-1), que é criptografado usando um tamanho de bloco de 8 bytes (por exemplo, DES). Com o byte padding.length, isso gera um total de 79 bytes. Para tornar o total um múltiplo inteiro de 8, um byte de preenchimento é acrescentado.

A última etapa do processamento do Protocolo de Registro SSL é anexar um cabeçalho no início, consistindo nos seguintes campos:

- **Tipo de conteúdo (content type)(8 bits):** O protocolo da camada mais alta, usado para processar o fragmento transportado.
- **Versão principal (major version)(8 bits):** Indica a versão principal do SSL em uso. Para SSLv3, o valor é 3.
- **Versão secundária (minor version)(8 bits):** Indica a versão secundária em uso. Para SSLv3, o valor é 0.
- **Tamanho compactado (compressed length) (16 bits):** O tamanho em bytes do fragmento de texto claro (ou fragmento compactado, se a compactação for usada). O valor máximo é $2^{14} + 2.048$.

Os tipos de conteúdo definidos são change_cipher_spec, alert, handshake e application_data. Os três primeiros são os protocolos específicos do SSL, explicados a seguir. Observe que nenhuma distinção é feita entre as diversas aplicações (por exemplo, HTTP) que poderiam usar SSL; o conteúdo dos dados criados por tais aplicações não é visível para o SSL.

A Figura 17.4 ilustra o formato do registro SSL.

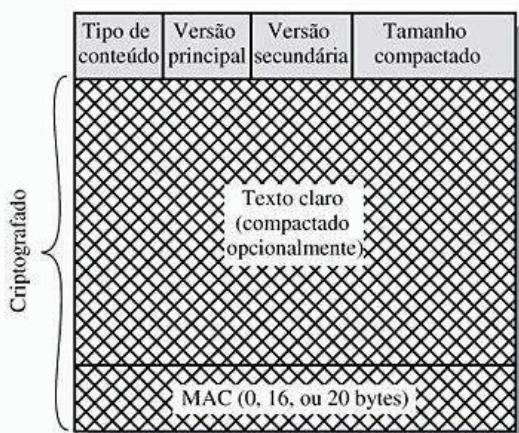


Figura 17.4 Formato do registro SSL.

Protocolo de mudança de especificação de cifra

O Protocolo de Mudança de Especificação de Cifra é um dos três protocolos específicos do SSL que utilizam o Protocolo de Registro SSL, e é o mais simples. Esse protocolo é composto por uma única mensagem (Figura 17.5a), que consiste em um único byte com o valor 1. A única finalidade dessa mensagem é fazer com que o estado pendente seja copiado para o estado atual, o que atualiza o conjunto de cifras a ser usado nessa conexão.

1 byte

1

(a) Protocolo de mudança de especificação de cifra

1 byte 3 bytes

Tipo Tamanho

≥ 0 bytes

Conteúdo

(c) Protocolo de estabelecimento da conexão

1 byte 1 byte

Nível	Alerta
-------	--------

(b) Protocolo de alerta

≥ 1 byte

Conteúdo opaco

(d) Outro protocolo de camada superior (por exemplo, HTTP)

Figura 17.5 Payload do Protocolo de Registro SSL.

Protocolo de alerta

O Protocolo de Alerta é usado para transmitir alertas relacionados ao SSL para as partes envolvidas. Assim como outras aplicações que usam SSL, as mensagens de alerta são compactadas e criptografadas, conforme especificado pelo estado atual.

Cada mensagem nesse protocolo consiste em dois bytes (Figura 17.5b). O primeiro byte tem o valor warning(1) ou fatal(2) para transportar o grau de gravidade da mensagem. Se o nível for fatal, o SSL encerrará imediatamente a conexão. Outras conexões na mesma sessão podem continuar, mas nenhuma conexão nova nessa sessão pode ser estabelecida. O segundo byte contém um código que indica o alerta específico. Primeiramente, listamos os alertas que são sempre fatais (definições da especificação SSL).

- **unexpected_message:** uma mensagem não apropriada foi recebida.
- **bad_record_mac:** um MAC incorreto foi recebido.
- **decompression_failure:** a função de descompactação recebeu uma entrada imprópria (por exemplo, incapaz de descompactar ou descompactar para mais do que o tamanho máximo permitido).
- **handshake_failure:** o emissor foi incapaz de negociar um conjunto aceitável de parâmetros de segurança dadas as opções disponíveis.
- **illegal_parameter:** um campo em uma mensagem de estabelecimento de sessão estava fora de intervalo ou inconsistente com outros campos.

O restante dos alertas são os seguintes:

- **close_notify:** notifica o destinatário de que o emissor não enviará mais mensagens nessa conexão. Cada parte precisa enviar um alerta close_notify antes de fechar o lado de escrita de uma conexão.
- **no_certificate:** pode ser enviado em resposta a uma solicitação de certificado se nenhum certificado apropriado estiver disponível.
- **bad_certificate:** um certificado recebido foi adulterado (por exemplo, continha uma assinatura que não foi validada corretamente).
- **unsupported_certificate:** o tipo do certificado recebido não é admitido.
- **certificate_revoked:** um certificado foi revogado por seu assinante.
- **certificate_expired:** um certificado expirou.
- **certificate_unknown:** algum outro problema não especificado surgiu no processamento do certificado, tornando-o inaceitável.

Protocolo de estabelecimento de sessão

A parte mais complexa do SSL é Protocolo de Estabelecimento de Sessão. Esse protocolo permite que o servidor e o cliente autentiquem um ao outro e negociem um algoritmo de criptografia e de MAC e chaves criptográficas a serem usadas para proteger dados enviados em um registro SSL. O Protocolo de Estabelecimento de Sessão é usado antes que quaisquer dados de aplicação sejam transmitidos.

Esse protocolo é composto por uma série de mensagens trocadas entre cliente e servidor. Todas elas têm o formato mostrado na Figura 17.5c. Cada mensagem tem três campos:

- **Tipo (1 byte):** Indica uma de 10 mensagens. A Tabela 17.2 lista os tipos de mensagem definidos.
- **Tamanho (3 bytes):** O tamanho da mensagem em bytes.
- **Conteúdo (≥ 0 bytes):** Os parâmetros associados a essa mensagem; estes estão listados na Tabela 17.2.

Tabela 17.2 Tipos de mensagem do Protocolo de Estabelecimento de Sessão SSL

Tipo de mensagem	Parâmetros
hello_request	Nulo
client_hello	versão, aleatório, ID de sessão, conjunto de cifras, método de compactação
server_hello	versão, aleatório, ID de sessão, conjunto de cifras, método de compactação
certificate	cadeia de certificados X.509v3
server_key_exchange	parâmetros, assinatura
certificate_request	tipo, autoridades
server_done	Nulo
certificate_verify	Assinatura
client_key_exchange	parâmetros, assinatura
finished	valor de hash

A Figura 17.6 mostra a troca inicial necessária para estabelecer uma conexão lógica entre cliente e servidor. A troca pode ser vista como tendo quatro fases.

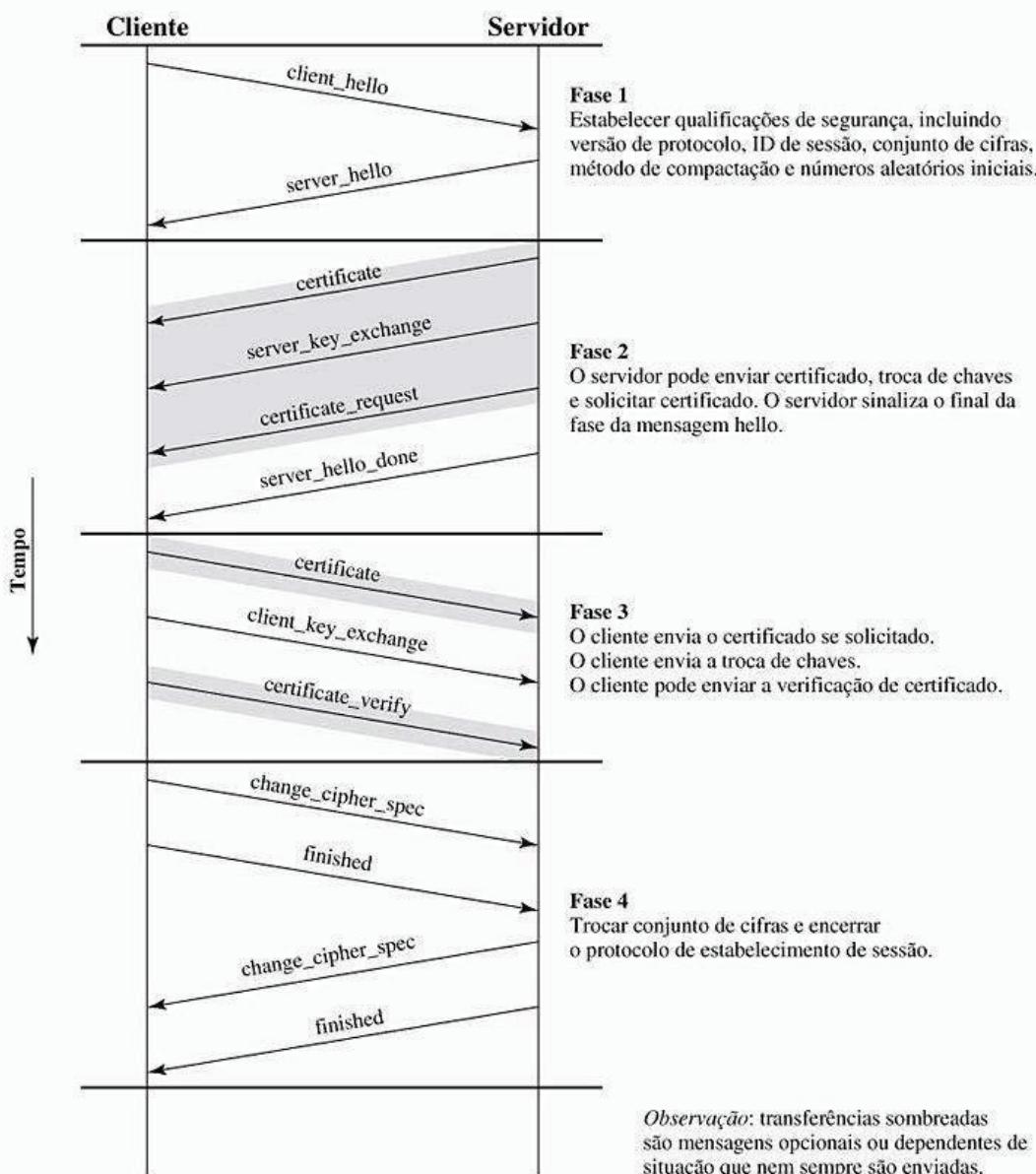


Figura 17.6 Ação do Protocolo de Estabelecimento de Sessão.

Fase 1. Estabelecer capacidades de segurança Esta fase é usada para iniciar uma conexão lógica e estabelecer as capacidades de segurança que serão associadas a ela. A troca é iniciada pelo cliente, que envia uma mensagem `client_hello` com os seguintes parâmetros:

- **Versão (version):** A versão SSL mais alta entendida pelo cliente.
- **Aleatório (random):** Uma estrutura aleatória gerada pelo cliente, consistindo em um carimbo de tempo de 32 bits e 28 bytes gerados por um gerador de números aleatórios seguro. Esses valores servem como nonces e são usados durante a troca de chaves para impedir ataques por repetição.
- **ID de sessão (session ID):** Um identificador de sessão de tamanho variável. Um valor diferente de zero indica que o cliente deseja atualizar os parâmetros de uma conexão existente ou criar uma nova conexão nesta sessão. Um valor zero indica que o cliente deseja estabelecer uma nova conexão em uma nova sessão.
- **Conjunto de cifras (CipherSuite):** Uma lista que contém as combinações de algoritmos criptográficos admitidos pelo cliente, em ordem decrescente de preferência. Cada elemento da lista (cada conjunto de cifras) define um algoritmo de troca de chaves e uma especificação de cifra; estas serão explicadas posteriormente.
- **Método de compactação (compression method):** Uma lista dos métodos de compactação que o cliente admite.

Depois de enviar a mensagem `client_hello`, o cliente espera pela mensagem `server_hello`, que contém os mesmos parâmetros da mensagem `client_hello`. Para a mensagem `server_hello`, as convenções a seguir se aplicam. O campo Versão contém a menor versão sugerida pelo cliente e a maior admitida pelo servidor. O campo Aleatório é gerado

pelo servidor e é independente do campo Aleatório do cliente. Se o campo ID de sessão do cliente for diferente de zero, o mesmo valor será usado pelo servidor; caso contrário, o campo ID de Sessão do servidor conterá o valor de uma nova sessão. O campo Conjunto de Cifras contém o único conjunto de cifras selecionado pelo servidor a partir daqueles propostos pelo cliente. O campo Método de Compactação contém o método de compactação selecionado pelo servidor a partir daqueles propostos pelo cliente.

O primeiro elemento do parâmetro Conjunto de Cifras é o método da troca de chaves (ou seja, o meio pelo qual as chaves criptográficas para a criptografia convencional e MAC são trocadas). Os seguintes métodos de troca de chaves são admitidos:

- **RSA:** A chave secreta é criptografada com a chave pública RSA do receptor. Um certificado de chave pública para a chave do receptor precisa ser disponibilizado.
- **Diffie-Hellman fixo:** Essa é uma troca de chaves Diffie-Hellman em que o certificado do servidor contém os parâmetros públicos Diffie-Hellman assinados pela autoridade certificadora (CA). Isto é, o certificado de chave pública contém os parâmetros de chave pública Diffie-Hellman. O cliente oferece esses parâmetros de chave pública Diffie-Hellman ou em um certificado, se a autenticação do cliente for exigida, ou em uma mensagem de troca de chaves. Esse método resulta em uma chave secreta fixa entre duas partes, com base no cálculo Diffie-Hellman usando as chaves públicas fixas.
- **Diffie-Hellman efêmero:** Essa técnica é usada para criar chaves secretas efêmeras (temporárias, de uso único). Nesse caso, as chaves públicas Diffie-Hellman são trocadas, assinadas usando a chave RSA ou DSS privada do emissor. O receptor pode usar a chave pública correspondente para verificar a assinatura. Os certificados são usados para autenticar as chaves públicas. Esta parece ser a mais segura das três opções Diffie-Hellman, pois resulta em uma chave temporária e autenticada.
- **Diffie-Hellman anônimo:** O algoritmo Diffie-Hellman básico é usado, sem autenticação. Ou seja, cada lado envia seus parâmetros Diffie-Hellman públicos para o outro, sem autenticação. Essa técnica é vulnerável a ataques de homem no meio (man-in-the-middle), em que o atacante realiza Diffie-Hellman anônimo com ambas as partes.
- **Fortezza:** A técnica definida para o esquema Fortezza.

Após a definição de um método de troca de chaves, existe a Especificação de Cifra (CipherSpec), que inclui os seguintes campos:

- **Algoritmo de cifra (CipherAlgorithm):** qualquer um dos algoritmos mencionados anteriormente: RC4, RC2, DES, 3DES, DES40, IDEA, Fortezza
- **Algoritmo MAC (MACAlgorithm):** MD5 ou SHA-1
- **Tipo de cifra (CipherType):** fluxo ou bloco
- **É exportável (IsExportable):** Verdadeiro ou falso
- **Tamanho de hash (HashSize):** 0, 16 (para MD5), ou 20 (para SHA-1) bytes
- **Material da chave (Key Material):** uma seqüência de bytes que contém dados usados na geração das chaves de escrita
- **Tamanho do IV (IV size):** o tamanho do vetor de inicialização para a criptografia CBC (Cipher Block Chaining)

Fase 2. Autenticação de servidor e troca de chaves O servidor inicia essa fase enviando seus certificados, caso precise ser autenticado; a mensagem contém um ou uma cadeia de certificados X.509. A **mensagem de certificado (certificate message)** é exigida para qualquer método e troca de chaves que tenham sido acordados, exceto Diffie-Hellman anônimo. Observe que, se o Diffie-Hellman fixo for usado, essa mensagem de certificado funcionará como a mensagem de troca de chaves do servidor, pois contém os parâmetros Diffie-Hellman públicos do servidor.

Em seguida, uma mensagem **server_key_exchange** pode ser enviada, se for necessário. Ela não é exigida em dois casos: (1) O servidor enviou um certificado com parâmetros Diffie-Hellman fixos ou (2) a troca de chaves RSA será usada. A mensagem **server_key_exchange** é necessária para o seguinte:

- **Diffie-Hellman anônimo:** o conteúdo da mensagem consiste nos dois valores Diffie-Hellman globais (um número primo e uma raiz primitiva desse número) mais a chave Diffie-Hellman pública do servidor (ver Figura 10.7).
- **Diffie-Hellman efêmero:** o conteúdo da mensagem inclui os três parâmetros Diffie-Hellman fornecidos para Diffie-Hellman anônimo, mais uma assinatura desses parâmetros.
- **Troca de chaves RSA, em que o servidor está usando RSA, mas tem uma chave RSA apenas de assinatura:** por conseguinte, o cliente não pode simplesmente enviar uma chave secreta criptografada com a chave pública do servidor. Em vez disso, o servidor precisa criar um par temporário de chaves pública/privada RSA e usar a mensagem **server_key_exchange** para enviar a chave pública. O conteúdo da mensagem inclui os dois parâmetros da chave pública RSA temporária (expoente e módulo, ver Figura 9.5) mais uma assinatura desses parâmetros.
- **Fortezza**

Mais alguns detalhes sobre as assinaturas são garantidos. Como sempre, uma assinatura é criada tomando-se o hash de uma mensagem e criptografando-o com a chave privada do emissor. Nesse caso, o hash é definido como:

```
hash(ClientHello.random || ServerHello.random || ServerParams)
```

Assim, o hash abrange não apenas os parâmetros Diffie-Hellman ou RSA, mas também os dois nonces das mensagens hello iniciais. Esse procedimento fornece garantias contra ataques por repetição e falsificação de identidade. No caso de uma assinatura DSS, o hash é realizado usando-se o algoritmo SHA-1. No caso de uma assinatura RSA, tanto MD5 quanto um hash SHA-1 são calculados, e a concatenação dos dois hashes (36 bytes) é criptografada com a chave privada do servidor.

Em seguida, um servidor não anônimo (servidor não usando Diffie-Hellman anônimo) pode solicitar um certificado do cliente. A mensagem **certificate_request** inclui dois parâmetros: **certificate_type** e **certificateAuthorities**. A mensagem **certificate_type** indica o algoritmo de chave pública e seu uso:

- RSA, somente assinatura
- DSS, somente assinatura
- RSA para Diffie-Hellman fixo; nesse caso, a assinatura só é usada para autenticação, enviando um certificado assinado com RSA
- DSS para Diffie-Hellman fixo; novamente, usado apenas para autenticação
- RSA para Diffie-Hellman efêmero
- DSS para Diffie-Hellman efêmero
- Fortezza

O segundo parâmetro na mensagem **certificate_request** é uma lista dos nomes distintos de autoridades certificadoras aceitáveis.

A mensagem final na Fase 2, e que é sempre exigida, é a **mensagem server_done**, que é enviada pelo servidor para indicar o final do hello do servidor e mensagens associadas. Depois de enviar essa mensagem, o servidor esperará pela resposta de um cliente. Essa mensagem não possui parâmetros.

Fase 3. Autenticação do cliente e troca de chaves Ao receber a mensagem **server_done**, o cliente deve verificar se o servidor forneceu um certificado válido, caso exigido, e verificar se os parâmetros **server_hello** são aceitáveis. Se tudo estiver satisfatório, o cliente envia uma ou mais mensagens de volta ao servidor.

Se o servidor tiver solicitado um certificado, o cliente iniciará essa fase enviando uma **mensagem certificate**. Se nenhum certificado apropriado estiver disponível, o cliente enviará um alerta **no_certificate**.

Em seguida vem a **mensagem client_key_exchange**, que precisa ser enviada nessa fase. O conteúdo da mensagem depende do tipo de troca de chaves, da seguinte forma:

- **RSA:** O cliente gera um *pré-segredo-mestre* (pre-master secret) de 48 bytes e o criptografa com a chave pública do certificado do servidor ou chave RSA temporária de uma mensagem **server_key_exchange**. Seu uso para calcular um *segredo-mestre* é explicado mais adiante.
- **Diffie-Hellman efêmero ou anônimo:** Os parâmetros Diffie-Hellman públicos do cliente são enviados.
- **Diffie-Hellman fixo:** Os parâmetros Diffie-Hellman públicos do cliente foram enviados em uma mensagem **certificate**, de modo que o conteúdo dessa mensagem é nulo.
- **Fortezza:** Os parâmetros Fortezza do cliente são enviados.

Finalmente, nessa fase, o cliente pode enviar uma mensagem **certificate_verify** para oferecer verificação explícita de um certificado do cliente. Essa mensagem só é enviada após qualquer certificado do cliente que tenha capacidade de assinatura (isto é, todos os certificados exceto aqueles contendo parâmetros Diffie-Hellman fixos). Essa mensagem assina um código de hash com base nas mensagens anteriores, definido da seguinte forma:

```
CertificateVerify.signature.md5_hash
MD5(master_secret || pad_2 || MD5(handshake_messages ||
master_secret || pad_1));
Certificate.signature.sha_hash
SHA(master_secret || pad_2 || SHA(handshake_messages ||
master_secret || pad_1));
```

onde **pad_1** e **pad_2** são os valores definidos anteriormente para o MAC, **handshake_messages** refere-se a todas as mensagens Protocolo de Estabelecimento de Sessão enviadas ou recebidas a partir de **client_hello** mas não incluindo essa mensagem, e **master_secret** é o segredo calculado cuja construção é explicada mais adiante nesta seção. Se a chave privada do usuário for DSS, então ela é usada para criptografar o hash SHA-1. Se a chave privada do usuário for RSA, ela é usada para criptografar a concatenação dos hashes MD5 e SHA-1. De qualquer forma, a finalidade é

verificar se o cliente possui a chave privada para o certificado do cliente. Mesmo que alguém estivesse usando o certificado do cliente sem autorização, ele não conseguiria enviar essa mensagem.

Fase 4. Término Esta fase completa a configuração de uma conexão segura. O cliente envia uma **mensagem change_cipher_spec** e copia o CipherSpec pendente para a CipherSpec atual. Observe que essa mensagem não é considerada parte do Protocolo de Estabelecimento de Sessão, mas sim enviada usando o Protocolo de Mudança de Especificação de Cifra. O cliente, então, envia imediatamente uma **finished message (mensagem de conclusão)** sob os novos algoritmos, chaves e segredos. A mensagem de conclusão verifica se os processos de troca de chaves e autenticação foram bem-sucedidos. O conteúdo da mensagem de conclusão é a concatenação de dois valores de hash:

```
MD5(master_secret || pad2 || MD5(handshake_messages ||
    Sender || master_secret || pad1))
SHA(master_secret || pad2 || SHA(handshake_messages ||
    Sender || master_secret || pad1))
```

onde Sender é um código que identifica que o emissor é o cliente e handshake_messages são todos os dados de todas as mensagens de estabelecimento de sessão até o momento, não incluindo esta mensagem.

Em resposta a essas duas mensagens, o servidor envia sua própria mensagem **change_cipher_spec**, transfere o CipherSpec pendente para a atual, e envia sua finished message. Nesse ponto, o estabelecimento da conexão está completo e cliente e servidor podem começar a trocar dados da camada de aplicação.

Cálculos criptográficos

Dois outros itens são interessantes: a criação de uma chave-mestra compartilhada por meio da troca de chaves e a geração de parâmetros criptográficos do segredo-mestre.

Criação de segredo-mestre O segredo-mestre (master secret) compartilhado é um valor de 48 bytes (384 bits) de uso único gerado para esta sessão por meio da troca de chaves segura. A criação é feita em dois estágios. Primeiro, um **pre_master_secret** é trocado. Segundo, o **master_secret** é calculado pelas duas partes. Para a troca de **pre_master_secret**, existem duas possibilidades:

- **RSA:** Um **pre_master_secret** de 48 bytes é gerado pelo cliente, criptografado com a chave RSA pública do servidor, e enviado ao servidor. O servidor decriptografa o texto cifrado usando sua chave privada para recuperar o **pre_master_secret**.
- **Diffie-Hellman:** Cliente e servidor geram uma chave pública Diffie-Hellman. Depois que estas forem trocadas, cada lado realizará o cálculo Diffie-Hellman para criar o **pre_master_secret** compartilhado.

Os dois lados agora calculam o **master_secret** da seguinte forma:

```
master_secret = MD5(pre_master_secret || SHA('A' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random)) ||
    MD5(pre_master_secret || SHA('BB' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random)) ||
    MD5(pre_master_secret || SHA('CCC' ||
    pre_master_secret || ClientHello.random ||
    ServerHello.random))
```

onde ClientHello.random e ServerHello.random são os dois valores de nonce trocados nas mensagens hello iniciais.

Geração de parâmetros criptográficos O CipherSpecs requer um segredo MAC de escrita do cliente, um segredo MAC de escrita do servidor, uma chave de escrita do cliente, uma chave de escrita do servidor, um IV de escrita do cliente e um IV de escrita do servidor, que são gerados a partir do segredo-mestre, nesta ordem. Esses parâmetros são gerados a partir do segredo-mestre, fazendo o hashing desse valor e gerando uma seqüência de bytes de tamanho suficiente para todos os parâmetros necessários.

A geração dos parâmetros relativos a chaves e segredos a partir do segredo-mestre usa o mesmo formato utilizado na geração do segredo-mestre a partir do pré-segredo-mestre:

```
key_block = MD5(master_secret || SHA('A' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('BB' || master_secret ||
    ServerHello.random || ClientHello.random)) ||
    MD5(master_secret || SHA('CCC' || master_secret ||
    ServerHello.random || ClientHello.random))
```

```
MD5(master_secret || SHA('CCC' || master_secret ||
ServerHello.random || ClientHello.random)) || ...
```

até que uma saída suficiente tenha sido gerada. O resultado dessa estrutura algorítmica é uma função pseudo-aleatória. Podemos pensar no master_secret como o valor de semente pseudo-aleatória para a função. Os números aleatórios de cliente e servidor podem ser vistos como salt values (valores de sal) para complicar a criptoanálise (veja no Capítulo 18 uma explicação sobre o uso dos salt values).

Transport Layer Security

O TLS é uma iniciativa de padronização do IETF cujo objetivo é produzir um padrão de Internet que seja uma versão do SSL. O TLS é definido como um Proposed Internet Standard (padrão proposto para a Internet) na RFC 2246, a qual é muito semelhante à SSLv3. Nesta seção, destacamos as diferenças.

Número de versão O formato do registro do TLS é o mesmo formato do registro do SSL (Figura 17.4), e os campos no cabeçalho têm os mesmos significados. A única diferença está nos valores de versão. Para a versão atual do TLS, a versão principal é 3 e a versão secundária é 1.

Message Authentication Code Existem duas diferenças entre os esquemas de MAC do SSLv3 e do TLS: o algoritmo real e o escopo do cálculo do MAC. O TLS utiliza o algoritmo HMAC definido na RFC 2104. Lembre-se, do Capítulo 12, de que o HMAC é definido da seguinte forma:

$$\text{HMAC}(M) = \text{H}[(K^+ \oplus \text{opad}) \parallel \text{H}[K^+ \oplus \text{ipad}] \parallel M]$$

onde

H	= função de hash embutida (para TLS, MD5 ou SHA-1)
M	= entrada de mensagem para HMAC
K ⁺	= chave secreta complementada com zeros à esquerda, de modo que o resultado é igual ao tamanho do bloco do código de hash (para MD5 e SHA-1, tamanho de bloco = 512 bits)
ipad	= 00110110 (36 em hexadecimal) repetido 64 vezes (512 bits)
opad	= 01011100 (5C em hexadecimal) repetido 64 vezes (512 bits)

O SSLv3 usa o mesmo algoritmo, exceto pelo fato de que os bytes de complementação são concatenados com a chave secreta em vez de realizar o XOR desses valores com a chave secreta complementada até o tamanho do bloco. O nível de segurança deverá ser praticamente o mesmo nos dois casos.

Para o TLS, o cálculo do MAC comprehende os campos indicados na expressão a seguir:

```
HMAC_hash(MAC_write_secret, seq_num || TLSCompressed.type ||
TLSCompressed.version || TLSCompressed.length ||
TLSCompressed.fragment)
```

O cálculo do MAC abrange todos os campos cobertos pelo cálculo do SSLv3, mais o campo TLSCompressed.version, que é a versão do protocolo que está sendo empregada.

Função pseudo-aleatória O TLS utiliza uma função pseudo-aleatória referenciada como PRF (Pseudo Random Function) para expandir segredos em blocos de dados para fins de geração ou validação de chave. O objetivo é utilizar um valor de segredo compartilhado relativamente pequeno, mas para gerar blocos de dados maiores de um modo que seja seguro contra os tipos de ataques feitos a funções de hash e MACs. O PRF é baseado na seguinte função de expansão de dados (Figura 17.7):

```
P_hash(secret, seed) = HMAC_hash(secret, A(1) || seed) ||
HMAC_hash(secret, A(2) || seed) ||
HMAC_hash(secret, A(3) || seed) || ...
```

onde A() é definido como

$$\begin{aligned} A(0) &= \text{semente} \\ A(i) &= \text{HMAC_hash(segredo, } A(i-1)) \end{aligned}$$

A função de expansão de dados utiliza o algoritmo HMAC, seja com MD5 ou SHA-1 como função de hash básica. Como podemos ver, P_hash pode ser repetido tantas vezes quantas forem necessárias para produzir a quantidade de dados exigida. Por exemplo, se P_SHA-1 fosse usado para gerar 64 bytes de dados, ele teria de ser repetido quatro vezes, produzindo 80 bytes de dados, dos quais os últimos 16 seriam descartados. Nesse caso, P_MD5 também teria de ser repetido quatro vezes, produzindo exatamente 64 bytes de dados. Observe que cada repetição envolve duas execuções do HMAC, cada uma por sua vez envolvendo duas execuções do algoritmo de hash embutido.

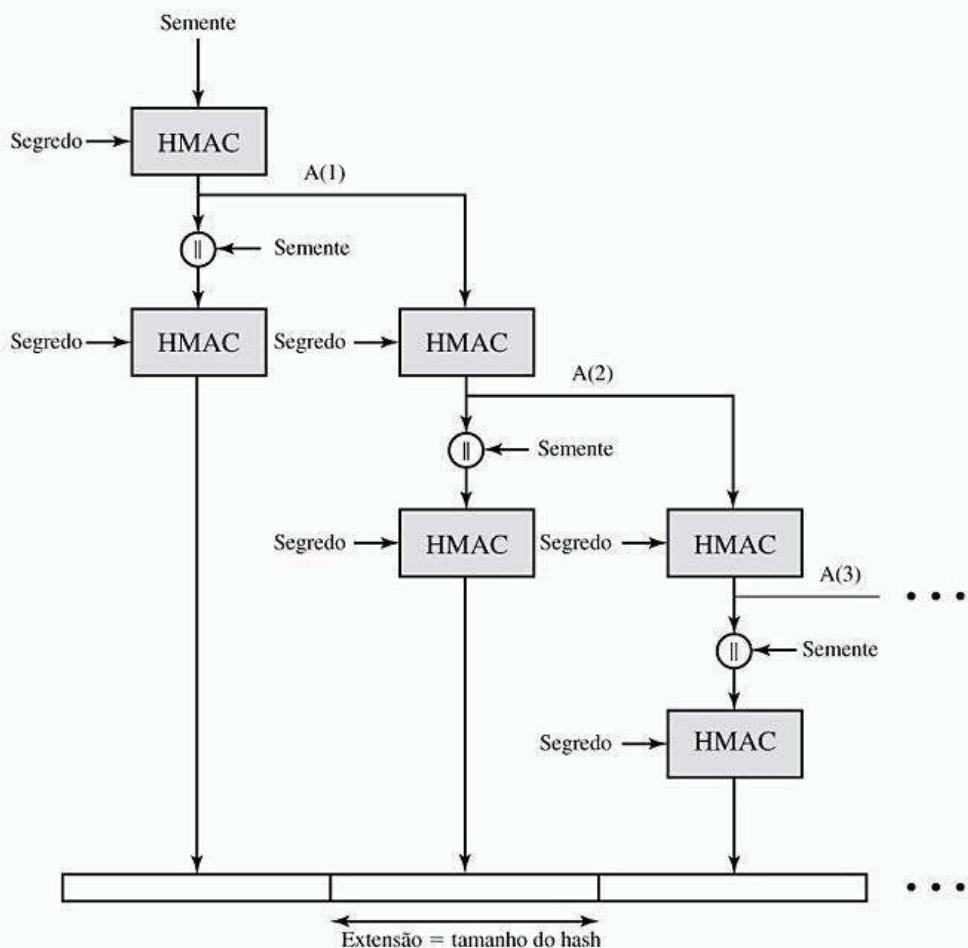


Figura 17.7 Função P_hash(segredo,semente) do TLS.

Para tornar a PRF tão segura quanto possível são utilizados dois algoritmos de hash, de modo a garantir a segurança da PRF se, pelo menos, um dos algoritmos permanecer seguro. A PRF é definida como

$$\text{PRF}(\text{secret}, \text{label}, \text{seed}) = \text{P_MD5}(S1, \text{label} \parallel \text{seed}) \oplus \text{P_SHA-1}(S2, \text{label} \parallel \text{seed})$$

Essa função tem como entrada um valor secreto, um rótulo de identificação e um valor de semente. Ela produz uma saída de tamanho arbitrário. A saída é criada dividindo-se o valor secreto em duas metades (S1 e S2) e realizando-se o P_hash em cada metade, usando MD5 em uma metade e SHA-1 na outra. Os dois resultados passam por um XOR para produzir a saída; para essa finalidade, P_MD5 geralmente terá de ser repetido mais vezes do que P_SHA-1 para produzir uma quantidade igual de dados para a entrada na função XOR.

Códigos de alerta O TLS admite todos os códigos de alerta definidos na SSLv3, com exceção de no_certificate. Diversos códigos adicionais são definidos no TLS; destes, os seguintes são sempre fatais:

- **decryption_failed:** um texto cifrado de uma maneira inválida; ou ele não era um múltiplo par do tamanho do bloco ou seus valores de complementação, quando verificados, estavam incorretos.
- **record_overflow:** um registro do TLS foi recebido com um payload (texto cifrado) cujo tamanho ultrapassa $2^{14} + 2.048$ bytes, ou o texto cifrado foi decriptografado para um tamanho maior que $2^{14} + 1.024$ bytes.
- **unknown_ca:** uma cadeia de certificados válida ou cadeia parcial foi recebida, mas o certificado não foi aceito porque o certificado da CA não pôde ser localizado ou não correspondeu a nenhuma CA conhecida e confiável.
- **access_denied:** um certificado válido foi recebido, mas quando a conectividade de acesso foi aplicada, o emissor decidiu não prosseguir com a negociação.
- **decode_error:** uma mensagem não pôde ser decodificada, pois um campo estava fora do seu intervalo especificado ou o tamanho da mensagem foi incorreto.

- **export_restriction:** foi detectada uma negociação que não estava em conformidade com as restrições de exportação sobre o tamanho da chave.
- **protocol_version:** a versão do protocolo que o cliente tentou negociar é reconhecida, mas não é aceita.
- **insufficient_security:** retornado no lugar de handshake_failure quando uma negociação falhou especificamente porque o servidor exige cifras mais seguras do que aquelas admitidas pelo cliente.
- **internal_error:** um erro interno não relacionado à outra parte ou à exatidão do protocolo torna impossível continuar.

O restante dos novos alertas inclui o seguinte:

- **decrypt_error:** uma operação criptográfica de estabelecimento de sessão falhou, o que inclui ser incapaz de verificar uma assinatura, descriptografar uma troca de chaves ou validar uma mensagem acabada.
- **user_canceled:** este estabelecimento de conexão está sendo cancelado por algum motivo não relacionado a uma falha de protocolo.
- **no_renegotiation:** enviado por um cliente em resposta a uma solicitação hello ou pelo servidor em resposta a um hello do cliente após o estabelecimento inicial da sessão. Uma mensagem dessas normalmente resultaria em renegociação, mas esse alerta indica que o emissor não é capaz de renegociar. Essa mensagem é sempre uma advertência.

Conjuntos de cifras Existem várias diferenças pequenas entre os conjuntos de cifras disponíveis no SSLv3 e no TLS:

- **Troca de chaves:** O TLS admite todas as técnicas de troca de chaves do SSLv3, com exceção do Fortezza.
- **Algoritmos de criptografia simétrica:** o TLS inclui todos os algoritmos de criptografia simétrica encontrados no SSLv3, com exceção do Fortezza.

Tipos de certificado do cliente O TLS define os seguintes tipos de certificado a serem solicitados em uma mensagem certificate_request: rsa_sign, dss_sign, rsa_fixed_dh e dss_fixed_dh. Todos esses são definidos no SSLv3. Além disso, o SSLv3 inclui rsa_ephemeral_dh, dss_ephemeral_dh e fortezza_kea. O Diffie-Hellman efêmero envolve a assinatura dos parâmetros Diffie-Hellman com RSA ou DSS; para o TLS, os tipos rsa_sign e dss_sign são usados para essa função; um tipo de assinatura separado não é necessário para assinar parâmetros Diffie-Hellman. O TLS não inclui o esquema Fortezza.

Mensagens certificate_verify e finished Na mensagem certificate_verify, os hashes MD5 e SHA-1 são calculados apenas sobre handshake_messages. Lembre-se de que, para o SSLv3, o cálculo de hash também inclui o segredo-mestre e complementações. Esses campos extras não representam segurança adicional.

Assim como a finished message (mensagem de conclusão) no SSLv3, a finished message no TLS é um hash com base no master_secret compartilhado, as mensagens de estabelecimento de sessão (handshake_messages) anteriores, e um rótulo que identifica o cliente ou servidor. O cálculo é um pouco diferente. Para o TLS, temos

```
PRF(master_secret, finished_label, MD5(handshake_messages) ||
     SHA-1(handshake_messages))
```

onde finished_label é a string “client finished” para o cliente e “server finished” para o servidor.

Cálculos criptográficos O pre_master_secret para o TLS é calculado da mesma maneira que no SSLv3. Assim como no SSLv3, o master_secret no TLS é calculado como uma função de hash do pre_master_secret e os dois números aleatórios de hello. A forma do cálculo do TLS é diferente da usada no SSLv3, e é definida da seguinte forma:

```
master_secret = PRF(pre_master_secret, "master secret",
                      ClientHello.random || ServerHello.random)
```

O algoritmo é executado até que 48 bytes de saída pseudo-aleatória sejam produzidos. O cálculo dos parâmetros relativos a chaves e segredos (chaves secretas MAC, chaves de criptografia de sessão e IVs) é definido assim:

```
key_block = PRF(master_secret, "key expansion",
                 SecurityParameters.server_random ||
                 SecurityParameters.client_random)
```

até que uma saída suficiente tenha sido gerada. Assim como no SSLv3, key_block é uma função de master_secret e os números aleatórios de cliente e servidor, mas, para o TLS, o algoritmo real é diferente.

Complementação No SSL, a complementação acrescentada antes da criptografia de dados do usuário é a quantidade mínima exigida para que o tamanho total dos dados a serem criptografados seja um múltiplo do tamanho de

bloco da cifra. No TLS, a complementação pode ser qualquer quantidade que resulte em um total que seja um múltiplo do tamanho de bloco da cifra, até um máximo de 255 bytes. Por exemplo, se o texto claro (ou texto compactado, se for usada compactação) mais MAC mais bytes do tamanho do preenchimento tiver 79 bytes de extensão, então o tamanho do preenchimento, em bytes, pode ser 1, 9, 17 e assim por diante, até 249. Um tamanho de complementação variável pode ser usado para frustrar ataques com base em uma análise dos tamanhos das mensagens trocadas.

17.3 SECURE ELECTRONIC TRANSACTION

O SET é uma especificação aberta de criptografia e segurança, criada para proteger transações envolvendo cartões de crédito pela Internet. A versão atual, SETv1, surgiu de um pedido de padrões de segurança pela MasterCard e Visa em fevereiro de 1996. Diversas empresas estiveram envolvidas no desenvolvimento da especificação inicial, incluindo IBM, Microsoft, Netscape, RSA, Terisa e Verisign. A partir de 1996 ocorreram diversos testes de conceito e por volta de 1998 a primeira onda de produtos compatíveis com o SET estava disponível.

O SET não é, por si só, um sistema de pagamentos. Em vez disso, é um conjunto de protocolos e formatos de segurança que permite que os usuários empreguem a infra-estrutura de pagamento por cartão de crédito existente em uma rede aberta, como a Internet, de modo seguro. Basicamente, o SET oferece três serviços:

- Um canal de comunicações seguro entre todas as partes envolvidas em uma transação
- Confiança no uso de certificados digitais X.509v3
- Privacidade, pois as informações só estão disponíveis às partes em uma transação quando e onde forem necessárias

O SET é uma especificação complexa, definida em três livros emitidos em maio de 1997:

- **Livro 1:** Descrição do negócio (80 páginas)
- **Livro 2:** Guia do programador (629 páginas)
- **Livro 3:** Definição formal do protocolo (262 páginas)

Isso compreende um total de 971 páginas de especificação. Em contraste, a especificação do SSLv3 tem 63 páginas e a especificação do TLS tem 80 páginas. Por conseguinte, nesta seção, fornecemos apenas um resumo dessa especificação de muitas facetas.

Visão geral do SET

Uma boa maneira de começar nossa explicação do SET é examinar os requisitos de negócios do SET, seus principais recursos e os participantes das transações SET.

Requisitos O Livro 1 da especificação SET lista os seguintes requisitos de negócios para o processamento de pagamento seguro com cartões de crédito pela Internet e outras redes:

- **Fornecer confidencialidade das informações de pagamento e de pedido:** É necessário garantir aos proprietários do cartão que essas informações são seguras e acessíveis apenas pelo destinatário pretendido. A confidencialidade também reduz o risco de fraude por uma das partes da transação ou por terceiros maliciosos. O SET utiliza criptografia para fornecer confidencialidade.
- **Garantir a integridade de todos os dados transmitidos:** Isto é, garantir que nenhuma mudança no conteúdo ocorra durante a transmissão de mensagens SET. Assinaturas digitais são usadas para fornecer integridade.
- **Fornecer autenticação de que o proprietário do cartão é um usuário legítimo de uma conta de cartão de crédito:** Um mecanismo que vincula o proprietário do cartão a um número de conta específico reduz a incidência de fraudes e o custo geral de processamento do pagamento. Assinaturas e certificados digitais são usados para verificar se um proprietário de cartão é um usuário legítimo de uma conta válida.
- **Oferecer autenticação de que um comerciante pode aceitar transações de cartão de crédito por meio de seu relacionamento com uma instituição financeira:** Esse é um complemento do requisito anterior. Os proprietários de cartão precisam ser capazes de identificar os comerciantes com quem podem realizar transações seguras. Novamente, são usados assinaturas e certificados digitais.
- **Garantir o uso das melhores práticas de segurança e das melhores técnicas de projeto de sistemas para proteger todas as partes legítimas em uma transação de comércio eletrônico:** O SET é uma especificação bem testada, com base em algoritmos criptográficos e protocolos altamente seguros.
- **Criar um protocolo que não dependa dos mecanismos de segurança de transporte nem impeça seu uso:** O SET pode operar com segurança sobre uma pilha TCP/IP ‘bruta’. Porém, o SET não interfere no uso de outros mecanismos de segurança, como IPSec e SSL/TLS.
- **Facilitar e encorajar a interoperabilidade entre provedores de software e rede:** Os protocolos e formatos SET são independentes da plataforma de hardware, sistema operacional e software Web.

Principais recursos do SET

Para atender os requisitos esboçados acima, o SET incorpora os seguintes recursos:

- **Confidencialidade de informações:** Informações da conta do proprietário do cartão e do pagamento são protegidas enquanto trafegam pela rede. Um recurso interessante e importante do SET é que ele impede que o comerciante descubra o número de cartão de crédito do proprietário do cartão; isso só é fornecido ao banco emissor. A criptografia convencional pelo DES é usada para fornecer confidencialidade.
- **Integridade dos dados:** As informações de pagamento enviadas dos proprietários de cartão para os comerciantes incluem informações do pedido, dados pessoais e instruções de pagamento. O SET garante que esse conteúdo de mensagem não seja alterado em trânsito. Assinaturas digitais RSA, incluindo códigos de hash SHA-1, oferecem integridade de mensagem. Certas mensagens também são protegidas por HMAC usando SHA-1.
- **Autenticação de conta do proprietário do cartão:** O SET permite que os comerciantes verifiquem se um proprietário de cartão é um usuário legítimo de um número de conta de cartão válido. O SET utiliza certificados digitais X.509v3 com assinaturas RSA para essa finalidade.
- **Autenticação do comerciante:** O SET permite que os proprietários de cartão verifiquem se um comerciante tem um relacionamento com uma instituição financeira, permitindo que ele aceite cartões de pagamentos. O SET utiliza certificados digitais X.509v3 com assinaturas RSA para essa finalidade.

Observe que, diferentemente de IPSec e SSL/TLS, o SET oferece apenas uma escolha para cada algoritmo criptográfico. Isso faz sentido, pois o SET é uma aplicação isolada, com um conjunto de requisitos exclusivo, enquanto o IPSec e o SSL/TLS buscam oferecer suporte a uma grande variedade de aplicações.

Participantes do SET A Figura 17.8 indica os participantes do sistema SET, que incluem os seguintes:

- **Proprietário do cartão:** No ambiente eletrônico, consumidores e compradores corporativos interagem com comerciantes a partir de computadores pessoais pela Internet. Um proprietário de cartão é um proprietário autorizado de um cartão de pagamentos (por exemplo, MasterCard, Visa) emitido por um emissor.
- **Comerciante:** Um comerciante é uma pessoa ou organização que possui bens ou serviços para vender ao proprietário do cartão. Normalmente, esses bens e serviços são oferecidos por meio de um site Web ou por e-mail. Um comerciante que aceita cartões de pagamentos precisa ter um relacionamento com um acquirer.
- **Emissor:** A instituição financeira, como um banco, que oferece um cartão de pagamentos ao proprietário do cartão. Normalmente, as contas são solicitadas e abertas por correio ou pessoalmente. Em última instância, é o emissor que é responsável pelo pagamento do débito do proprietário do cartão.
- **Acquirer:** Instituição financeira que estabelece uma conta com um comerciante e processa autorizações de cartão de pagamentos e pagamentos. Os comerciantes normalmente aceitarão mais de uma bandeira de cartão de crédito, mas não desejam lidar com as várias redes de cartão bancário ou com os vários emissores individuais. O acquirer oferece autenticação ao comerciante, indicando que determinada conta de cartão está ativa e que a compra proposta não ultrapassa o limite de crédito. O acquirer também oferece transferência eletrônica de pagamentos à conta do comerciante. Posteriormente, o acquirer é reembolsado pelo emissor, por meio de algum tipo de rede de pagamentos para transferência eletrônica de fundos.
- **Gateway de pagamento:** Função operada pelo acquirer ou por um terceiro designado que processa mensagens de pagamento do comerciante. O gateway de pagamento realiza a interface entre o SET e as redes de pagamento de cartão bancário existentes, para funções de autorização e pagamento. O comerciante troca mensagens SET com o gateway de pagamento pela Internet, enquanto o gateway de pagamento possui alguma conexão direta ou de rede com o sistema de processamento financeiro do acquirer.
- **Autoridade certificadora (CA):** Entidade confiável para emitir certificados de chave pública X.509v3 para proprietários de cartão, comerciantes e gateways de pagamento. O sucesso do SET dependerá da existência de uma infra-estrutura de CA disponível para essa finalidade. Conforme explicamos nos capítulos anteriores, é utilizada uma hierarquia de CAs, de modo que os participantes não precisam ser certificados diretamente por uma autoridade-raiz.

Agora, descrevemos rapidamente a seqüência de eventos exigidos para uma transação. Depois, veremos alguns dos detalhes criptográficos.

1. **O cliente abre uma conta.** O cliente obtém uma conta de cartão de crédito, por exemplo, MasterCard ou Visa, com um banco que aceita pagamentos eletrônicos e SET.
2. **O cliente recebe um certificado.** Após a verificação apropriada da identidade, o cliente recebe um certificado digital X.509v3, o qual é assinado pelo banco. O certificado atesta a chave pública RSA do cliente e sua data de expiração. Ele também estabelece uma correlação, garantida pelo banco, entre o par de chaves do cliente e seu cartão de crédito.

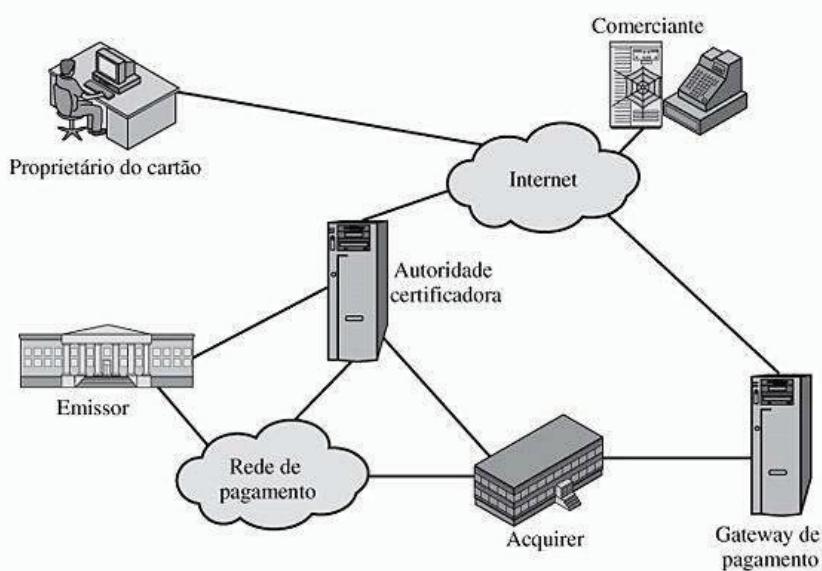


Figura 17.8 Componentes do comércio eletrônico seguro.

3. **Os comerciantes têm seus próprios certificados.** Um comerciante que aceita uma certa bandeira de cartão precisa possuir dois certificados para as duas chaves públicas pertencentes ao comerciante: uma para assinar mensagens e uma para troca de chaves. O comerciante também precisa de uma cópia do certificado de chave pública do gateway de pagamento.
4. **O cliente faz um pedido.** Esse é um processo envolvendo o cliente que, em primeiro lugar, navega pelo site Web do comerciante para selecionar itens e saber seus preços. O cliente então envia uma lista dos itens a serem comprados para o comerciante que, em retorno, fornece um formulário de pedido, o qual contém a lista de itens, seus preços, um preço total e um número do pedido.
5. **O comerciante é verificado.** Além do formulário de pedido, o comerciante envia uma cópia de seu certificado, para que o cliente possa verificar se está lidando com uma loja válida.
6. **O pedido e o pagamento são enviados.** O cliente envia o pedido e as informações de pagamento ao comerciante, juntamente com o certificado do cliente. O pedido confirma a compra dos itens no formulário de pedido. O pagamento contém detalhes do cartão de crédito. As informações de pagamento são criptografadas de modo que não possam ser lidas pelo comerciante. O certificado do cliente permite que o comerciante confirme a identidade do cliente.
7. **O comerciante solicita autorização de pagamento.** O comerciante envia as informações de pagamento ao gateway de pagamento, solicitando confirmação de que o crédito disponível do cliente é suficiente para essa compra.
8. **O comerciante confirma o pedido.** O comerciante envia a confirmação do pedido ao cliente.
9. **O comerciante fornece os bens ou serviços.** O comerciante envia os bens ou fornece o serviço ao cliente.
10. **O comerciante solicita o pagamento.** Essa solicitação é enviada ao gateway de pagamento, que trata de todo o processamento do pagamento.

Assinatura dual

Antes de examinarmos os detalhes do protocolo SET, vamos explicar uma inovação importante apresentada no SET: a assinatura dual (dual signature). A finalidade da assinatura dual é vincular duas mensagens destinadas a dois destinatários diferentes. Nesse caso, o cliente deseja enviar as informações do pedido (OI) para o comerciante e a informação de pagamento (PI) para o banco. O comerciante não precisa saber o número do cartão de crédito do cliente, e o banco não precisa saber os detalhes do pedido do cliente. O cliente recebe proteção extra em termos de privacidade mantendo esses dois itens separados. Porém, os dois itens precisam ser vinculados de modo que possam ser usados para resolver disputas, se for preciso. O vínculo é necessário para que o cliente possa provar que esse pagamento é para esse pedido e não para algum outro bem ou serviço.

Para perceber a necessidade do vínculo, suponha que os clientes enviem duas mensagens ao comerciante: uma OI assinada e uma PI assinada, e o comerciante passe a PI para o banco. Se o comerciante puder captar outra OI desse cliente, ele pode afirmar que essa OI acompanha a PI, em vez da OI original. O vínculo evita isso.

A Figura 17.9 mostra o uso de uma assinatura dual para atender ao requisito do parágrafo anterior. O cliente apanha o hash (usando SHA-1) da PI e o hash da OI. Esses dois hashes são então concatenados e é calculado o hash

do resultado. Finalmente, o cliente criptografa o hash final com sua chave privada para assinatura, criando a assinatura dual. A operação pode ser resumida como

$$DS = E(PR_c, [H(H(PI) \parallel H(OI))])$$

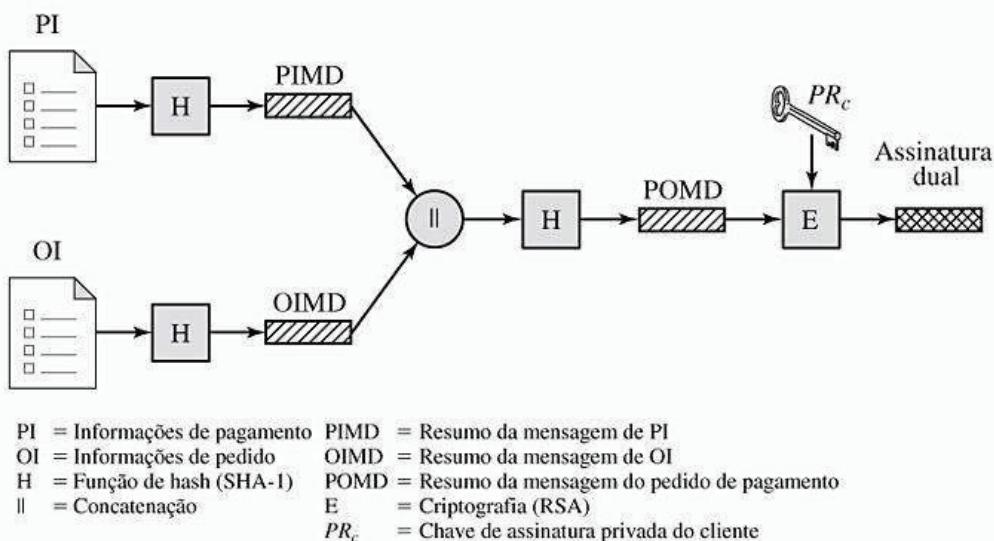


Figura 17.9 Construção da assinatura dual.

onde PR_c é a chave privada do cliente para assinatura. Agora, suponha que o comerciante possua a assinatura dual (DS) da OI e do resumo da mensagem da PI (PIMD). O comerciante também tem a chave pública do cliente, obtida do certificado do cliente. Então, o comerciante pode calcular os valores

$$H(PIMD \parallel H[OI]); D(PU_c, DS)$$

onde PU_c é a chave pública do cliente para assinatura. Se esses dois valores forem iguais, então o comerciante confirmou a assinatura. De modo semelhante, se o banco possuir DS, PI, o resumo da mensagem da OI (OIMD) e a chave pública do cliente, então o banco calculará

$$H(H[OI] \parallel OIMD); D(PU_c, DS)$$

Novamente, se esses dois valores forem iguais, então o banco confirmou a assinatura. Resumindo,

1. O comerciante recebe a OI e confirma a assinatura.
2. O banco recebe a PI e confirma a assinatura.
3. O cliente vinculou a OI à PI e pode comprovar essa relação.

Por exemplo, suponha que o comerciante queira mudar a OI nessa transação para tirar algum proveito. Assim, ele teria que encontrar outra OI cujo hash fosse igual ao hash da OI atual (OIMD). Com SHA-1, isso é considerado inviável. Assim, o comerciante não pode vincular outra OI a essa PI.

Processamento de pagamento

A Tabela 17.3 lista os tipos de transação admitidos pelo SET. A seguir, vemos alguns detalhes nas seguintes transações:

- Solicitação de compra
- Autorização de pagamento
- Captação de pagamento

Solicitação de compra Antes que a transação de solicitação de compra se inicie, o proprietário do cartão já concluiu a navegação, seleção e pedido. O final dessa fase preliminar ocorre quando o comerciante envia um formulário de pedido concluído ao cliente. Todo esse processamento anterior ocorre sem o uso do SET.

A transação de solicitação de compra consiste em quatro mensagens: iniciar solicitação, iniciar resposta, solicitação de compra e resposta de compra.

Tabela 17.3 Tipos de transação SET

Registro do proprietário do cartão	Os proprietários de cartão precisam ser registrados em uma CA antes que possam enviar mensagens SET aos comerciantes.
Registro do comerciante	Os comerciantes precisam ser registrados em uma CA antes que possam trocar mensagens SET com clientes e gateways de pagamento.
Solicitação de compra	Mensagem do cliente a comerciante, contendo a OI para o comerciante e a PI para o banco.
Autorização de pagamento	Troca entre comerciante e gateway de pagamento para autorizar determinado valor para uma compra em determinada conta de cartão de crédito.
Captação de pagamento	Permite que o comerciante solicite o pagamento do gateway de pagamento.
Consulta e status de certificado	Se a CA for incapaz de concluir o processamento de uma solicitação de certificado rapidamente, ela enviará uma resposta ao proprietário do cartão ou comerciante, indicando que o solicitante deverá verificar novamente mais tarde. O proprietário do cartão ou comerciante envia a mensagem de consulta de certificado para determinar o status da solicitação de certificado e receber o certificado se a solicitação tiver sido aprovada.
Consulta de compra	Permite que o proprietário do cartão verifique o status do processamento de um pedido depois que a resposta da compra tiver sido recebida. Observe que essa mensagem não inclui informações como o status de bens que não estejam em estoque, mas sim o status da autorização, captação e processamento de crédito.
Cancelamento da autorização	Permite que um comerciante corrija solicitações de autorização anteriores. Se o pedido não for completado, o comerciante cancela a autorização inteira. Se parte do pedido não for completa (como quando os bens não estão em estoque), o comerciante cancela parte do valor da autorização.
Cancelamento de captação	Permite que um comerciante corrija erros nas solicitações de captação, como valores de transação que foram inseridos incorretamente por um funcionário.
Crédito	Permite que um comerciante emita um crédito para a conta de um proprietário de cartão, por exemplo quando os bens tenham sido devolvidos ou danificados durante a entrega. Observe que a mensagem de crédito do SET sempre é iniciada pelo comerciante, e não pelo proprietário do cartão. Toda a comunicação entre o proprietário do cartão e o comerciante que resulta no processamento de um crédito acontece fora do SET.
Cancelamento de crédito	Permite que um comerciante corrija um crédito solicitado anteriormente.
Solicitação de certificado de gateway de pagamento	Permite que um comerciante consulte o gateway de pagamento e receba uma cópia dos certificados atuais de troca de chaves e de assinatura do gateway.
Administração em lote	Permite que um comerciante comunique informações ao gateway de pagamento com relação a lotes do comerciante.
Mensagem de erro	Indica que um respondedor rejeitou uma mensagem porque há falha nos testes de verificação de formato ou de conteúdo.

Para enviar mensagens SET ao comerciante, o proprietário do cartão precisa ter uma cópia dos certificados do comerciante e do gateway de pagamento. O cliente solicita os certificados na mensagem **Iniciar Solicitação** (Initiate Request), enviada ao comerciante. Essa mensagem inclui a bandeira do cartão de crédito que o cliente está usando. A mensagem também inclui uma ID atribuída a esse par de solicitação/resposta pelo cliente e um nonce usado para garantir o correto posicionamento em relação ao tempo.

O comerciante gera uma resposta e a assina com sua chave de assinatura privada. A resposta inclui o nonce do cliente, outro nonce para o cliente retornar na próxima mensagem e uma ID de transação para essa transação de compra. Além da resposta assinada, a mensagem **Iniciar Resposta** (Initiate Response) inclui o certificado de assinatura do comerciante e o certificado de troca de chaves do gateway de pagamento.

O proprietário do cartão confirma os certificados do comerciante e do gateway por meio de respectivas assinaturas nos seus certificados feitas pela CA e depois cria a OI e a PI. A ID da transação atribuída pelo comerciante é colocada tanto na OI quanto na PI. A OI não contém dados explícitos do pedido, como o número e o preço dos itens. Em vez disso, ela contém uma referência do pedido gerada na troca entre comerciante e cliente durante a fase de compra, antes da primeira mensagem SET. Em seguida, o proprietário do cartão prepara a mensagem **Solicitação de Compra** (Purchase Request) (Figura 17.10). Para essa finalidade, o proprietário do cartão gera uma chave de criptografia simétrica de uso único, K_s . A mensagem inclui o seguinte:

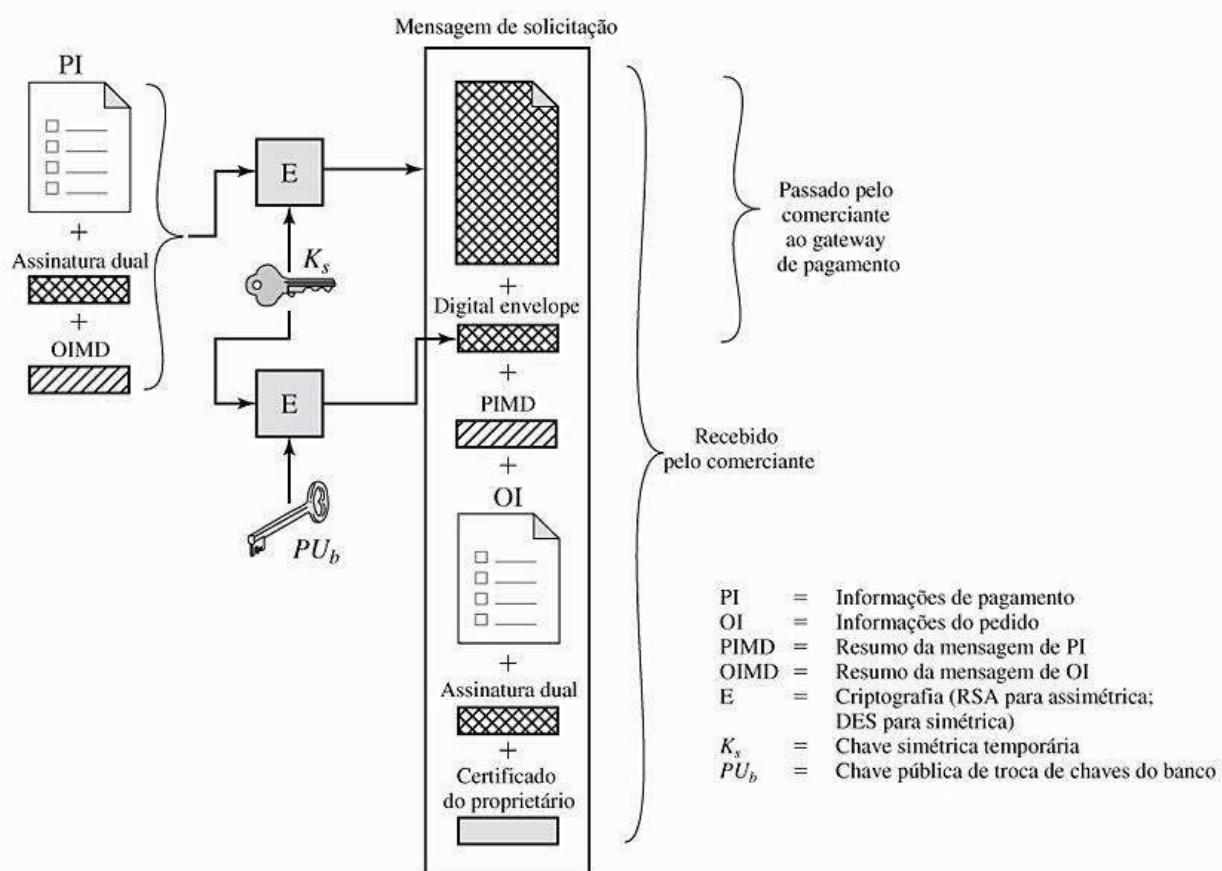


Figura 17.10 O proprietário do cartão envia a solicitação de compra.

1. **Informações relacionadas à compra.** Essas informações serão encaminhadas ao gateway de pagamento pelo comerciante e contêm:

- PI
- Assinatura dual, calculada a partir da PI e da OI, assinada com a chave privada de assinatura do cliente
- O resumo da mensagem de OI (OIMD)

O OIMD é necessário para o gateway de pagamento verificar a assinatura dual, conforme explicado anteriormente. Todos esses itens são criptografados com K_s . O item final é

- O envelope digital. Este é formado criptografando-se K_s com a chave pública de troca de chaves do gateway de pagamento. Ele é chamado de envelope digital porque esse envelope precisa ser aberto (decryptografado) antes que os outros itens listados anteriormente possam ser lidos. O valor de K_s não se torna disponível ao comerciante. Portanto, o comerciante não pode ler qualquer uma dessas informações relacionadas ao pagamento.

2. **Informações relacionadas ao pedido.** Essas informações são necessárias para o comerciante e são formadas pela

- OI
- Assinatura dual, calculada a partir da PI e da OI, assinada com a chave privada de assinatura do cliente

- O resumo da mensagem de PI (PIMD) O PIMD é necessário para que o comerciante verifique a assinatura dual. Observe que a OI é enviada às claras.
3. **Certificado do proprietário do cartão.** Este contém a chave pública de assinatura do proprietário do cartão. Ele é necessário para o comerciante e para o gateway de pagamento.

Quando o comerciante recebe a mensagem de solicitação de compra, ele realiza as seguintes ações (Figura 17.11):

1. Confirma os certificados do proprietário do cartão por meio das respectivas assinaturas realizadas pela CA.
2. Confirma a assinatura dual usando a chave pública de assinatura do cliente. Isso garante que o pedido não foi alterado em trânsito e que foi assinado usando a chave privada de assinatura do proprietário do cartão.
3. Processa o pedido e encaminha as informações de pagamento ao gateway de pagamento para autorização (descrito mais adiante).
4. Envia uma resposta de compra ao proprietário do cartão.

A mensagem de **Resposta de Compra** inclui um bloco de resposta que confirma o pedido e referencia o número da transação correspondente. Esse bloco é assinado pelo comerciante usando sua chave privada de assinatura. O bloco e sua assinatura são enviados ao cliente juntamente com o certificado de assinatura do comerciante.

Quando o programa do proprietário do cartão recebe a mensagem de resposta de compra, ele verifica o certificado do comerciante e depois verifica a assinatura no bloco de resposta. Finalmente, ele toma alguma atitude com base na resposta, como exibir uma mensagem ao usuário ou atualizar um banco de dados com o status do pedido.

Autorização de pagamento Durante o processamento de um pedido de um proprietário de cartão, o comerciante consegue a autorização da transação com o gateway de pagamento. A autorização de pagamento garante que a transação foi aprovada pelo emissor e garante que o comerciante receberá o pagamento; o comerciante pode, portanto, fornecer os serviços ou bens ao cliente. A troca de autorização de pagamento consiste em duas mensagens: **Solicitação de Autorização** (Authorization Request) e **Resposta de Autorização** (Authorization Response).

O comerciante envia uma mensagem de **Solicitação de Autorização** ao gateway de pagamento, formada por:

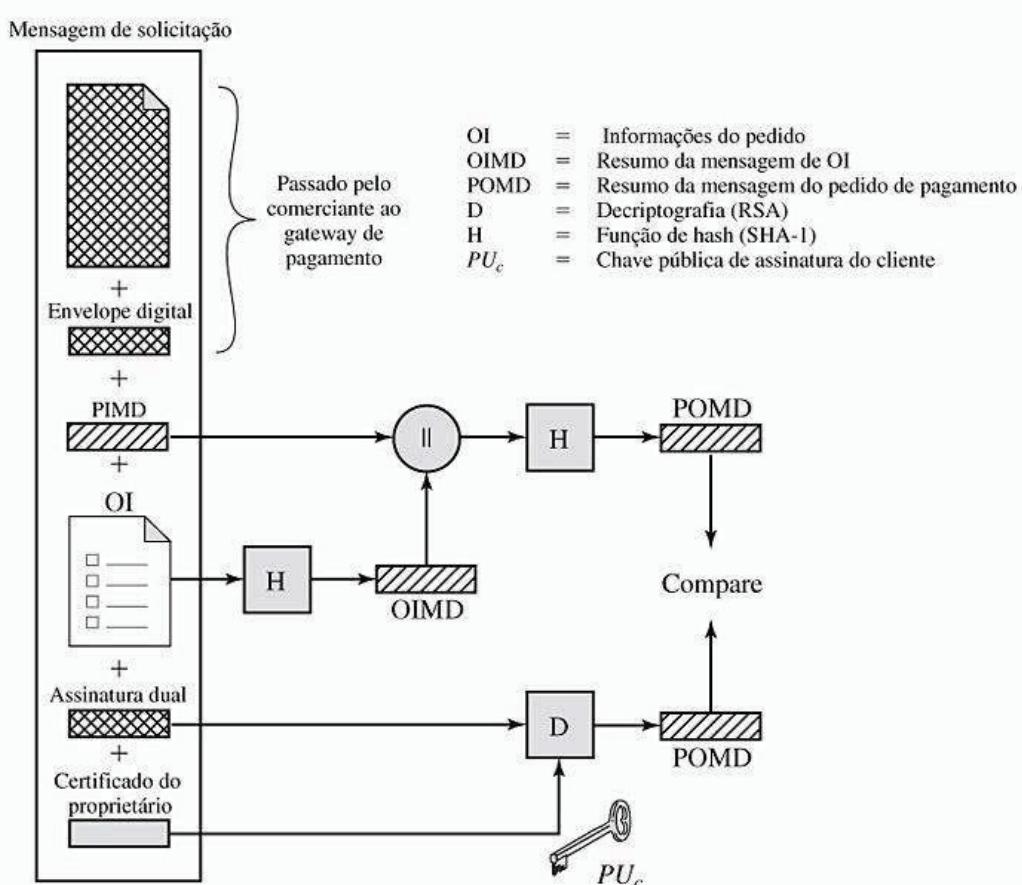


Figura 17.11 Comerciante verifica a solicitação de compra do cliente.

1. **Informações relacionadas à compra.** Essas informações foram obtidas do cliente e consistem em
 - A PI
 - A assinatura dual, calculada a partir da PI e da OI, assinada com a chave privada de assinatura do cliente
 - O resumo da mensagem de OI (OIMD)
 - O envelope digital
2. **Informações relacionadas à autorização.** Essas informações são geradas pelo comerciante e consistem em
 - Um bloco de autorização que inclui a ID da transação, assinado com a chave privada de assinatura do comerciante e criptografado com uma chave simétrica de uso único pelo comerciante.
 - Um envelope digital formado pela criptografia da chave de uso único usando a chave pública de troca de chaves do gateway de pagamento.
3. **Certificados.** O comerciante inclui o certificado de chave de assinatura do proprietário do cartão (usado para verificar a assinatura dual), o certificado de chave de assinatura do comerciante (usado para verificar a assinatura do comerciante) e o certificado de troca de chaves do comerciante (necessário na resposta do gateway de pagamento).

O gateway de pagamento realiza as seguintes tarefas:

1. Verifica todos os certificados
2. Decriptografa o envelope digital do bloco de autorização para obter a chave simétrica e depois decriptografa o bloco de autorização
3. Verifica a assinatura do comerciante no bloco de autorização
4. Decriptografa o envelope digital do bloco de pagamento para obter a chave simétrica e depois decriptografa o bloco de pagamento
5. Verifica a assinatura dual no bloco de pagamento
6. Verifica se a ID da transação recebida do comerciante combina com a da PI recebida (indiretamente) do cliente
7. Solicita e recebe uma autorização do emissor

Tendo obtido autorização do emissor, o gateway de pagamento retorna uma mensagem de **Resposta de Autorização** (Authorization Response) ao comerciante. Ela inclui os seguintes elementos:

1. **Informações relacionadas à autorização.** Inclui um bloco de autorização, assinado com a chave privada de assinatura do gateway e criptografado com uma chave simétrica de uso único, gerada pelo gateway. Também inclui um envelope digital que contém a chave de uso único criptografada com a chave pública de troca de chaves do comerciante.
2. **Informações do token de captação.** Essas informações serão usadas para efetuar o pagamento posteriormente. Esse bloco tem o mesmo formato de (1), a saber, um token de captação assinado, criptografado, juntamente com um envelope digital. Esse token não é processado pelo comerciante. Em vez disso, ele precisa ser retornado, como se encontra, com uma solicitação de pagamento.
3. **Certificado.** O certificado de chave de assinatura do gateway.

Com a autorização do gateway, o comerciante pode fornecer os bens ou serviços ao cliente.

Captação de pagamento Para obter o pagamento, o comerciante inicia com o gateway de pagamento uma transação de captura de pagamento, formada por uma mensagem de solicitação de captura e uma mensagem de resposta de captura.

Para a mensagem de **Solicitação de Captação** (Capture Request), o comerciante gera, assina e criptografa um bloco de solicitação de captação, que inclui o valor do pagamento e a ID da transação. A mensagem também inclui o token de captação criptografado recebido anteriormente (na Resposta de Autorização) para essa transação, além da chave de assinatura do comerciante e dos certificados de troca de chaves.

Quando o gateway de pagamento recebe a mensagem de solicitação de captação, ele decifra e verifica o bloco de solicitação de captação e decifra e verifica o bloco do token de captação. Depois, ele verifica a consistência entre a solicitação e o token de captação. Depois, cria uma solicitação de compensação que é enviada ao emissor pela rede de pagamento privada. Essa solicitação faz com que sejam transferidos fundos para a conta do comerciante.

O gateway, então, notifica o pagamento ao comerciante em uma mensagem de **Resposta de Captação** (Capture Response). A mensagem inclui um bloco de resposta de captação que o gateway assina e criptografa. A mensagem também inclui o certificado de chave de assinatura do gateway. O software do comerciante armazena a resposta de captação a ser usada para reconciliação com o pagamento recebido do acquirer.

17.4 LEITURA E SITES WEB RECOMENDADOS

[RESC01] é um boa explicação detalhada sobre SSL e TLS.

A visão geral mais detalhada do SET está no Livro 1 da especificação, disponível no site Web sobre SET da MasterCard. Outra excelente visão geral é [MACG97]. [DREW99] também é uma boa fonte.

DREW99 Drew, G. *Using SET for secure electronic commerce*. Upper Saddle River: Prentice Hall, 1999.

MACG97 Macgregor, R.; Ezvan, C.; Liguori, L. e Han, J. *Secure electronic transactions: credit card payment on the Web in theory and practice*. IBM RedBook SG24-4978-00, 1997. Disponível em www.redbooks.ibm.com.

RESC01 Rescorla, E. *SSL and TLS: designing and building secure systems*. Reading: Addison-Wesley, 2001.

Sites Web recomendados:



- **Netscape's SSL Page:** Contém a especificação SSL.
- **Transport Layer Security Charter:** RFCs e drafts de Internet mais recentes para TLS.
- **OpenSSL Project:** Um projeto para desenvolver software SSL e TLS de fonte aberta. O site inclui documentos e links.

17.5 PRINCIPAIS TERMOS, PERGUNTAS DE REVISÃO E PROBLEMAS

Principais termos

acquirer
assinatura dual
autoridade certificadora (CA)
comerciante

emissor
gateway de pagamento
proprietário do cartão
Secure Electronic Transaction (SET)

Secure Socket Layer (SSL)
Transport Layer Security (TLS)

Perguntas de revisão

- | | | | |
|------|--|------|--|
| 17.1 | Quais são as vantagens de cada uma das três abordagens mostradas na Figura 17.1? | 17.6 | definem uma conexão de sessão SSL. |
| 17.2 | Que protocolos estão incorporados ao SSL? | 17.7 | Que serviços são fornecidos pelo Protocolo de Registro SSL? |
| 17.3 | Qual é a diferença entre uma conexão SSL e uma sessão SSL? | 17.8 | Que etapas estão envolvidas na transmissão do Protocolo de Registro SSL? |
| 17.4 | Liste e defina resumidamente os parâmetros que definem um estado de sessão SSL. | 17.9 | Liste e defina resumidamente as principais categorias de participantes do SET. |
| 17.5 | Liste e defina resumidamente os parâmetros que | | O que é uma assinatura dual e qual é sua finalidade? |

Problemas

- 17.1 No SSL e TLS, por que existe um Protocolo de Mudança de Especificação de Cifra separado, em vez de incluir uma mensagem `change_cipher_spec` no Protocolo de Estabelecimento de Sessão?
- 17.2 Considere as seguintes ameaças à segurança na Web e descreva como cada uma é impedida por um recurso específico do SSL.
 - a. Ataque criptoanalítico por força bruta: Um teste completo de todas as chaves possíveis para um algoritmo de criptografia convencional.
 - b. Ataque de dicionário com texto claro conhecido: Muitas mensagens terão texto claro previsível, como o comando GET do HTTP. Um atacante cria um dicionário contendo cada criptografia possível da mensagem de texto claro conhecido. Quando uma mensagem criptografada é

- interceptada, o atacante apanha a parte contendo o texto claro conhecido criptografado e pesquisa o texto cifrado no dicionário. O texto cifrado deverá ser igual a uma entrada criptografada com a mesma chave secreta. Se for igual a mais de uma, cada uma delas pode ser experimentada em relação ao texto cifrado completo para determinar a entrada correta. Esse ataque é especialmente eficaz quanto a tamanhos de chave pequenos (por exemplo, 40 bits).
- c. Ataque por repetição: Mensagens de estabelecimento de conexão SSL anteriores são repetidas.
- d. Ataque de homem no meio (man-in-the-middle): Um atacante se interpõe durante a troca de chave, atuando como cliente perante o servidor e como servidor perante o cliente.

- e. Sniffing de senha: As senhas em HTTP ou outro tráfego de aplicação são interceptadas sem autorização.
- f. Falsificação do IP (IP spoofing): Usa endereços de IP forjados para enganar um host para aceitar dados falsos.
- g. Seqüestro de IP (IP hijacking): Uma conexão ativa, autenticada, entre dois hosts é interrompida e o atacante toma o lugar de um dos hosts.
- h. Inundação de SYN (SYN flooding): Um atacante envia mensagens SYN do TCP para solicitar uma conexão, mas não responde à mensagem

final para estabelecer a conexão totalmente. O módulo TCP atacado normalmente deixa a conexão ‘meio aberta’ por alguns minutos. As mensagens SYN repetidas podem congestionar os módulos TCP.

- 17.3 Com base no que você aprendeu neste capítulo, é possível no SSL que o receptor reordene blocos de registro SSL que chegam fora de ordem? Em caso positivo, explique como isso pode ser feito. Em caso negativo, por que não?