Segurança da Informação– GBC083

Prof. Rodrigo Sanches Miani – FACOM/UFU

Segurança na camada de transporte

Segurança da Informação– GBC083

Objetivo

- Estudar o protocolo SSL/TLS, utilizado para prover segurança no nível de transporte.

- Um dos recursos mais utilizados nas transações da Web.

Tópicos

1. Introdução

2. SSL
   - Breve histórico
   - Camadas
   - Sessão SSL
   - Protocolos (handshake, change cypher, alert e record)

Introdução

Segurança da Informação– GBC083

Introdução

- Praticamente todas as empresas, a maioria das agências do governo, e muitos indivíduos, agora possuem Websites;
- O número de indivíduos e empresas com acesso à Internet não para de crescer e praticamente todos usam navegadores Web;
- Comércio eletrônico é um dos inúmeros exemplos de aplicação desse cenário;
- Internet e a Web são extremamente vulneráveis a comprometimentos de vários tipos.

Introdução

- Desafios da segurança na Web:
  - A Web é uma grande plataforma de negócios.
  - Navegadores Web são fáceis de usar, portais Web são fáceis de desenvolver, servidores Web são fáceis de instalar... garantir segurança é complexo.
  - Um servidor Web, caso explorado, pode ser a porta de entrada para o restante da rede.
  - Usuários de sistemas Web são, em geral, leigos ou pouco capacitados para uso de informática.

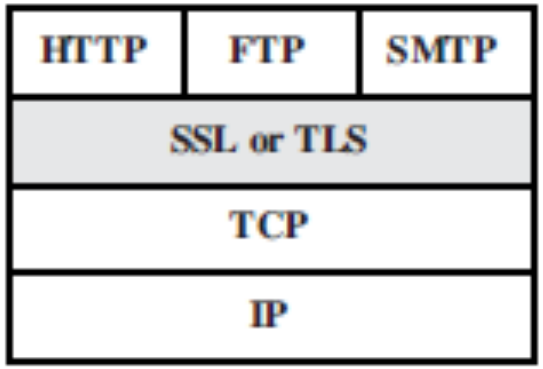
Ameaças na Web

	Ameaças	Consequências	Contramedidas
Integridade	■ Modificação de dados do usuário ■ Navegador cavalo de Tróia ■ Modificação de memória ■ Modificação de tráfego de mensagem em trânsito	■ Perda de informações ■ Comprometimento da máquina ■ Vulnerabilidade a todas as outras ameaças	Checksums criptográficos
Confidencialidade	■ Espionagem na rede ■ Roubo de informações do servidor ■ Roubo de dados do cliente ■ Informações sobre configuração de rede ■ Informações sobre qual cliente fala com o servidor	■ Perda de informações ■ Perda de privacidade	Criptografia, proxies Web
Negação de serviço	■ Encerramento de threads do usuário ■ Inundação da máquina com solicitações falsas ■ Preenchimento do disco ou da memória ■ Isolamento da máquina por ataques de DNS	■ Interrupção ■ Incômodo ■ Impede que usuário realize o trabalho	Difícil de impedir
Autenticação	■ Personificação de usuários legítimos ■ Falsificação de dados	■ Má representação do usuário ■ Crença de que informações falsas são válidas	Técnicas criptográficas

Soluções para Segurança na Web

- *IPsec* - segurança no nível de rede:
  - Pode ser transparente para usuários finais.
  - Filtra o tráfego que necessita de segurança.

- *SSL/TLS* – segurança na camada de transporte:



(b) Transport level

Breve histórico

Segurança da Informação– GBC083

SSL/TLS

- SSL (Secure Socket Layer) foi criado pela Netscape;
- A versão 3 foi publicada como um Internet draft;
- IETF formou grupo de trabalho denominado TLS (Transport Layer Security), que publicou a primeira versão do TLS;
- TLSv1 é praticamente um SSLv3.1.

TLS (Transport Layer Security)

- Trabalho da IETF para padronizar o SSL.
- TLS vs 1: especificado no RFC 2246 (1999).
- TLS vs 1.2 especificado no RFC 5246 (2008).
- TLS vs 1.3: finalizado! RFC 8446
  - https://tools.ietf.org/html/rfc8446
  - Diversas modificações:
  - https://medium.com/@vanrijn/what-is-new-with-tls-1-3-e991df2caaac
  - https://kinsta.com/blog/tls-1-3/

TLS 1.2 x TLS 1.3

- Desempenho do handshake é bem melhor…
  - Número de passos diminuiu e agora parâmetros enviados pelo servidor podem estar cifrados.

- Remoção de funções inseguras.
  - MD5, SHA-1, DES, 3DES…
  - RSA Key Transport!!
    - https://www.theinquirer.net/inquirer/news/2343117/ietf-drops-rsa-key-transport-from-ssl
    - https://blog.trailofbits.com/2019/07/08/fuck-rsa/

HTTPS

- HTTPS = HTTP over SSL.
- Combinação de HTTP e SSL para prover comunicação segura entre navegadores e servidores.
- URL iniciada por "HTTPS" ao invés de "HTTP".
- Normalmente, usa porta 443, ao invés da porta 80.

Camadas

Segurança da Informação– GBC083

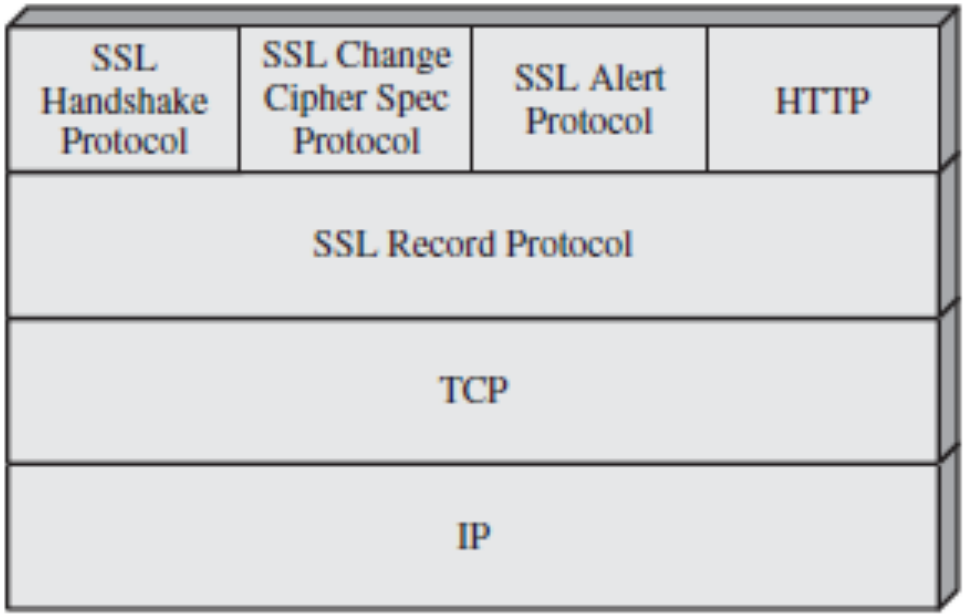
SSL/TLS

- SSL/TLS provê, de modo geral, as seguintes soluções de segurança para uma conexão entre duas aplicações (camada de transporte – TCP):
  - Confidencialidade.
  - Integridade.
  - Autenticação.

- SSL não é um protocolo isolado, mas duas camadas de protocolo.

SSL

- SSL tem duas camadas de protocolos;



Sessão SSL

Segurança da Informação– GBC083

SSL

- Dois conceitos importantes:
  - **Conexão**: relações ponto-a-ponto estabelecidas na camada de transporte. Dentro do escopo do SSL/TLS, toda conexão tem uma sessão.
  - **Sessão**: associação entre cliente e servidor criada a partir de um processo de *handshaking SSL/TLS*. Vários parâmetros de criptografia são definidos em uma sessão, podendo ser usados em várias conexões.

Sessão SSL

- Conexões futuras podem utilizar os mesmos dados de uma sessão já estabelecida;

- Parâmetros:
  - Identificador da sessão.
  - Certificado do par: o certificado X.509 do peer. Pode ser *null*.
  - Método de compressão.
  - Algoritmo de criptografia (ex.: AES).
  - Algoritmo de *hash* (ex.: SHA-1).
  - Senha de 48 bytes compartilhada entre cliente e servidor (segredo mestre).

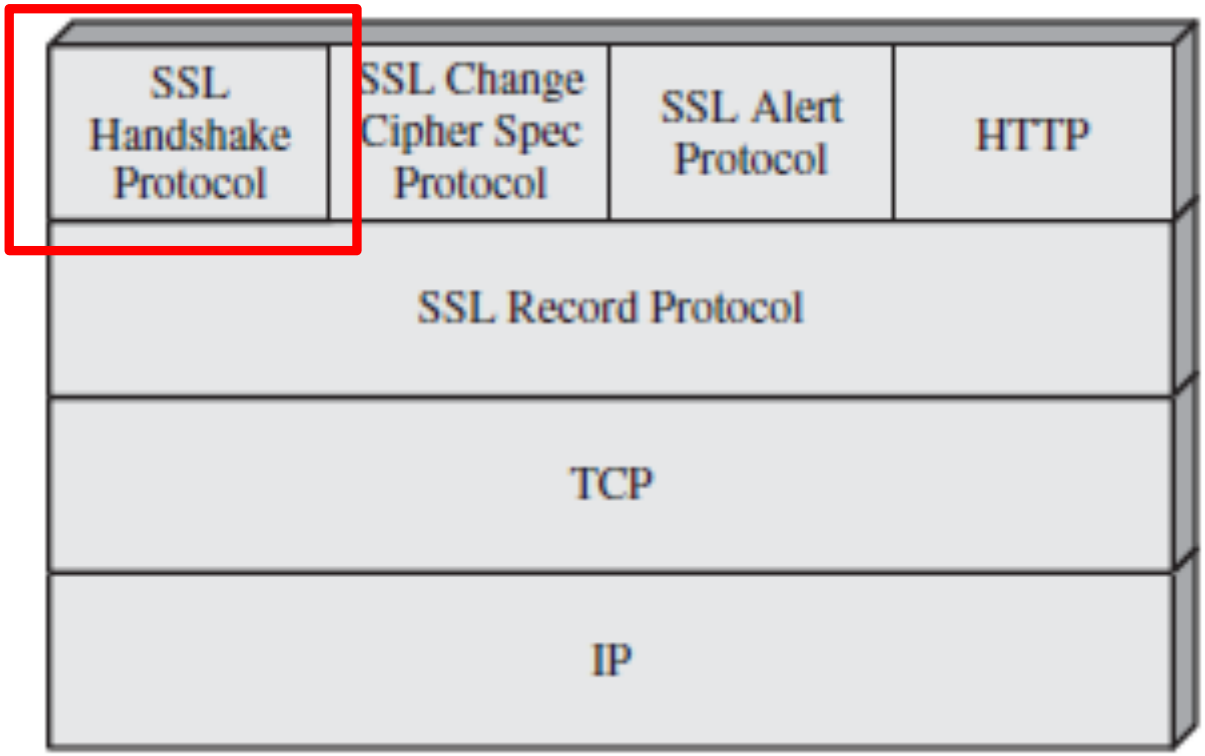
Funcionamento do SSL/TLS

Segurança da Informação– GBC083

Ideia sobre o funcionamento do SSL/TLS

- A seguir veremos o funcionamento do SSLv3;
- Ele se assemelha bastante a padronização TLS;
- Contudo, a recomendação atual é não usar o SSLv3 e seguir as recomendações propostas no TLSv1.0 e superiores.
  - Sempre que possível, usar o TLS v1.3.

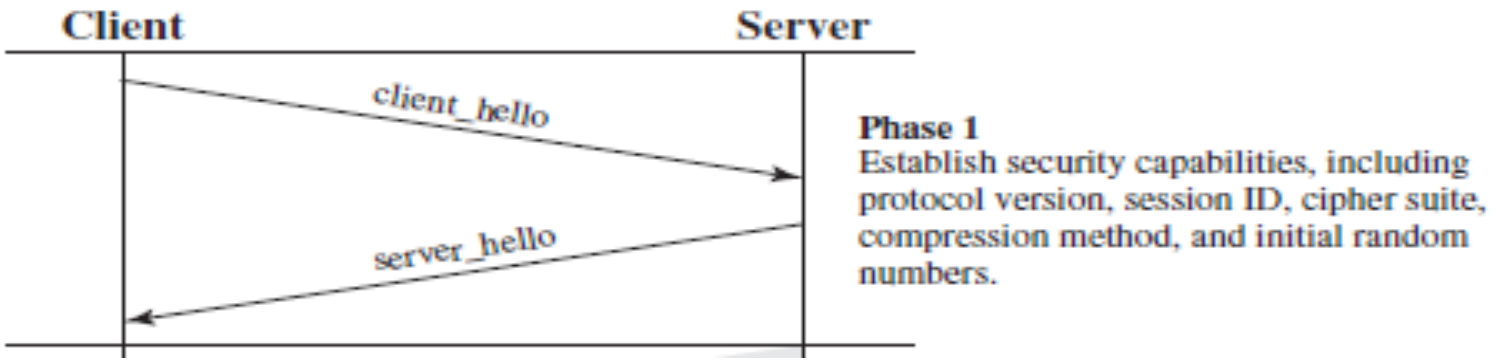
SSL


SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
HTTP
SSL Record Protocol
TCP
IP

SSL: Handshake protocol

- É a parte mais complexa do SSL;
- Envolve uma série de mensagens trocadas entre servidor e cliente para autenticação de ambos e troca de parâmetros de criptografia e verificação de integridade;
- Ocorre antes que qualquer dado da aplicação seja trocado entre cliente e servidor;
- Podemos dividir o handshake em 4 fases.

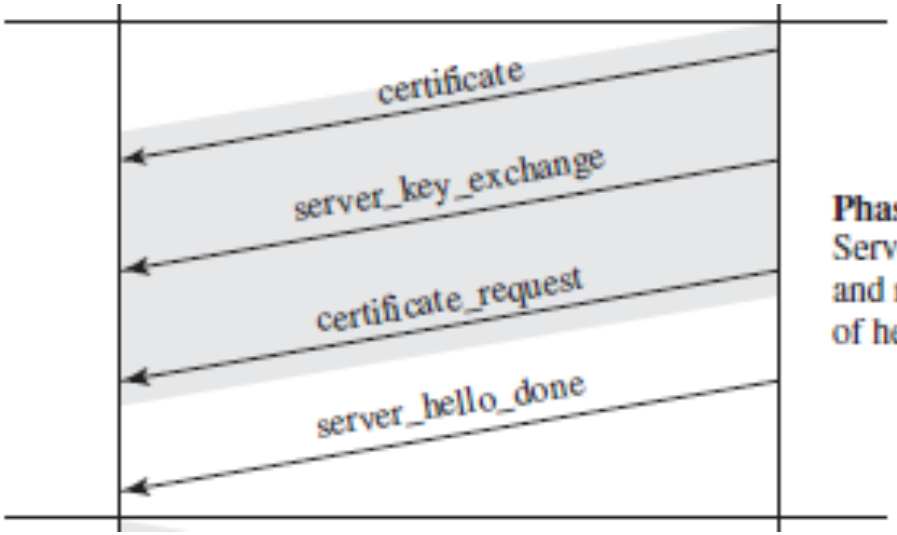
Handshake protocol: fase 1



- **Client_hello:**
  - Versão do protocolo SSL.
  - Random (número aleatório junto de um carimbo de tempo).
  - Id da sessão.
  - Parâmetros de cifragem (troca de chaves, algoritmos de criptografia, hash, tamanho das chaves, tamanho do IV…).
  - Método de compressão.

- **Server_hello:**
  - Responde a requisição do cliente, escolhendo uma opção entre as propostas;
  - Tem os mesmos parâmetros do client_hello.

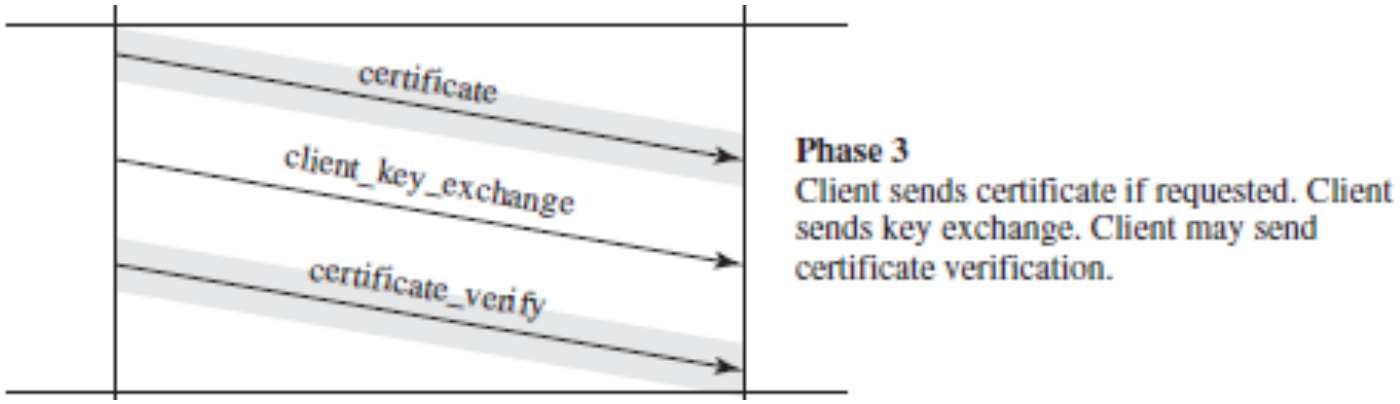
Handshake protocol: fase 2



Phase 2
Server may send certificate, key exchange, and request certificate. Server signals end of hello message phase.

- Autenticação do servidor e troca de chaves.
  1. Servidor envia seu certificado no formato X.509.
  2. Servidor envia mensagem *server_key_exchange* (não é exigida caso a troca de chaves seja feita usando o RSA).
  3. Servidor pode requerer um certificado do cliente (*certificate_request*).
  4. Mensagem *server_done* indica que servidor terminou seu trabalho.

Handshake protocol: fase 3



- Autenticação do cliente e troca de chaves.
  - Cliente verifica se as mensagens enviadas pelo servidor na fase 2 são satisfatórias.
  - Em caso positivo, prossegue com a fase 3.
  - Se o servidor requisitou um certificado, ele é enviado.
  - Cliente envia o seu segredo (*client_key_exchange*).
  - Cliente pode oferecer verificação explícita de seu certificado (*certificate_verify*).

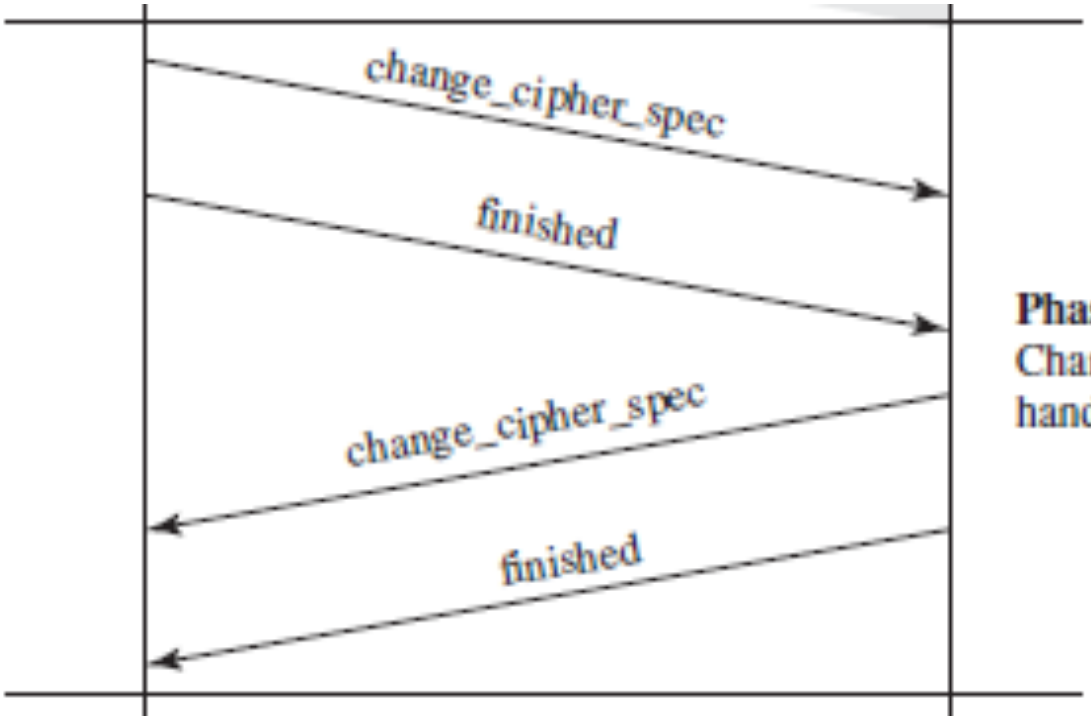
Handshake protocol: fase 3

- *client_key_exchange*
  - o cliente gera um *segredo (pre_master_secret)* de 48 bytes e o cifra com a chave pública do servidor (no caso do RSA)

- *certificate_verify*
  - Envia um código de hash das mensagens trocadas durante o handshake cifrado com a chave privada do cliente

```
CertificateVerify.signature.sha_hash =
   SHA(master_secret || pad_2 || SHA(handshake_messages ||
       master_secret || pad_1));
```

Handshake protocol: fase 4



Phase 4
Change cipher suite and finish handshake protocol.

Handshake protocol: fase 4

- Cliente e servidor confirmam as especificações de criptografia que serão utilizadas e encerram o *handshake*;
- O conteúdo da mensagem "*finished*" é o seguinte valor de hash:

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

- A partir daí, dados de aplicação podem ser trocados de maneira cifrada.

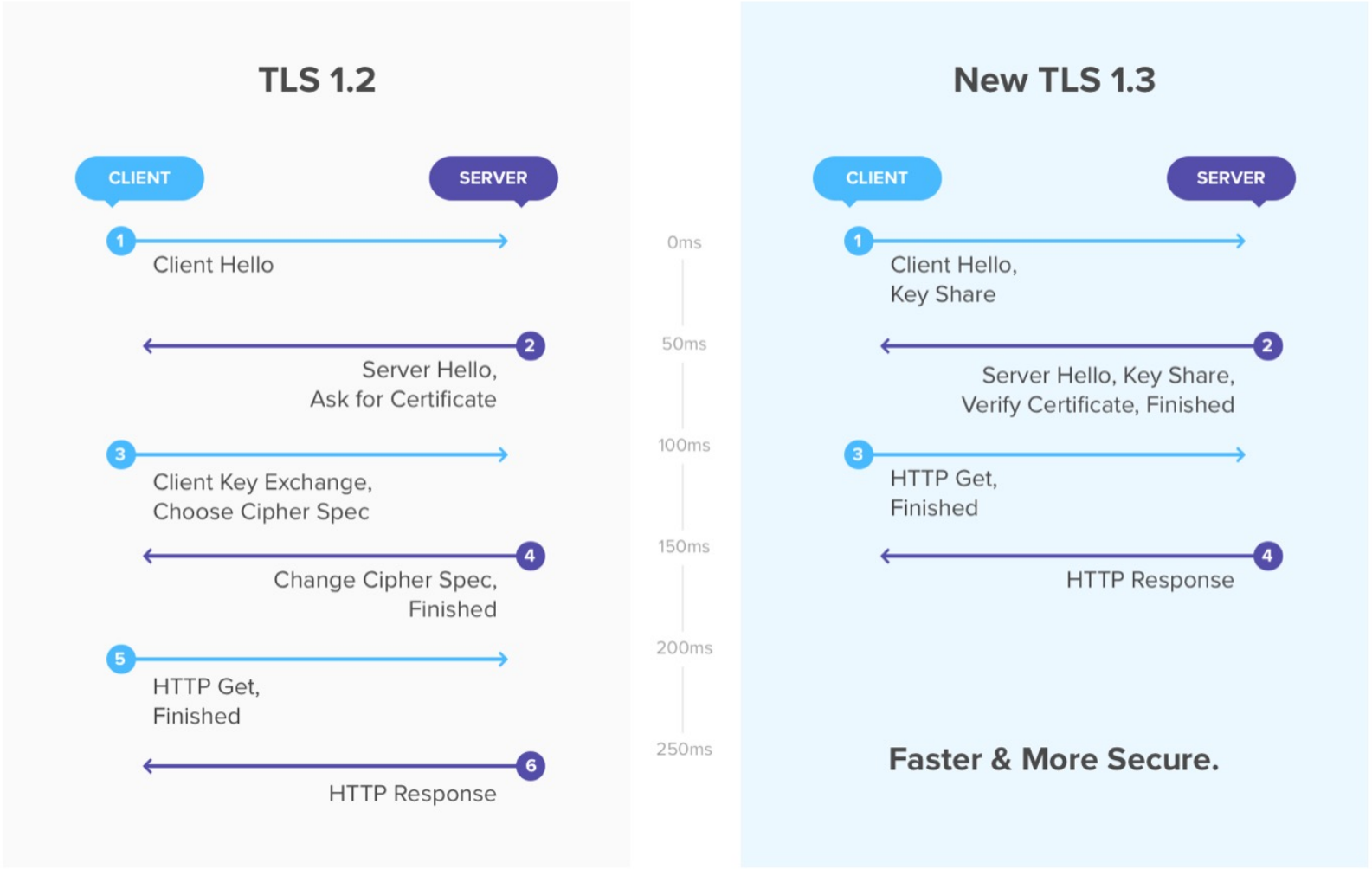
Handshake protocol: fase 4

- Importante: a fase 4 é a primeira onde as mensagens são trafegadas cifradas;

- Ou seja, após o envio de *change_cipher_spec* a mensagem *finished* será enviada cifrada e autenticada (uso da função de hash) com os parâmetros recém criados;

- Caso ambos os lados verifiquem que está tudo certo, a comunicação começa.

Handshake protocol: 1.2 x 1.3



Handshake protocol: chave mestre (segredo compartilhado)

- O segredo mestre compartilhado é um valor de 48 bytes (384 bits) de uso único gerado para esta sessão por meio da troca de chave segura;

- A criação é feita em dois estágios:
  1. Primeiro, um *pre_master_secret* é trocado – isso aconteceu na fase 3 (*client_key_exchange*);
  2. Segundo, o *master_secret* é calculado pelas duas partes.

Handshake protocol: chave mestre (segredo compartilhado)

```
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))
```

Handshake protocol: criação dos valores aleatórios

Ao longo do *handshake*, cliente e servidor usarão os parâmetros trocados para criar os números aleatórios que serão usados como chave dos algoritmos simétricos. Isso é feito da seguinte forma:

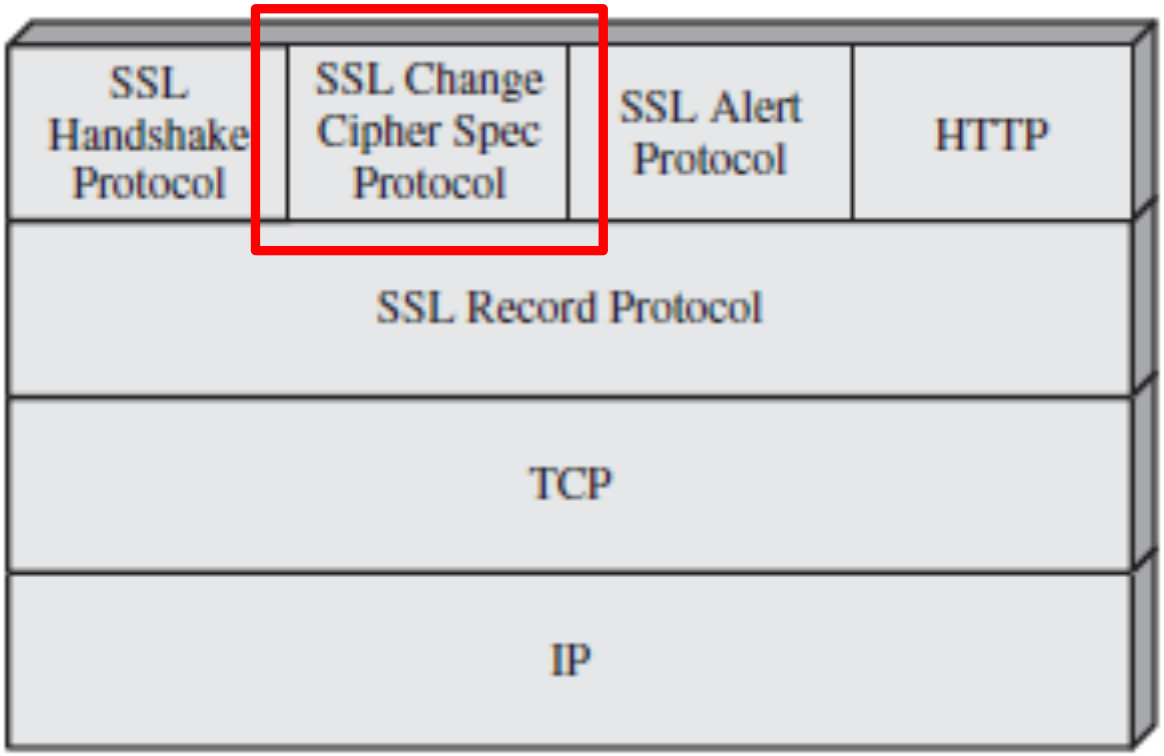
```
key_block = MD5(master_secret || SHA('A' || master_secret ||
                ServerHello.random || ClientHello.random)) ||
            MD5(master_secret || SHA('BB' || master_secret ||
                ServerHello.random || ClientHello.random)) ||
            MD5(master_secret || SHA('CCC' || master_secret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

Handshake protocol: criação dos valores aleatórios

- Seis chaves em sequência são geradas:
  1. 2 chaves para cifrar/decifrar;
  2. 2 valores de MAC (verificação de integridade – serão usados a seguir)
  3. 2 valores para vetores de inicialização

- A função anterior (*key_block*) é executada em *loop* até atingir o tamanho necessário para as seis chaves.

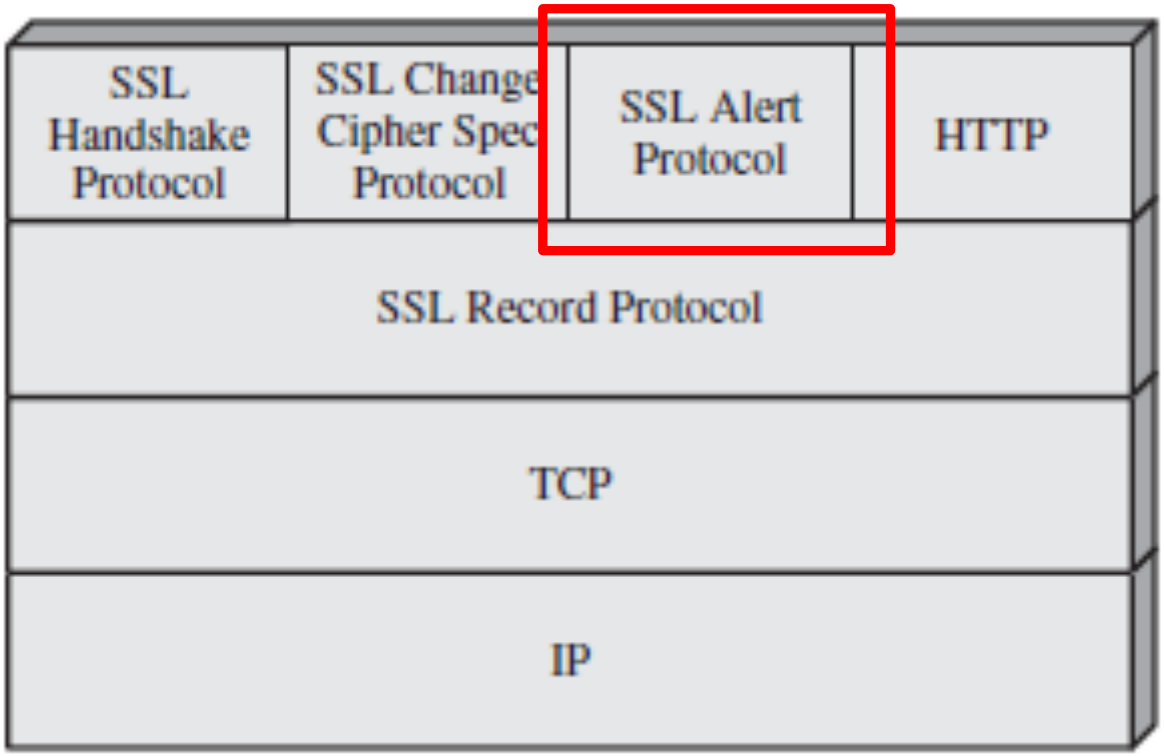
SSL


SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
HTTP
SSL Record Protocol
TCP
IP

Change cipher spec protocol

- Usado para mudar/confirmar parâmetros de criptografia utilizados (fase 4 do handshake);
- Consiste em uma única mensagem;
  - Somente um byte com valor 1.
- Ao receber a mensagem com valor 1, o receptor dela recolhe os parâmetros que estão no estado *pending* e atribui para o estado *current*.

SSL

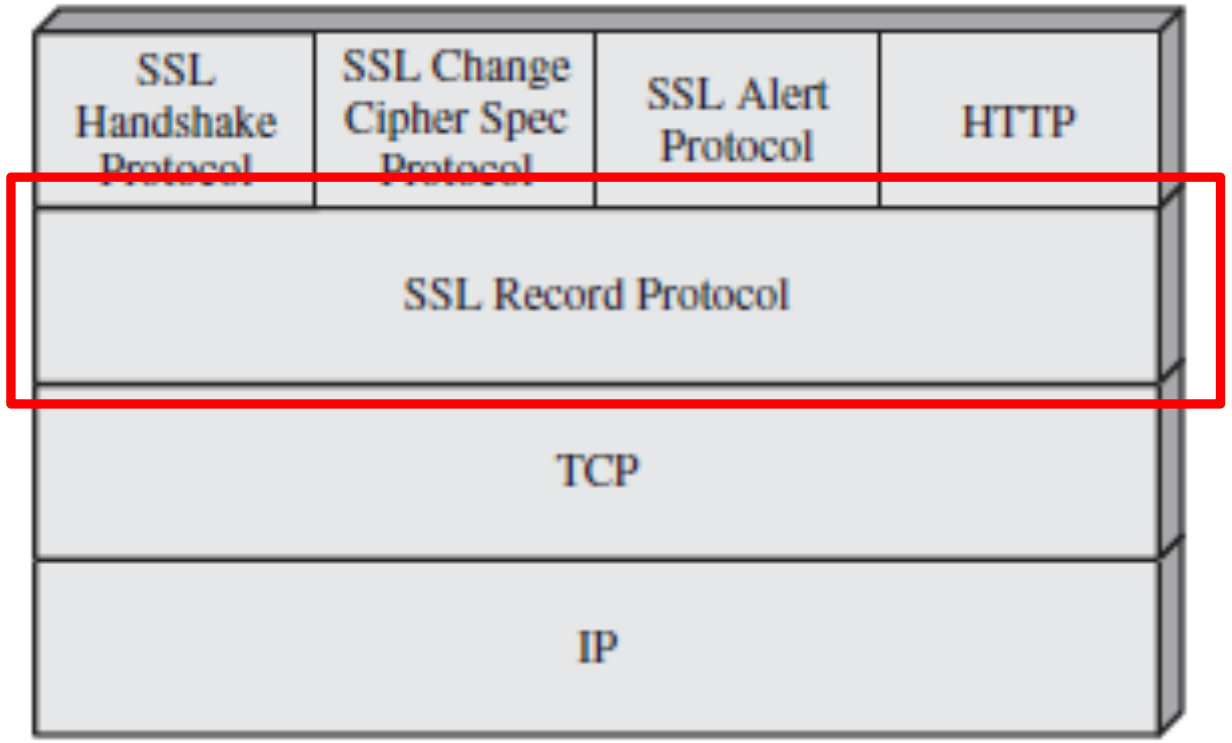

SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
HTTP
SSL Record Protocol
TCP
IP

Alert protocol

- Utilizado para avisar o *host* de situações importantes (advertência ou fatal). Exemplos:
  - Mensagem não esperada.
  - MAC incorreto recebido.
  - Falha no handshake.
  - Etc…

- Dois bytes:
  - Primeiro – advertência (certificado revogado, certificado expirado, notificação de que o emissor não enviará mais mensagens…)
  - Segundo – fatal (problema com o handshake, MAC incorreto..)

SSL

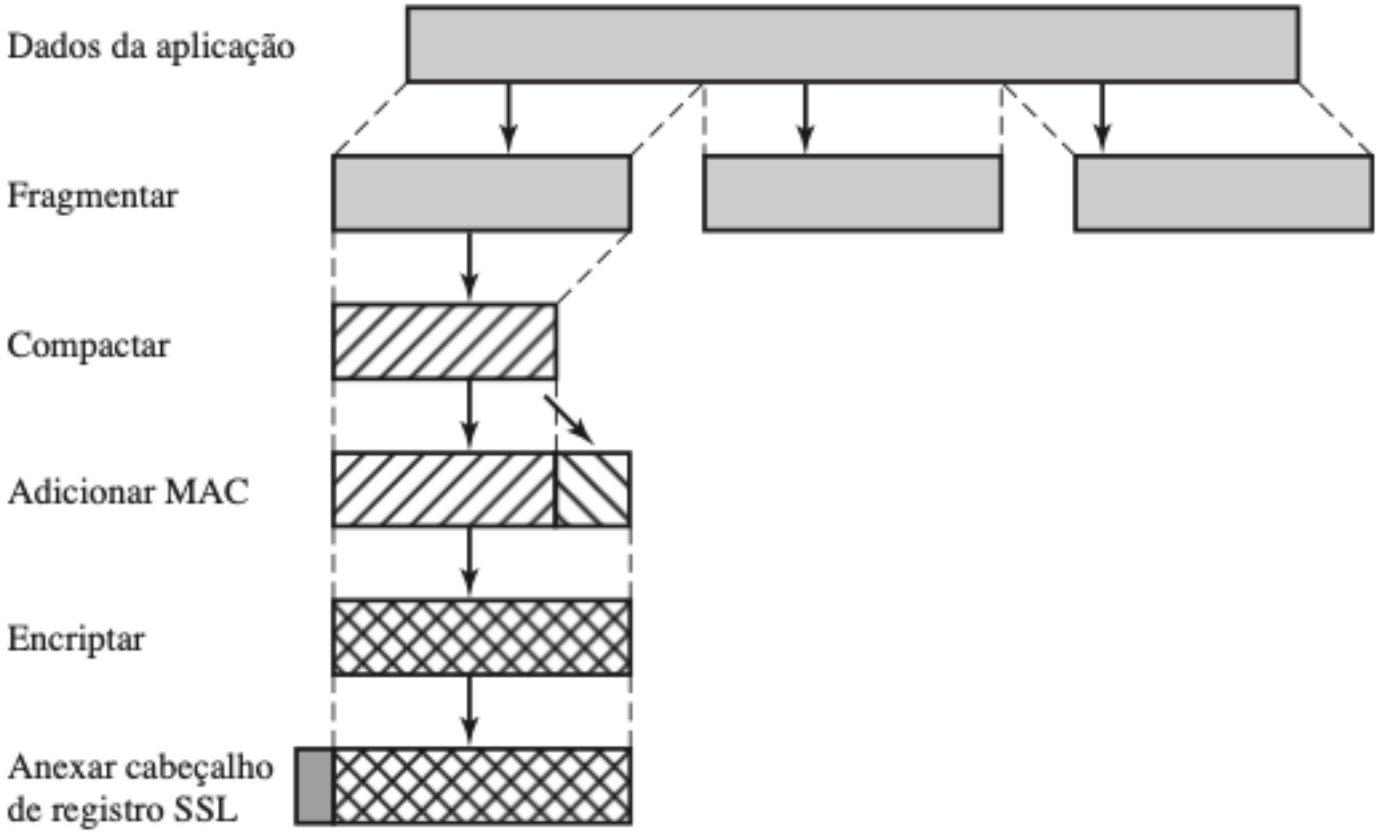

SSL Handshake Protocol
SSL Change Cipher Spec Protocol
SSL Alert Protocol
HTTP
SSL Record Protocol
TCP
IP

Record protocol

- Provê os seguintes serviços:
  - Confidencialidade.
  - Autenticidade/Integridade das mensagens.

- Utiliza informações definidas no processo de *handshake*;

Record protocol



Record protocol – MAC (message authentication code – integridade)

```
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)

Record protocol

- A última etapa do protocolo de registro envolve anexar um cabeçalho com diversos campos como: tipo de conteúdo (HTTP, SMTP, FTP, por exemplo), versão do TLS/SSL e tamanho em bytes do fragmento do texto claro;

- Isso, junto com o MAC_write_secret que foi gerado no *handshake* com o auxílio do cliente e do servidor é o suficiente para o destino checar o MAC.

Conclusões

Segurança da Informação– GBC083

Conclusões

- O protocolo SSL/TLS nada mais é do que uma materialização dos algoritmos criptográficos vistos durante o curso (AES, RSA, funções de hash, MAC, assinaturas e certificados digitais);
- Grande parte do acesso aos sistemas Web são feitos usando o SSL/TLS como uma forma de garantir a confidencialidade, integridade, autenticidade e não-repúdio;
- Órgãos de padronização, empresas e a comunidade acadêmica estão alinhados para o desenvolvimento do SSL/TLS.

Roteiro de estudos

1. Leitura das seções 17.1, 17.2 e 17.3. do livro "Criptografia e segurança de redes. Princípios e práticas". William Stallings;
2. Estudo da vídeo-aula referente ao tópico 14;
3. https://cabulous.medium.com/tls-1-2-andtls-1-3-handshake-walkthrough-4cfd0a798164
4. Resolução do TP-6.