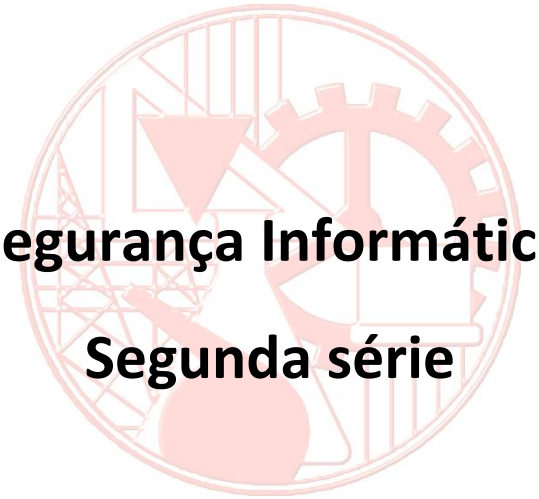
Instituto **S**uperior de **E**ngenharia de **L**isboa

Licenciatura em Engenharia Informática e de Computadores

Semestre de Inverno 2016/2017



Segurança Informática

Segunda série

LI51N

Trabalho elaborado pelo Grupo 1:

Sara Sobral N.º 40602

Eduardo António N.º 40686

1. No contexto do protocolo TLS:

1.1. Quando o *handshake* é baseado na primitiva RSA, de que forma é protegido o envio do *pre-master secret* entre o cliente e o servidor? Esta proteção garante a sua confidencialidade e integridade?

ClientHello	C → S: client capabilities
ServerHello	C ← S: parameter definitions
Certificate	C ← S: server certificate (**KeS**)
CertificateRequest(*)	C ← S: Trusted CAs
ServerHelloDone	C ← S: synchronization
Certificate(*)	C → S: client certificate (**KvC**)
ClientKeyExchange	C → S: **Enc(KeS: pre_master_secret)**
CertificateVerify(*)	C → S: **Sign(KsC: handshake_messages)**
ChangeCipherSpec	C → S: record protocol parameters change
Finished	C → S: **{PRF(master_secret, handshake_messages)}**
ChangeCipherSpec	C ← S: record protocol parameters change
Finished	C ← S: **{PRF(master_secret, handshake_messages)}**

Depois do envio do certificado pelo cliente para o servidor é estabelecido o *premaster secret (ClientKeyExchange).* Usando o RSA, o *premaster secret* é encriptado com a chave publica do servidor.

Todas as mensagens *handshake* (excepto *ClientKeyExchange*) são enviadas desprotegidas pelo *record protocol* (*null cipher suite*). O Record Protocol fornece autenticação dos dados e integridade e confidencialidade.

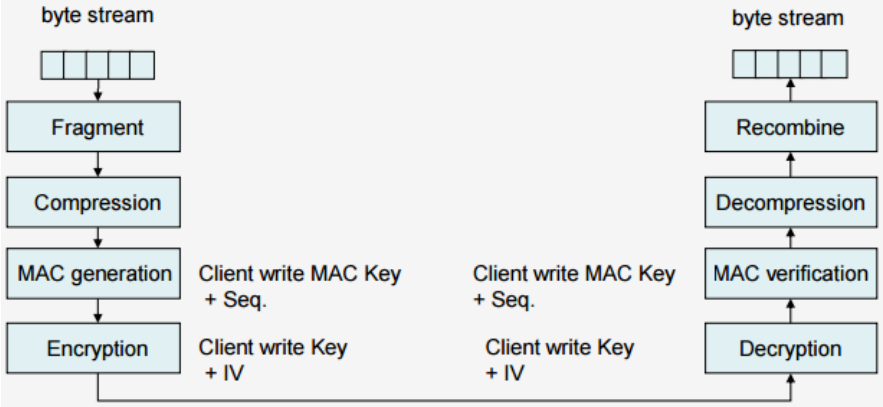
1.2. Qual a importância da propriedade *perfect forward secrecy*? O estabelecimento de chaves usando cifra assimétrica fornece esta propriedade?

Perfect Forward Secrecy é uma propriedade que garante que uma comunicação é confidencial, e espera-se que esta se mantenha confidencial no futuro mesmo que as chaves do certificado do servidor ou o certificado ou usuário/senha do cliente) sejam comprometidas. Alternativamente, se uma única comunicação for comprometida (a chave de sessão foi comprometida) isso não compromete a confidencialidade de todas as comunicações anteriores a ela. O estabelecimento de chaves usando cifra assimétrica fornece esta propriedade.

1.3. Quantas chaves usa o record protocol? Porque motivo são usados conjuntos de chaves diferentes em cada sentido da comunicação?

As chaves para a criptografia simétrica são geradas exclusivamente para cada conexão e baseiam-se num segredo compartilhado negociado no início da sessão.

O record protocol usa dois pares de chaves MAC key e Encryption Key. Um dos pares é usado pelo cliente e outro pelo servidor.



São utilizados conjuntos de chaves diferentes em cada sentido da comunicação para evitar injeções de massagens, pois como o cliente usa chaves diferentes das do servidor as mensagens injetadas serão identificadas como falsas.

2. Considere uma aplicação web que autentica os seus utilizadores com base em passwords.

2.1. Admita que foi utilizado um salt de 4 bits para produzir a informação de validação. Se o atacante quiser pré-calcular o dicionário para este cenário, quantos cálculos terão de ser feitos para cada entrada do dicionário?

Para cada entrada do dicionário terão de ser feitos 16 cálculos dado o salt ser de 4 bits logo 2^4 = 16

2.2. Considere que a aplicação web mantem o estado de autenticação num cookie. Qual destes dois esquemas deve ser sempre usado: esquema de cifra simétrico ou esquema MAC?

MAC.

3. Explique duas diferenças entre os esquema de autenticação HTTP Basic [1] e Digest [2]. Estes esquemas de autenticação garantem a confidencialidade da password?

O esquema de autenticação HTTP Basic usa codificação base64 não criptografada. Só deve ser usada onde a segurança da camada de transporte é fornecida, como https porque este esquema envia a passwors em clarod

O esquema de autenticação Digest comunica as credenciais em um formulário criptografado aplicando uma função de hash ao nome de usuário, à senha, a um valor de nonce fornecido pelo servidor, ao método HTTP e ao URI solicitado. Não oferece nenhuma proteção de confidencialidade além de proteger a atual password.

4. No contexto da framework de autorização OAuth 2.0 :

4.1. No fluxo authorization code, durante um pedido de autorização (authorization request), se o dono de recursos já estiver autenticado no cliente e no servidor de autorização, o formulário de consentimento é apresentado?

Não. Pois a autenticação é feita em duas fases apresentação das credenciais pelo utilizador (ex.: "username+password"), obtenção dum autenticador e apresentação do autenticador automaticamente pelo "user-agent". Como estas fases já foram realizadas sabemos que os autenticadores persistem entre sessões

4.2. Qual a vantagem do fluxo authorization code em comparação com o fluxo implícito?

O fluxo authorization code é usado para obter tokens de acesso e atualizar tokens e é otimizado para clientes confidenciais. Como este é um fluxo baseado em redireccionamento, o cliente deve ser capaz de interagir com o user-agent do proprietário do recurso e capaz de receber solicitações de entrada (via redireccionamento) do servidor de autorização. (ex.: logins de aplicações web como o Facebook ou o Google).

O fluxo implícito é semelhante ao fluxo authorization code.. É usado para obter tokens de acesso (não suporta a emissão de tokens de atualização) e é otimizado para clientes públicos conhecidos por operar um URI de redireccionamento específico. Como é um fluxo baseado em redireccionamento, o cliente deve ser capaz de interagir com o user-agent do proprietário do recurso (geralmente um navegador da Web) e capaz de receber solicitações de entrada (via redireccionamento) do servidor de autorização. Não pode manter um segredo cliente porque todo o código e armazenamento da aplicação web é facilmente acessível.

Ao contrário do fluxo authorization code, no qual o cliente faz pedidos de autorização separados e para um token de acesso, o cliente recebe o token de acesso como resultado do pedido de autorização. O fluxo implícito não inclui a autenticação do cliente e depende da presença do proprietário do recurso e do registro do URI de redireccionamento. Como o token de acesso é codificado no URI de redireccionamento, ele pode ser exposto ao proprietário do recurso e a outros aplicativos que residem no mesmo dispositivo.

5. No contexto do fluxo authorization code do protocolo OpenID Connect:

5.1. Para que serve o ID Token?

O ID token é um token de segurança que contém afirmações sobre a autenticação de um usuário final por um servidor de autorização ao usar um cliente e, potencialmente, outras reivindicações solicitadas. O ID token é representado como um **JSON Web token (JWT)**.

5.2. Qual destas duas entidades desempenha o papel de relying party: a aplicação cliente ou o resource server?

Cliente.

6. Adicione ao programa desenvolvido na primeira serie de exercícios a possibilidade de usar uma chave derivada de uma password para transportar a chave que cifra a mensagem. Utilize a norma PKCS#5 [4] para derivar a chave a partir da password, tal como descrito na Secção 4.8 do RFC 7518 [5].

Classe: KeyFromPassword.java

Classe de teste: KeyFromPasswordTest.java

7. Realize uma aplicação Web com as seguintes funcionalidades:

_ Os utilizadores são autenticados através do fornecedor de identidade social Google, usando o protocolo OpenID Connect [6].

Id do cliente:

311695581632-ioq4ip2mn23ltf8jljoibjtgpk0td7pj.apps.googleusercontent.com

chave secreta do cliente:

-GUbXQ4KJdTZARojIVXK6IRo

_ Os utilizadores autenticados podem consultar a listagem de issues de um projeto GitHub [7].

_ Os utilizadores autenticados podem criar uma task Google a partir de issues do GitHub [8].

Incompleto.

1. O cliente redirecionará o usuário para o servidor de autorização com os seguintes parâmetros:
    a. response_type com o valor code
    b. client_id com o identificador do cliente
    c. redirect_uri, uri para o qual será redirecionada a pagina apos validação (opcional)
    d. scope
    e. state com o CSRF token, deve-se armazenar o valor do token CSRF na sessão do usuário a ser validado quando retorna (opcional)
2. Todos os parâmetros serão validados pelo servidor de autorização.
3. Se o usuário aprova o cliente, é redirecionado do servidor de autorização para o URI de redireccionamento.
4. O cliente irá agora enviar um pedido POST para o servidor de autorização com os seguintes parâmetros:
    a. grant_type com o valor de authorization_code
    b. client_id
    c. client_secret com o segredo do cliente
    d. redirect_uri
    e. code com o código de autorização da query string
5. O servidor de autorização irá responder com um objeto JSON contendo as seguintes propriedades:
        a. token_type este será geralmente o termo "portador"
        b. expires_in com um inteiro representando o TTLdo token de acesso (ou seja, quando o token expira)
        c. access_token o token de acesso em si
        d. refresh_token um token de atualização que pode ser usado para adquirir um novo token de acesso quando o original expira
6. Token de autorização (code) é entregue à aplicação cliente por redirecção do user-agent.