1. (a) Porque motivo a propriedade perfect forward secrecy não é garantida usando o processo base com RSA para estabelecimento do master_secret?

No processo base com RSA, ao ser estabelecido o master_secret, o cliente gera um número aleatório de sessão, pre_master_secret, e envia ao servidor. É neste momento que o servidor utiliza a sua chave privada para encriptar a pre_master_secret, utilizando o algoritmo RSA. Posteriormente o servidor, envia o resultado da encriptação, master_secret, ao cliente, e consequentemente, o cliente utiliza a sua chave privada para decifrar o master_secret.
O que é colocado em causa, é o facto de no processo base com RSA, o master_secret é derivado da chave pública do servidor. Caso a chave pública do servidor, seja comprometida, então qualquer um poderá decifrar todas as comunicações anteriores que foram encriptadas utilizando esse master_secret.
Desta forma caso o intuito seja preservar a propriedade, perfect forward secrecy, PFS, o processo em que se estabelece o master_secret, deve ser utilizado um segredo que seja compartilhado, mas que não seja derivado da chave pública do servidor.

(b) Identifique dois possíveis ataques ao record protocol e explique as técnicas usadas para os prevenir.

Dois possíveis ataques ao record protocol podem ser: um ataque man-in-the-middle (MITM), onde o atacante intercepta as comunicações entre o cliente e o servidor e assume o controlo da sessão. Desta forma o atacante pode ler, modificar ou até interromper as comunicações. De modo a prevenir este tipo de ataques, o record protocol, utiliza o protocolo TLS, que fornece autenticação e criptografia ponta a ponta. A autenticação garante que apenas o cliente e o servidor autorizados possam participar na sessão, enquanto a criptografia garante que as comunicações sejam incompreensíveis a terceiros (atacantes). O segundo, poderá ser um ataque de replay, onde o atacante captura uma mensagem encriptada e retransmite-a posteriormente. Desta forma o atacante pode tentar usar a mensagem para engar o cliente ou o servidor. O modo que o record protocol, utiliza para prevenir este tipo de ataques é a utilização de um campo de sequência nos cabeçalhos do protocolo. Este campo identifica a sequência de cada pacote de forma única. Assim este campo é utilizado pelo cliente ou pelo servidor, para verificar se uma mensagem já foi recebida.

2. Considere uma aplicação web que usa como informação de validação das passwords a forma $h_u = H(pwd_u||salt_u)$, sendo H uma função de hash, pwd_u a password do utilizador u e $salt_u$ um número aleatório gerado no momento do registo do utilizador u (|| representa a concatenação de bits)

O uso da técnica conhecida como CAPTCHA (Completely Automated Public Turing Test to Tell Compu- ters and Humans Apart) contribui para mitigar ataques de dicionário à informação de validação? Explique.

Sim, o uso de CAPTCHA contribui para mitigar ataques de dicionário à informação de validação. Os ataques de dicionário são uma forma de brute force attack, que usa uma lista de palavras ou frases comuns para adivinhar passwords. Os CAPTCHAs são uma técnica que pode ser usada para dificultar o uso de ataques de dicionário, pois exigem que o utilizador interaja com a aplicação de uma forma que o computador não possa replicar. No caso da aplicação descrita, web, o uso de CAPTCHA pode ajudar a mitigar ataques de dicionário de duas formas: a primeira, em dificultar o uso de bots para tentar adivinhar as passwords, isto é, impedir a utilização de programas automatizados para o preenchimento de formulários com intuito de adivinhar passwords. O segundo, em ajudar identificar os utilizadores humanos que estão a tentar adivinhar passwords, caso um utilizador falhe repetitivamente um CAPTCHA, o mais provável é que este, esteja a tentar adivinhar uma password. A aplicação web pode então bloquear o utilizador ou então, exigir que este forneça informações adicionais para provar que é humano.

> Considere uma aplicação web que mantém estado de autenticação entre o browser e a aplicação servidor usando cookies. No cookie é guardado um JSON web token (JWT) com o identificador do utilizador. Como é que a aplicação servidor pode detetar se o conteúdo do cookie foi adulterado no browser?

A aplicação servidor pode detetar se o conteúdo do cookie foi adulterado no browser através de um ou mais das seguintes técnicas: Verificar o formato do token o token JWT, deve estar no formato correto, conforme definido na especificação JWT. A aplicação servidor pode verificar o formato do token utilizando uma função de validação JWT. Verificar a assinatura do token, o token JWT, deve ser assinado utilizando uma chave secreta. A aplicação servidor pode verificar a assinatura do token, utilizando essa chave secreta. Por fim verificar o tempo de validade do token, o token JWT deve ter um tempo de validade. A aplicação servidor pode verificar o tempo de validade do token para verificar se este ainda se encontra válido.

3. Considere a norma OAuth 2.0 e OpenID Connect no fluxo authorization code grant:
    1. (a) Em que situação está previsto o uso da estrutura JWT e com que objetivo?

A situação de usu da estrutura JWT, é utilizada na OpenID Connect para representar tokens de identidade. Quando um cliente solicita tokens de acesso e refresh no fluxo Authorization Code Grant do Open ID Connect, os tokens retornados podem ser formatados como JWTs. O objectivo principal do uso de JWTs é permitir que as informações de identidade sejam compactadas num formato autocontido e assinado digitalmente. Isto significa que o cliente pode confiar nas informações do token sem precisar consultar o Entity Provider em qualquer momento. Para além disto, o uso de JWTs facilita a interoperabilidade entre diferentes sistemas, pois o formato é padronizado e facilmente suportado. O token de ID gerado durante o fluxo de Authorization Code Grant, contém informações sobre o utilizador autenticado, como o ID, scopes concedidos e outro tipo de infomações do perfil. Em suma permite que o cliente obtenha informações sobre o utilizador de maneira segura e eficiente.

2. (b) No OAuth 2.0, após o dono de recursos ter autorizado e consentido o uso de um recurso, descreva as ações da aplicação cliente para conseguir fazer os pedidos ao servidor de recursos.

No fluxo Authorization Code Grant do OAuth 2.0, após o proprietário do recurso (tulizador) ter autorizado e consentido o acesso, a aplicação cliente precisa de seguir algumas etapas para obter e utilizar os *tokens* de acesso necessários para fazer pedidos ao servidor de recursos, as principais ações são:

1- Redireccionamento para a Autorização:
Após obter a autorização do utilizador, a aplicação cliente redirecciona o utilizador para o servidor de autorização com uma solicitação de código de autorização.
A solicitação inclui informações como o identificador do cliente, *scopes* necessários e um *URI* de redirecionamento.

2- Obtenção do Código de Autorização:
O utilizador autentica-se no servidor de autorização, e este, se a autorização for concedida, retorna um código de autorização para o *URI* de redirecionamento especificado pela aplicação cliente.

3- Troca do Código de Autorização por *tokens* de Acesso:
A aplicação cliente, agora com o código de autorização, faz uma solicitação direta ao servidor de autorização para trocar o código por *tokens* de acesso.
A solicitação inclui o código de autorização, o identificador do cliente, o segredo do cliente (se aplicável) e o *URI* de redirecionamento.

4- Recepção dos *tokens* de Acesso:
O servidor de autorização valida a solicitação e, se tudo estiver correto, retorna um conjunto de *tokens* de acesso, que geralmente inclui um *token* de acesso, um *token* de atualização e informações adicionais.

5- Utilização do *Token* de Acesso para aceder aos recursos:
Com o *token* de acesso, a aplicação cliente pode agora fazer solicitações ao servidor de recursos em nome do utilizador.
O *token* de acesso é incluído nas solicitações como um cabeçalho *HTTP* ou num outro método de autenticação suportado.