Q1

1. Sim. Para se comunicar com confidência, basta saber a chave pública de Alice.
2. Sim. A assinatura digital é vista com a chave pública. O processo é o seguinte: Bob utiliza sua chave privada para cifrar a mensagem e enviar para Alice. Essa mensagem será decifrada com a chave pública, também de Bob. Então, como Mallory possui a chave pública de Bob, ele(a) pode verificar a assinatura digital.
3. Alice deve encriptar sua mensagem utilizando a chave privada dela e depois a chave pública de Bob.

Q2

A mensagem passa por um algoritmo simétrico depois é feito o controle de erros.

Q3

O hash serve para garantir a integridade da mensagem. O código de autenticação garante a autenticidade.

Q4

Assinatura depois confidencialidade. A assinatura digital é cifrada pela chave pública. A confidencialidade é cifrada pela chave privada.

Q5

Sim e não. Sim, pois só é possível decifrar a mensagem usando a chave. Não, pois a chave pode ser obtida através de força bruta, apesar de ser computacionalmente inviável.

Q6

1. SHA-2 E SHA-3
   - Características:
     - SHA-2 possui diferentes tamanhos de hash. 224, 256, 384 e 512 bits.
     - Única modificação na entrada gera uma saída completamente diferente. Mas a mesma entrada sempre gerará a mesma saída.

   - Modo de funcionamento:
     1. Antes da função hash, é calculado 8 hashs iniciais. Cada hash é calculado através da função toHex(first32BitOf(sqrt(n_{primo}))), usando os 8 primeiros números primos.

        Ex.:
        h0 = toHex(first32BitOf(sqrt(2))). 2 é o primeiro número primo.
        h1 = toHex(first32BitOf(sqrt(3))). 3 é o segundo número primo.
        h2 = toHex(first32BitOf(sqrt(5))). 5 é o terceiro número primo.
        ...

        Como sugere as funções, é calculado a raiz quadrada do número primo. Depois extrai-se os primeiros 32 bits do resultado dessa raiz. Depois esse binário é convertido para hexadecimal.

2. Depois é calculado ‘n’(64 para SHA-224 e SHA-256, 80 para SHA-384 e SHA-512 e 24 para qualquer divisão do SHA3) constantes hash obtidas através da função toHex(first32BitOfFractionPart(cbrt(n_{primo}))), usando os primeiros ‘n’ números primos.

   Como sugere as funções, é calculado a raiz cúbica do número primo. Depois extrai-se os primeiros 32 bits da parte fracional do resultado dessa raiz. Depois esse binário é convertido para hexadecimal.

3. A partir de então é feita operações sobre a entrada. Primeiro, a entrada é convertida para binário e preenchida até alcançar ‘n’ bits (512 bits para SHA-224 e SHA-256, 1024 para SHA-384 e SHA-512, 1152 para SHA3-224, 1088 para SHA3-256, 832 para SHA3-384 e 576 para SHA3-512). Esse preenchimento é feito, primeiramente, adicionando o bit ‘1’ à direita da cadeia binária. Após isso adiciona ‘0’ até sobrar o valor binário da entrada. Por exemplo, se a entrada for “abc”, a versão binária terá 24 bits. 24 em binário é 11000. 11000 serão os últimos bits do preenchimento. Entre 11000 e a versão binária da entrada, ficarão vários ‘0’.

4. Esse preenchimento é dividido em pedaços de 512 bits. Esses pedaços são divididos em pedaços de 32 bits. Ou seja, 16 grupos (words) de 32 bits para cada pedação de 512 bits.
5. Os hashs descobertos na etapa 1 são combinados com os pedaços de 32 bits e às hashs constantes da etapa 2. Como existem somente 16 grupos (words) e ‘n’ constantes hashs (em todas as versões são maiores que 16) é feita uma operação para completar os hashs que faltam.

   $W[x] = W[x-16] + \sigma_0(W[x-15]) + W[x-7]) + \sigma_1(W[x-2])$

   sendo que:
   $\sigma_0 = S^7(x) \text{ xor } S^{18}(x) \text{ xor } R^3(x)$ e $\sigma_1 = S^{17}(x) \text{ xor } S^{19}(x) \text{ xor } R^{10}(x)$
   onde:
   S = rotação para direita R = deslocamento para direita lógico

$t1 = Ch(h4,h5,h6) + W[x] + k[x] + \sigma_1(h4) + h7$
onde hn são os hashs da etapa 1, k é o hash da etapa 2 e
$Ch(x,y,z) = (x \wedge y) \text{ xor } (\sim x \wedge z)$
$t2 = Ma(h0,h1,h2) + \sigma_0(h0)$ onde Ma é $Ma(x,y,z) = (x \wedge y) \text{ xor } (x \wedge z) \text{ xor } (y \wedge z)$

novoh0 = t1 + t2
novoh4 = h3 + t1
novoh1 = h0
novoh2 = h1
novoh3 = h2
novoh5 = h4
novoh6 = h5
novoh7 = h6

h0 += novoh0
h1 += novoh1
h2 += novoh2
h3 += novoh3
h4 += novoh4

h5 += novoh5
h6 += novoh6
h7 += novoh7

Saída = h0 + h1 + h2+ h3 + h4 + h5 + h6 + h7

Fontes:https://youtu.be/PMOEdd4yzyU
https://en.wikipedia.org/wiki/SHA-2

- Vantagens: Rápido de calcular, resistente à colisão
- Desvantagens: Não é eficaz em embaralhar senhas, SHA-3 não é amplamente utilizado.

Fontes:
https://crypto.stackexchange.com/questions/43990/what-are-advantages-and-disadvantages-of-sha-256
https://crypto.stackexchange.com/questions/26064/advantages-disadvantages-of-bcrypt-vs-hash-salt

2. CMAC E HMAC
   - Características: HMAC faz o hash da chave e da entrada e concatena-as. Verifica se a entrada e a chave secreta são iguais após o cálculo da assinatura HMAC. CMAC encripta a entrada usando a chave e aplicando o algoritmo AES e calcula o MAC através da concatenação da chave. Para a verificação, a assinatura é comparada com o novo CMAC da entrada com a chave. CMAC é MAC com AES, usando blocos de cifras. HMAC é MAC usando hash. Ambos servem para verificar a integridade da mensagem.

     Fontes:
     https://crypto.stackexchange.com/questions/31898/difference-between-aes-cmac-and-aes-hmac
     https://crypto.stackexchange.com/questions/15721/use-cases-for-cmac-vs-hmac

   - Modo de funcionamento: HMAC
     1. Adiciona zeros à esquerda da chave secreta (binária) até alcançar ‘n’ bits.
     2. xor entre o resultado da etapa 1 com o binário 00110110 (36 em hexadecimal) repetido “n/8” vezes, para coincidir com o tamanho da chave secreta.
     3. Unir o resultado do xor com a entrada. A entrada vai à direita do resultado da etapa 2.
     4. Aplicar o hash no resultado da etapa 3.
     5. xor entre o resultado da etapa 1 com o binário 01011100 (5C em hexadecimal) repetido “n/8” vezes, para coincidir com o tamanho da chave secreta.
     6. Unir o resultado do xor com a entrada. A entrada vai à direita do resultado da etapa 5.
     7. Unir o resultado da etapa 4 à direita do resultado da etapa 6.
     8. Aplicar o hash no resultado da etapa 7.

   - Modo de funcionamento: CMAC
     1. Quando a entrada for um inteiro múltiplo ‘n’ do bloco de cifra ‘b’. Para AES, b = 128 e para DES (Data Encryption Standard) triplo, b = 64. A entrada será dividida em ‘n’ blocos (M1, M2, …, Mn).

2. Encripta usando AES, entre a chave ‘K’ e parte da entrada “M1”.
3. Encripta usando AES, entre a chave ‘K’ e parte da entrada com a operação xor da etapa 2 (“M2” xor etapa2). Repete-se para todas as partes da entrada restantes.

   Fonte: https://youtu.be/PH61OkF6Xts

- Vantagens:
  - CMAC:
    - Pode ser calculado sem consumo extra de CPU.
    - No mínimo 1 bloco de cifra bem criptografado garante a confidencialidade da entrada.
  - HMAC:
    - Quando o algoritmo de hash é seguro, garante que, mesmo com a chave, não seja possível produzir duas entradas diferentes.
    - Com duas chaves diferentes não é possível produzir um valor HMAC válido.
- Desvantagens:
  - HMAC:
    - Segurança não pode ser comprovada baseado no com base nas propriedades usuais em relação às colisões do hash subjacente.

Fonte: https://crypto.stackexchange.com/questions/18639/block-cipher-based-vs-hash-based-mac

Q7

EECS

Q8

Ela deve usar a chave pública do AC X para verificar a autenticidade do certificado de Bob. Se ela não possui o certificado da AC X, ela deve instalá-lo (comumente não é necessário, pois eles são pré-instalados pelos SOs ou browsers).

Q9

a) CN (Nome completo): VeriSign Universal Root Certification Authority

b)
- Tipo: Assimétrico.
- Algoritmo da chave: RSA
- Tamanho da chave: 2048

c)
- Algoritmo da assinatura: 1.2.840.113549.1.1.11
- Tamanho: 2048

d)
1. São geradas as chaves pública e privada.
2. A chave pública é enviada à Autoridade de Registro (AR), junto com os dados cadastrais.
3. AR envia os dados comprovados para a Autoridade Certificadora (AC).
4. AC emite o certificado para o diretório público. O certificado foi criado.

e) Obtém-se a mensagem do certificado e é usado o mesmo algoritmo hash. Se os dois hashes coincidirem, então o certificado é válido.

Q10

Ela deve instalar o certificado dele e comparar o hash.