Name: Mary Precious

1) Scenario: Alice wants to send Bob a long message, and she doesn't want Eve to be able to read it. Assume for this scenario that AITM is impossible.

    Solution:

Alice and Bob use Diffie-Helman to agree on a shared key, k. Using AES–256, Alice encrypts the message, M using AES(k, M) to produce a cipher text C, and sends it to Bob who decrypts it using AES_D(k, C).

    Why it works:

AES-256 is suitable for large data communication, as it is fast and has a strong security. However, both parties need to agree on the shared key. The key can't be sent on the internet, as it could be intercepted by Eve. By using the Diffie-Helman, they can share the key securely. The key is never really sent, but derived mathematically, based on secret numbers that Alice and Bob know. Since we assume that these secret numbers are kept securely by each party, Eve does not have access to it. Without the key, the encrypted message will just be gibberish to Eve. Eve cannot brute force the decryption of the message either as it will take an enormous amount of time to be able to pull that through, which is not computationally feasible.

2) Scenario: Alice wants to send Bob a long message. She doesn't want Mal to be able to modify the message without Bob detecting the change.

    Solution:

Alice and Bob both choose a secret key S_A and S_B respectively that they keep private. They also choose a public key P_A and P_B that they share with each other. Alice computes the hash of the message using H(M), and encrypts that using her secret key as E(S_A, H(M)). She then sends (M, E(S_A, H(M))) to Bob. Bob computes the following on the encrypted message he receives using Alice's public key as E(P_A, E(S_A, H(M))), and obtains H(M). He then performs the hash of the message M received, and compares it with this value he decrypted. If they are the same, then the message is intact. If any byte differs, then the message was modified.

Why it works:

Only Alice can sign a message that can be correctly decrypted using P_A. Using the hash function helps check the integrity of the data because if the values match, it means the message was sent by Alice and not tampered with. But if they don't, then the message was sent by Alice but modified on the way. The hash function is good because even one byte change on the message M will produce a totally different H(M).

3) Scenario: Alice wants to send Bob a long message (in this case, it's a signed contract between AliceCom and BobCom), she doesn't want Eve to be able to read it, and she

wants Bob to have confidence that it was Alice who sent the message. Assume for this scenario that AITM is impossible.

Solution:

Alice and Bob use Diffie-Helman to agree on a key , k. They also agree on a protocol for fixing the length of the public key infrastructure. Alice hashes the document D and obtains H(D). Both ALice and Bob decide on secret keys that they keep private and public keys that they share over the internet. Alice then encrypts the hashed document using E(S_A, H(D)). She concatenates the signed document with the encrypted hashed document to obtain D||E(S_A, H(D)) . Alice encrypts the result using AES(k, D||E(S_A, H(D)), which she sends to Bob.

Bob receives a chunk of bytes. He first applies AES_D(k, D||E(S_A, H(D))) to obtain D||E(S_A, H(D)). Based on the agreed length of the protocol, he separates D||E(S_A, H(D)) into two parts, X - the first bytes and Y the last bytes (based on the agreed infrastructure.) the last 256 bytes to have Y, and the first bytes constitute X. Bob then computes E(P_A, Y)) and H(X) and checks if they are equal. If they are equal then it means that the document was signed with Alice's private key. All is good. If not, he cuts the communication.

Why it woks:

Alice is the only person who has S_A. If the two documents match, it means that the document was from Alice and signed by her.

4) - Alice could say that someone somehow got a hold of her secret key, and sent the message (C' || Sig) to Bob, which is different from what she sent. They managed to stop her message (C|| Sig) from being transmitted to Bob somehow, and Bob instead received (C' || Sig).

   As the judge I would think this scenario to be plausible. If someone could get ALice's key, they could send a message that will pass the test as Alice, but it is not her.

   - Alice could say that someone intercepted the message (C || Sig), and somehow managed to change the value of C into a C' and sent that to Bob.

   As the judge I would think that that is not very likely, additionally if Sig was hashed, and H(C) produces the same result, then (C ||Sig) is most likely the message that ALice sent.

   – Alice could claim that Bob changed the file C after decrypting it and just appended the signature from the original message, which he is presenting to the court. Bob could obtain Sig from (C || Sig) and append it to a different file, we could verify by computing H(C) presented by Bob. If the value corresponds to that of the signature, then the correct file is being presented. Else, the scenario might be possible.

   - Alice could say that her signing app showed her one document, but computed H(C) over different bytes (e.g., bad viewer, wrong file handle, hidden/embedded content). So

she unintentionally signed C even though she thought she was signing something else. As a judge I would ask to see proofs of this signing app to judge whether it is possible in this case.

5) Sig_CA = E(S_CA, H("bob.com" || P_B))
   To verify Sig_CA, one will compute E(P_CA, Sig_CA), and check if it matches H("bob.com" || P_B).
6) It is not enough for Alice to know that the message was from Bob. The certificate is available publicly, so anyone can have access to it. Mal could send Bob's certificate to Alice and claim to be Bob. To ensure the authenticity of Bob, Alice and Bob can use Diffie-Helman to agree on a shared secret, k. Alice then encrypts a challenge using Bob's public key and appends that with their shared key, k, and sends it to Bob. i.e E(P_B, R || k). Bob receives the challenge, decrypts it using his private key (E(S_B, E(P_B, R || k). He verifies that the shared secret is correct, then solves the challenge and sends it back to Alice using her public key and k. I.e E(P_A, R' || k). Alice decrypts this using her secret key (E(S_A, E(P_A, R' || k), checks that k is valid and checks that R' is valid as well. Then they know they are talking to each other.

7) - Mal can send Bob's certificate to Alice and claim that she is Bob. Mal could steal S_B and use it to authenticate as the real Bob with Bob's certificate.
   - Mal can trick the certificate authorities to sign "bob.com" associated with her public key P_M.
   - Mal can register a different site with a name that looks similar to "bob.com" and get a certificate associated with P_M. She can use a different letter(e.g a greek omicron, which looks like o). So the certificate is correctly issued and Mal has the correct secret key. Mal can then redirect Alice to look at this her site instead of Bob's. She will then think she is talking to Bob.