Rebecca Fox and Yemi Shin

1. Alice wants to send Bob a long message, and she doesn't want Eve to be able to read it. (I say "Eve" here because I want you to assume for this scenario that person-in-the-middle is impossible, and give an answer that is as simple as possible under that assumption.)

If Alice and Bob want to send each other a long message, they need a key. Assuming they didn't agree on one ahead of time, Alice and Bob use Diffie-Hellman to agree on a shared secret key K. They then use that key K with a symmetric encryption algorithm like AES. Alice would encrypt the message ($S_K(M)$) and send it to Bob who would then decrypt it ($S_K^{-1}(S_K(M))$). Since only Alice and Bob would know the key, Eve would not be able to decrypt the message.

2. Alice wants to send Bob a long message. She doesn't want Mal to be able to intercept, read, and modify the message without Bob detecting the change.

For Alice to send Bob a long message that Mal cannot manipulate without Bob knowing, she can use a Public/Private key pair. Assuming that she has access to Bob's public key, she could encrypt the message M using his public key and can send Bob $C = E(P_{Bob}, M)$. Then, when Bob receives the message, he can decrypt it using his private key $M = E^{-1}(S_{Bob}, C)$. Because only Bob has his private key, Mal would not be able to decrypt the message, read it, and modify the contents of the message.

3. Alice wants to send Bob a long message, 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. (Again, don't worry about Mal and person-in-the-middle here.)

For Bob to have confidence that it was Alice who sent the message, she can use a public/private key with her signature. This means that she would encrypt the message first with her *secret key* before then encrypting that with Bob's public key: $C = E(P_{Bob}, E(S_{Alice}, M))$. Since Eve does not have access to Bob's private key she cannot read the message. Bob, however, is able to decrypt the message using his private key that only he has access to and using Alice's public key: $M = E^{-1}(P_{Alice}, E^{-1}(S_{Bob}, C))$. Since only Alice would have access to her own secret key, Bob can know that it was actually Alice who encrypted/sent the message. In addition, since the message is long, and it is difficult to sign a long message, you can use a hash function to make a much shorter digest that is easily signable.

4. Alice wants to send Bob a long message (in this case, it's a contract between AliceCom and BobCom). She doesn't want Eve to be able to read it. She wants Bob to have confidence that it was Alice who sent the message. She doesn't want Bob to be able to change the document and claim successfully in court that the changed version was the real version. And finally, Bob doesn't want Alice to be able to say in court that she never sent the contract in the first place.

If Alice doesn't want Eve to be able to read the message, confidentiality has to be achieved. Encrypting the document using public/private keys, or symmetric encryption such as AES would help. In terms of reassuring Bob that it was indeed Alice who sent the document (authentication), as well as making sure that the document does not get modified by Bob or any other party for that matter (integrity), this can be achieved by first using a hash function (such as SHA-256) to get a digest of the initial document, and then encoding this digest with Alice's secret key, which provides the digital signature, and finally concatenating this signature with the message and sending it to Bob. (Is the message encrypted? probably) Bob, on the other end of the line, would then use Alice's public key and signature to decrypt the digest, and then apply the hash function to the message to see if the digest he gets matches that of Alice. If they match, Bob can be sure that the message was not corrupted in some way along transmission, and that only the owner of Alice's private key (which we assume will just be Alice) could be the author of the document. Another way of ensuring both authenticity and integrity is using a HMAC. MAC (Message Authentication Codes) is a general method where Alice and Bob share a secret key, and Alice uses a MAC function on this secret key and the message to get a *code*, and concatenates this code to the message and sends it over to Bob. Bob would then apply the same MAC function with the secret key and the message, and see if the code he gets matches that of Alice. If they match, Bob can be sure that the message message was not corrupted, and because they share a secret key, based on our assumptions, it can be ensured that it was Alice who sent the message. HMAC is a popular way of doing MAC, where you hash the document concatenated with the secret key, so Bob can also verify the integrity of the received message by applying the hash function and verifying that the digests match.