Nicholas Vlahos-Sten

Jeff Ondich

Computer Security

5 October 2025

Cryptography Scenarios

1. Pretty simple. If AITM is not possible and Alice wants to send Bob the long message M, Alice and Bob can use Diffie-Hellman to agree on a shared secret and derive a shared key K which they will use as a key for the symmetric encryption algorithm AES(K,M). Alice will send Bob C, which is the ciphertext obtained from AES(K,M) and he will decrypt C with AES_D(K,C) to get her message. Eve cannot decrypt C without K, and it is practically computationally impossible for her to figure out K because it would require breaking Diffie-Hellman.
2. I am interpreting this as simply as possible, and am going to say that all Alice needs to do is send Bob her message M concatenated with a SHA-256 hash of the M, H(M). Once Bob gets the message he believes to be M, M', he can run H(M') and if H(M') doesn't equal H(M) he knows the message got modified somewhere in the middle, and knows not to trust it. If they do match, he's confident the message got modified. Now, I believe this only works if A) the attacker in the middle doesn't know Alice and Bob are using an SHA-256 hash (they don't know what hashing algorithm they're using) or if B) they can only modify the message, not the hash. They could create a new message with a new

hash and send it to Bob to circumvent this. I interpreted this question to be that Mal only can modify the message not the hash of the message so this would work. If Mal can do more, we'd need the public key hash I describe in question 3, but I opted for the simpler method here.

3. First, Alice is going to do the same steps that I laid out in question 1 to encrypt her message– create a Diffie-Hellman key and use that key to run AES. Now Eve cannot read the message, but she still needs Bob to trust that she sent it. So she creates a digital signature. She hashes the encrypted message M with H(M) like in question two, but also encrypts the (now appropriately short) hashed message with public key encryption. She then sends Bob M || sig where sig is $E(P_A,H(M))$. Once Bob receives this, he can take the M he received, call it M', and put it through SHA-256 to get H(M'). He then encrypts the sig Alice sent using Alice's public key to get $D' = E(P_A,Sig)$. If D' = H(M'), then the private key and public key encryptions match, which means that Alice's public key matches her private key and that Bob is certain he is in fact talking to Alice.
4. She could claim that an attacker in the middle intercepted her message and replaced the contract she sent with his own fake contract. A judge would not find this believable, since modifying the message would also make the sig not match, and Bob and the court would have been able to realize the sig doesn't match the message. Alice could more plausibly claim (if you get rid of the assumption that everyone's public key is correct) that Bob had never been in communication with Alice in the first place, that he initially messaged Mal and Mal sent the fake contract. In this case, Mal's public key and sig would match, and Bob was at fault

for believing that was Alice. As a judge I would like to see some evidence that this happened, and I would place the burden of proof on Alice to show that this unlikely event occurred. She could also claim that Bob acquired her secret key somehow and used it to encrypt a fake message to show in court. Since secret keys should not be breakable by mathematical means, as a judge I would have to see proof that the key was stored somewhere and that Bob hacked into that somewhere to read it. If she was just careless and shared her secret key, I guess in the law she it would still not be a breach of contract but poor move on her part.

5. The formula would be $Sig_{CA}= E(S_{CA}, H(\text{“Bob.Com”} \mid\mid P_B))$. By “Bob.Com” || P_B I mean that it computes the Hash of everything in the certificate except the signature (that it does not yet have).
6. It is not enough for Bob to just send the certificate, because Alice has to first verify that the Sig_{CA} on the certificate corresponds to P_{CA} (he could have just sent some made up certificate). So she should use the CA's public key to compute $E(P_{CA}, Sig_{CA})$ and compare the result to $H(\text{“Bob.Com”} \mid\mid P_B)$. Basically, Alice manually computes the hash of the certificate minus the sig and then checks if reversing the sig with the public key gives her the same result as the manual computation, which it will if sig was computed using S_{CA}. If they are equal, then Alice knows that the certificate was made with CA's digital signature, and since Alice trusts CA, if CA trusts Bob, then Alice trusts Bob.
7. For one, Mal could convince CA that it is Bob. If Mal says to CA “hey I'm Bob validate my key” and CA issues a challenge like “hey change something on your website” and Mal got access to Bob's website by breaking its password for

example, it could pass the challenge and convince CA that it is Bob. Then when it sends Cert_B to Alice, Alice verifies it and sees that CA said that Mal is Bob so Mal must be Bob. Alternatively, if Alice talks to Mal but thinks she's talking to Bob, Mal could then go ask Bob for Bob's certificate and send it to Alice pretending its its own certificate (although not sure how this would work if the website details on the certificate are clearly different you'd have to screw with them if you were Mal somehow). And I know there's the assumption that everyone has CA's public key, but Mal could also impersonate CA and issue a certificate to himself.