[Aidan Mcdonald](mailto:Aidan.McDonald@uconn.edu)

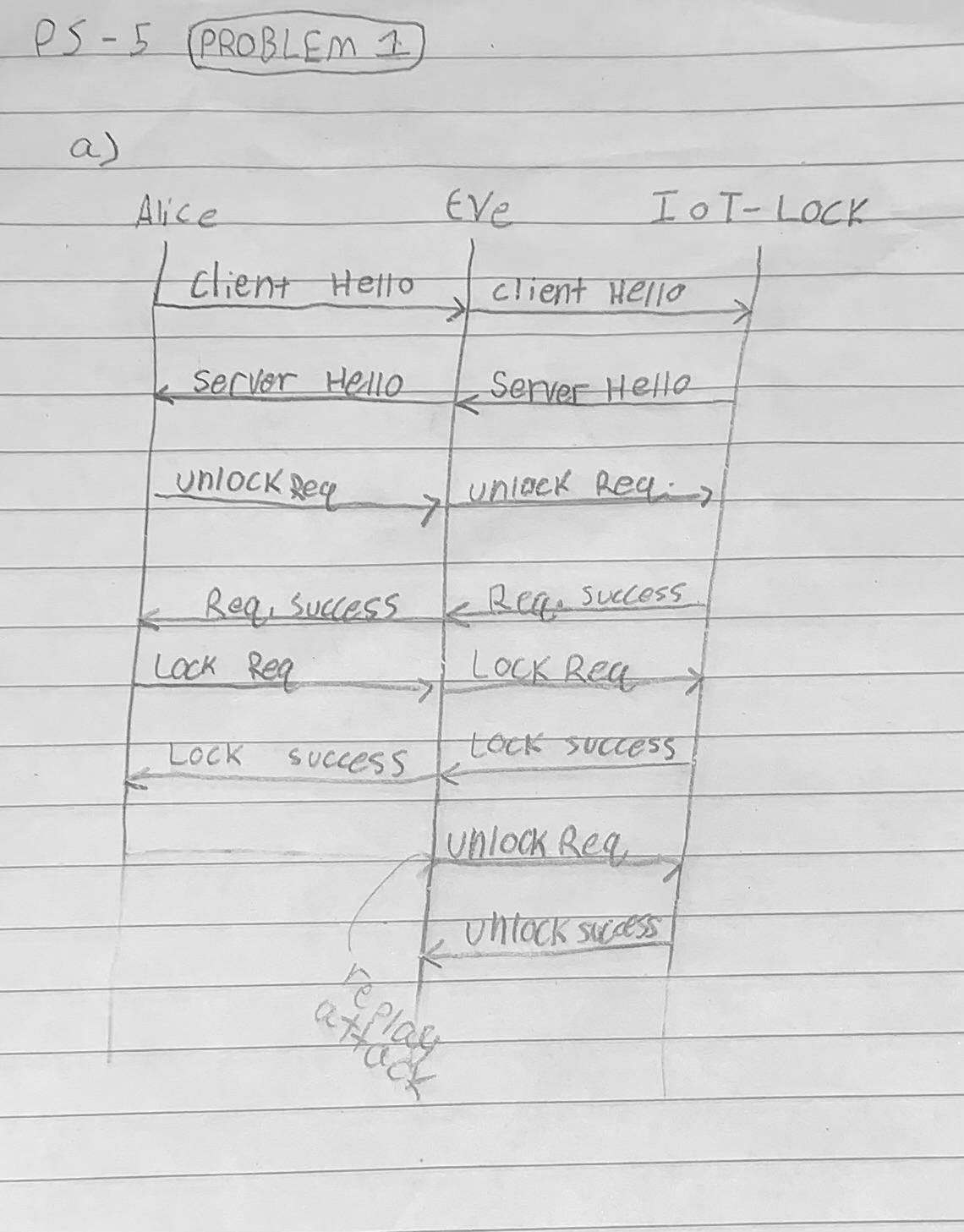
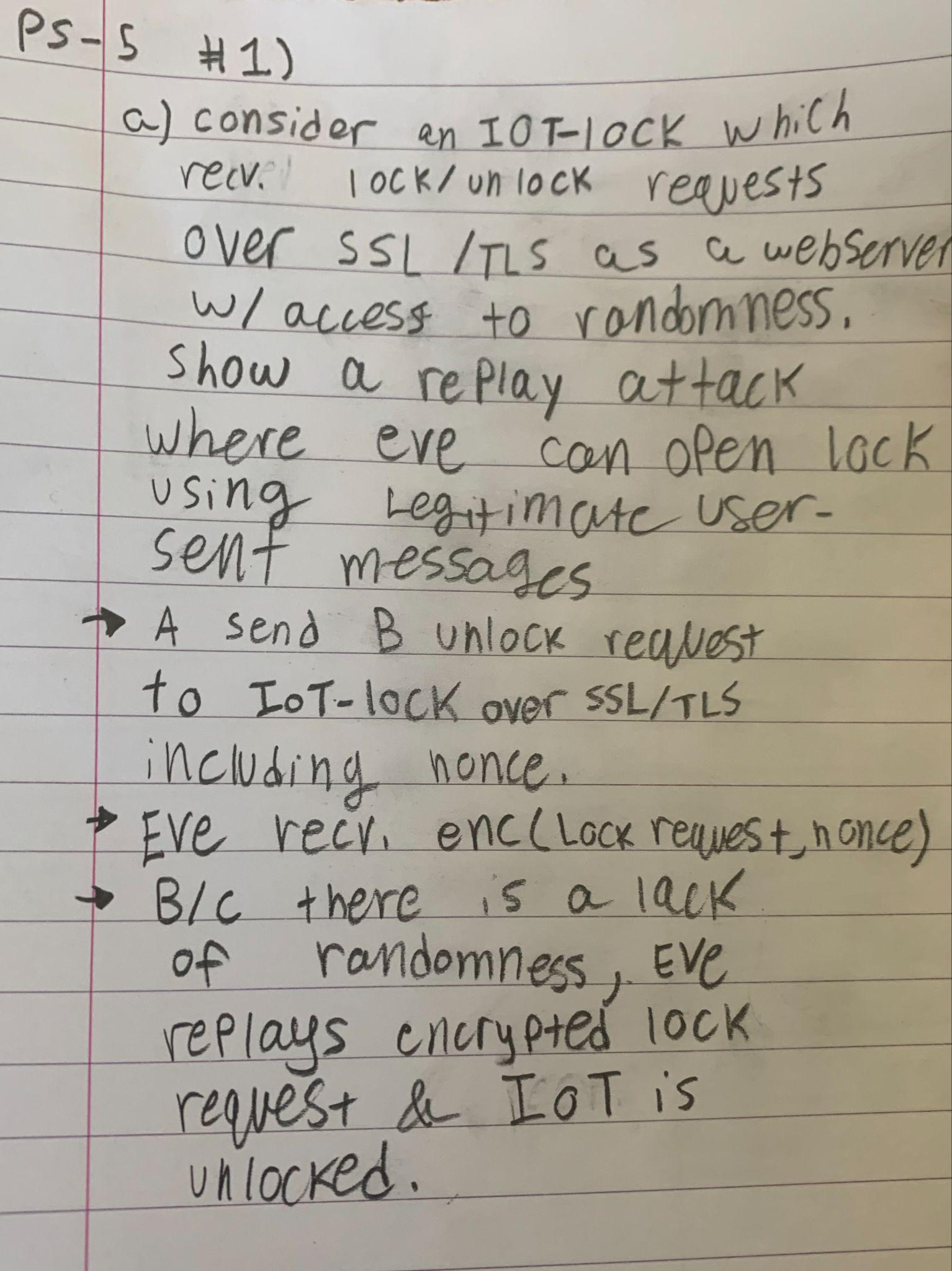
Tommy Jiang

Luke Pepin

CSE 3400 Problem Set 5

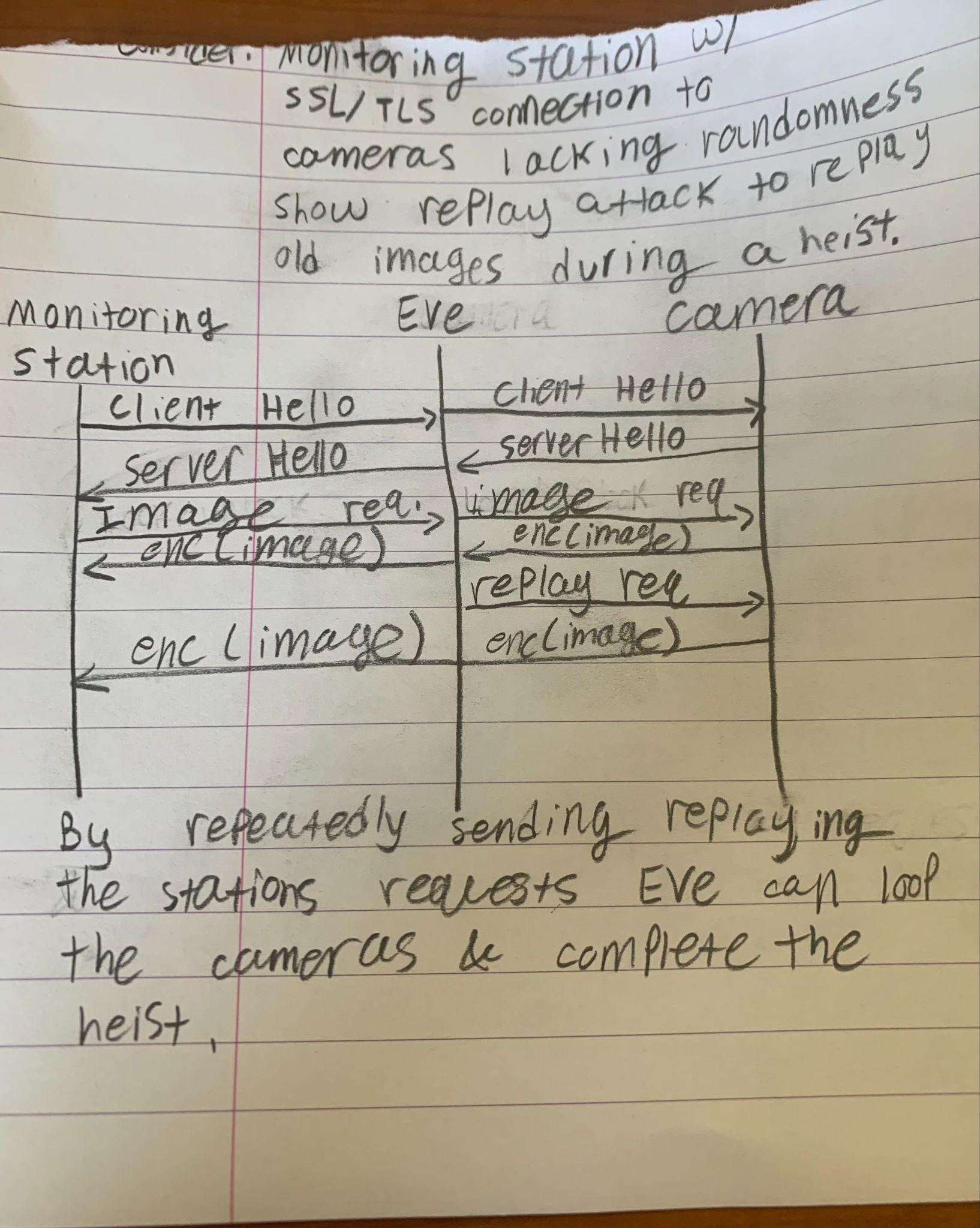
5/1/24

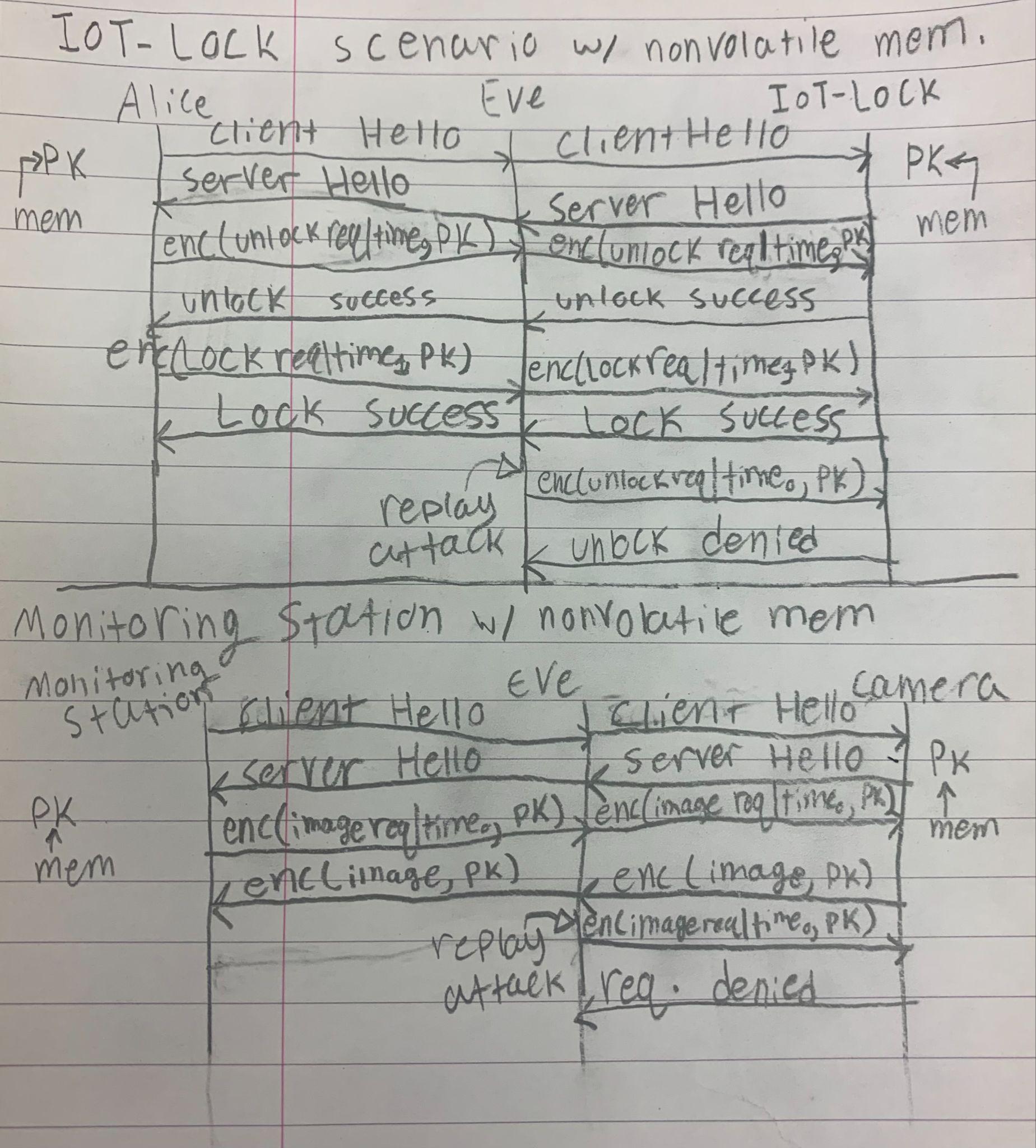
1. **Exercise 7.9** (Non-random client/server). Some devices may not have a source of random bits; in this exercise, we explore possible resulting vulnerabilities, and a possible work-around.
   1. Consider a IoT-lock, which receives the lock/unlock requests over SSL/TLS, as a web-server - but without access to a source of randomness. Show a replay attack allowing an eavesdropping attacker to open the lock by replaying messages of a legitimate user.



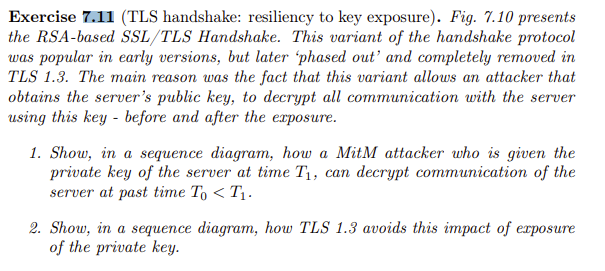
(explain more replay work what it by passes(looks good)

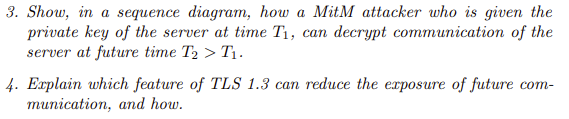
* 1. Consider a monitoring station displaying images from security cameras, by initiating SSL/TLS connections to the cameras and receiving the current images. Show a replay attack allowing an eavesdropping attacker to replay old images (while cracking the safe).

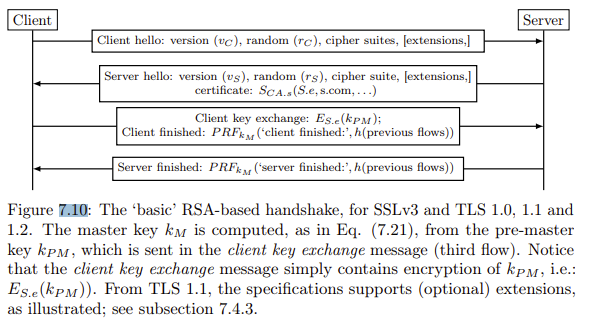


* 1. Assume that the devices have non-volatile memory. Show how it can be used to ensure secure interactions, even though the devices still do not have a source of random bits.

Even though the IoT lock, Alice, the monitoring station, and the camera do not have access to random bits but do have nonvolatile memory, we can have random private keys generated and kept in nonvolatile memory before they are sent out for use. This allows both scenarios to benefit from agreed upon random bits while still maintaining security given an eavesdropper and an insecure communication channel during handshake. By encrypting the timestamp appended after the request we can help prevent replay attacks as they will have a timestamp associated with them once they are decrypted by the camera or IoT lock.





 From the textbook pg 433 and 477 and 476

1)

Client Attacker Server

| | |

T0: Client hello | |

|------------------------------->

T1 | | (Attacker obtains key) || |

| | |

T2:| | Server hello |

|| < -------------------------------

| | |

T2 | | Certificate |

| |

Explanation:

Absolutely, that's a significant vulnerability in RSA-based SSL/TLS implementations. Once an attacker obtains the server's private key, they can retain/record it indefinitely, allowing them to decrypt past communication after that point i my graph be T1.

2)

Explanation: This sequence diagram correctly depicts the TLS 1.3 handshake process, showcasing how ephemeral session keys are utilized to secure communication. Unlike earlier versions, TLS 1.3 generates ephemeral session keys during the handshake, which are discarded after the session ends. This ensures that even if the server's private key is compromised after the handshake (T2 > T1), past and future communication remains secure. By utilizing ephemeral session keys, TLS 1.3 achieves forward secrecy, enhancing the resilience of the protocol against key exposure attacks.

3)

Client Attacker Server

| | |

| | |

| | |

| | |

| | |

| | |

| | |

| | |

| | |

| | |

| | |

| | |

| | (Attacker obtains key) |

| | |

| | (Decrypts future messages) |

Explanation: This diagram accurately demonstrates how a MitM attacker, armed with the server's private key obtained at time T1, can decrypt future communication between the client and server (T2 > T1). Once the attacker has obtained the key and recorded it, they can decrypt future messages because the server continues to use the same key, which the attacker possesses.

4)

Explanation:Since in TLS 1.3, the keys are discarded shortly after T2, any attack attempted after that point wouldn't be effective because the keys would no longer be available. Without access to the keys, attackers would be unable to decrypt past or future messages. After T2, any attempted attack would be futile as there would be no key to decrypt the communication.

**Exercise 8.12.** A website eve.com receives a TLS certificate from a CA CA.org. What should prevent eve.com from using this certificate to impersonate as the website for domain bob.com? Present a mistake that CA.org could make in the certificate, which would allow this attack.

To prevent eve.com from using a TLS certificate issued by CA.org to impersonate the website for the domain bob.com, some security measures must be in place. Such as:

**Certificate subject and Common Name (CN) validation:** This process verifies that the domain name specified in the certificate matches the actual domain of the website, preventing unauthorized use.

**Certificate Authority Authorization (CAA):** CAA records specify which certificate authorities are allowed to issue certificates for a domain, reducing the risk of unauthorized certificate issuance by restricting the set of permissible CAs.

**Certificate Transparency (CT):** CT mandates that all issued certificates are logged in publicly accessible repositories, enabling website owners and users to detect unauthorized certificates and potential security breaches.

These measures collectively ensure that only authorized and proper certificate usage happens on eve.com.

A mistake that CA.org could make in the certificate allowing this attack is failure to enforce proper certificate subject validation. If it doesn't properly validate the domain ownership during the certificate issuance process, they might issue a certificate for bob.com to eve.com. As a result, eve.com could obtain a TLS certificate for bob.com allowing them to impersonate the website for bob.com.

1. Suppose that an adversary, Eve, creates a public and private key and tricks Bob into registering pkE as a trusted certificate authority (that is, Bob thinks Eve is a trusted certificate authority). Assume that Alice is a web server and Bob a client and they connect to one another using SSL/TLS. Show a Man in the Middle attack allowing Eve to listen in on all communication between Alice and Bob even if Alice and Bob use SSL/TLS and even if Alice (the server) has a certificate from a real certificate authority (Alice does not trust Eve as a CA).

SSL/TLS Handshake with MitM Attack:

1. Bob initiates connection to Alice. However, Eve intercepts this connection since she tricked Bob into registering pkE as a trusted certificate authority.
2. Eve, pretending to be Alice, initiates a new SSL/TLS connection with Bob. Eve sends her own certificate signed by her private key.
3. Since Bob trusts Eve as a CA, Eve can impersonate Alice. Bob then accepts the certificate and believes he is communicating with Alice.
4. At the same time Eve initiates a new SSL/TLS handshake as a client with Alice who responds with her certificate signed by a real CA which Eve has no issues validating.

Now with 2 separate SSL/TLS connections: one where Bob is talking to Eve (thinking it is Alice) and another where Eve is connected with Alice (acting as a client).

SSL/TLS: 1 SSL/TLS: 2

Bob —----------------------------------------- Eve —------------------------------------------Alice

*Client “Server”:1, “Client”:2 “Server”*

On a given message, Bob encrypts it using Eve’s public key, mistakenly believing it is Alice’s public key. Eve receives and decrypts the message with her private key, gaining access to the plaintext which she can alter as desired. Subsequently, Eve can re-encrypt the message using Alice’s public key, acquired during the second SSL/TLS handshake, before sending it to Alice.

Since both Alice and Bob are deceived by Eve, communication from Alice to Bob is compromised. They maintain the same SSL/TLS connection, unwittingly employing Eve’s public and private keys for encryption and decryption.

As all communication flows through Eve, both Bob and Alice wrongly perceive they are directly communicating. In reality, Eve intercepts and potentially modifies all their communication.