HW3 COSC3371 Cybersecurity
Max Chalitsios
1808500

Problem 1

The Sender and Receiver classes each have its own pair of RSA keys (public and private) generated from the python RSA module. The Sender class also contains a message attribute for the plaintext message. The Sender encrypts the message with the Receiver's public key using the encrypt method provided by the RSA module. The ciphertext is stored in *sender.ciphertext*. The Sender then creates a digital signature for the message which involves computing a hash of the message with *rsa.compute_hash* method and signing the hash with the Sender's private key using *rsa.sign_hash* method. The signature is stored in *sender.signature*. The Receiver decrypts the ciphertext with their private key using the *rsa.decrypt* method and it is stored in *receiver.message*. Then the Receiver verifies the Sender's signature by checking if valid for the decrypted message and the Sender's public key using the *rsa.verify* method.

Problem 2

In this problem, we simulated an attack on the message. After the Sender signs the original plaintext, we alter the message by switching two bytes in the plaintext. This is equivalent to an attacker modifying the message after it has been signed. When the Receiver decrypts the message and tries to verify the signature, the verification fails because the signature was generated for the original message, not the altered message. We see this when handling the *rsa.VerificationError* exception from the RSA module when the verification fails.

Problem 3

In this problem, we simulated an attack on the signature. After the Sender signed the original plaintext, we altered the signature by flipping a bit in the second byte of the signature which is equivalent to an attacker modifying the signature after it has been generated. When the Receiver decrypts the message and tries to verify the altered signature, the verification fails and throws a *rsa.VerificationError* exception. The point is that any modification to the message or the signature after the signature has been generated will cause the signature verification to fail. This is what makes digital signatures secure against tampering.