Christian Szablewski-Paz & Anton Danylenko

CS 396

9 December 2022

HW 2

Problem 1:

Note, on Java 8, the encryptBlock() function was throwing this error: java.security.NoSuchAlgorithmException: Invalid transformation format:AES/ECB/. To fix this, we added "PKCS5Padding/" to the mode variable defined in MacSkeleton.

Mac Tag:

75D617D3BDB4131E2A80AD3CE6ED8AADF295FDF32A9B2DE2370936783F92381A

The algorithm we use first pads the message to the appropriate block size, and then splits the padded message into multiple blocks of size block size. The MAC is initialized to 0, and then the code algorithm over each block of the padded message, encrypting the current block using the given key. The encrypted block is then XORed with the MAC, and the result is used as the MAC for the next iteration. After all blocks have been processed, the resulting MAC is returned. Our current implementation works for messages of any size.

Problem 2:

If Cindy manages to steal Bob's secret key, she can use it to impersonate Bob and decrypt any messages that are encrypted with Bob's public key. In order to continue to do so even after Bob has registered a new public-key pair with the CA, Cindy could collaborate with Mallory in order to obtain the updated public key directory and certificate. They could do this by Mallory giving Cindy the updated directory and certificate, along with a fake proof of authenticity for Bob's new

public key so that Cindy could continue to decrypt messages sent to Bob without anyone becoming suspicious. This type of attack is known as a man-in-the-middle attack, where an attacker is able to intercept and manipulate communication between two parties without either party realizing it. The use of a fake proof of authenticity is known as a "replay attack".

One countermeasure would be to use a digital signature scheme in addition to the public-key encryption. In a digital signature scheme, each user has a unique digital signature that is generated using the user's secret key. Whenever a user sends a message, they can sign it with their digital signature. The recipient can then verify the signature using the sender's public key, ensuring that the message has not been tampered with and that it was sent by the actual user. Also, another thing you can do to prevent replay attacks is having the system use nonces in the authentication process. In this case, the CA could include a nonce in the proof of authenticity it gives to Mallory, then Mallory could include the nonce in the proof of authenticity it gives to Alice. Alice could then verify that the nonce is fresh and reject the proof if it wasn't. This would prevent an attacker from reusing an old proof of authenticity to in attempt to impersonate Bob.

Problem 3:

2. These operations are realized more efficiently in practice through the use of small public exponents, e. We would personally recommend using a small e down to e = 3. In RSA, it is the size of d that is significantly more important to protecting the message m than e. By the Chinese Remainder theorem, say e = 3, and a message m is encrypted with public keys n1, n2, and n3: you will then have ci = m^3 mod ni, and an attacker would be able to decrypt m. However, in

practice, padding is used for encryption. The weakness here would actually not be the small exponent, but it would be the use of improper padding for the encryption which is primarily concerned with the size of d.