COSC2804: PROGRAMMING STUDIO 2

Assignment 3

Team 9: Andrew Bibart (s3785126), Cathy Le (s3946815), Vidhathra Murali (s3944867) & William Dash (s3947523)

Encryption methods

Hash-then-encrypt

Hash-then-encrypt computes the hash of the message, and then proceeds to attach the message to the generated hash. Next, the entire thing is encrypted. Hash-then-encrypt is not an authenticated encryption technique as it is not secure.

Pros of using hash-then-encrypt:

- Compared to the other options, hash-then-encrypt is simpler to implement and more efficient to run. As there are less steps in it's process.

Cons of using hash-then-encrypt:

- Hash-then-encrypt only uses a single key in its entire process. This makes it less secure and much more susceptible to brute force attacks. Additionally, since it does not use MAC, it doesn't verify the authenticity of the data, meaning there is no way to confirm who sent the data to you.

MAC-then-encrypt

MAC-then-encrypt is a technique which first computes the MAC of the plaintext message (which is adding a secret key to ensure its integrity), and then it will attach it to the message. After that, the message is then encrypted in ciphertext using another separate key.

Pros of using MAC-then-encrypt:

- By using MAC on the plaintext message, this ensures integrity of the message's data before any ciphertext encryption occurs to the message.

Cons of using MAC-then-encrypt:

- Because the MAC is applied before the encryption process, this means that there is a possibility that a middleman attack will be able to see the message after the encryption occurs as there is no MAC attached to that message (meaning no integrity). This defeats the aim of trying to keep the data confidential.

Encrypt-then-MAC

Encrypt-then-MAC generates the ciphertext by encrypting the plaintext first, then appends a MAC based on the cyphertext.

Pros of using Encrypt-then-MAC:

- Ensures the integrity of the received message since any invalid or changed ciphertext will be rejected by comparing the MAC
- Encrypting the plaintext first protects sensitive information, so confidentiality is guaranteed

Cons of using Encrypt-then-MAC:

- Risk of timing attacks: the time variations to perform cryptographic operations can in some cases let the attacker gather information on the MAC verification and possibly gain insights into the encrypted message.
- Can take up space and resources if the message is large

Encrypt-and-MAC

Encrypt-and-MAC first computes the MAC on plaintext, encrypts the plaintext and then appends the MAC at the end of the ciphertext. The result returned is the pair of the ciphertext and MAC.

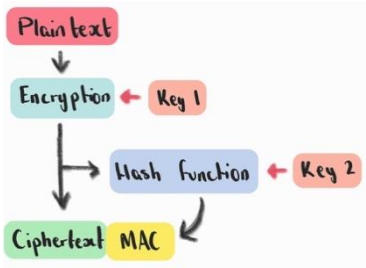
Pros of using Encrypt-and-MAC:

- Simple construct – the construction is very simple and easy to understand
- It is efficient in terms of in computational overhead as the MAC is computed on the plaintext message prior encryption and therefore reducing the processing time.

Cons of using Encrypt-and-MAC:

- Lack of integrity on the ciphertext as the MAC is taken against the ciphertext. Increases vulnerability to some cipher-chosen attacks on the cipher
- Does not preserve confidentiality – two encryptions of a message will have the same MAC tag, therefore the messages are equal

Report



Among various cryptography methods, Encrypt-then-MAC (EtM) is a commonly preferred method to ensure the confidentiality and the integrity of data. It is primarily used to achieve secure communication by protecting the information exchanged and preventing unauthorised access.

Encryption
EtM works by first encrypting the plaintext using a symmetric encryption algorithm such as Advanced Encryption Standard (AES) thus producing a ciphertext. This encryption process relies on a secret key that should only be known by the sender and the intended recipient.

MAC - Message Authentication Code
After producing the ciphertext, a MAC is generated. This MAC acts as a proof of integrity to ensure that the message is not tampered with.

The MAC is created by using a cryptographic hash function (CHF). A CHF takes data and translates it into a fixed length hash value. In the case of EtM the CHF will take the ciphertext and the secret key as a combined input and outputs a fixed length hash.

Appending and Transmission
The generated MAC will then be appended to the encrypted ciphertext so that the message is not only protected by encryption but also so that message manipulation cannot go undetected.

The ciphertext along with the appended MAC is sent to the intended recipient. Using the secret key, the recipient can decrypt the ciphertext and retrieve the original plaintext message.

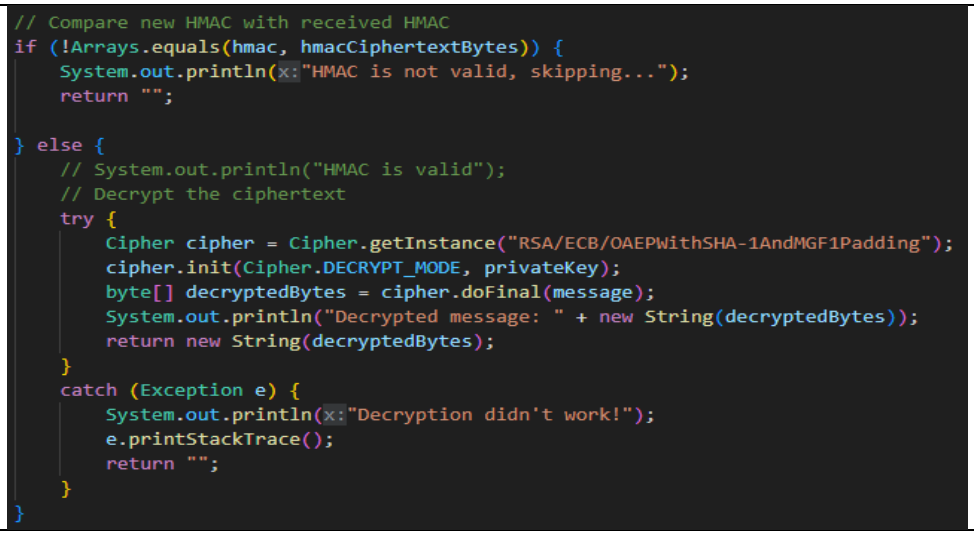
Decryption and Verification
To verify the integrity of the message the recipient recalculates the MAC using the decrypted plaintext using the same CHF and the shared key. The recipient can verify the authenticity of the message by comparing the recalculated MAC with the received MAC. If both MACs match, then it indicates that the message is authentic, and its integrity has not been compromised during transmission. However, if they don't match it implies that the message has been compromised and could have been tampered or manipulated with.

EtM is a method that ensures that the confidentiality of the message is preserved, and the integrity of the received message can be verified through the MAC.

Code Snippets

(1) - Generating the keys to be used in encryption, decryption, and the HMAC. – (RemoteSession.java, lines 81-90)	`// Initialising Key Objects`
	`KeyPairGenerator genPrivateKey = KeyPairGenerator.getInstance(algorithm:"RSA");`
	`KeyGenerator secretKeyGen = KeyGenerator.getInstance("HmacSHA256");`
	
	`// Initialising the Key`
	`genPrivateKey.initialize(keysize:2048);`
	
	`// Generating the Keys`
	`KeyPair keyPair = genPrivateKey.generateKeyPair();`
	`secretKey = secretKeyGen.generateKey();`

(2) - Encrypting the plaintext message using the public key, as well as computing the HMAC of the encrypted message. - (connection.py, lines 54-60)	`# Encrypting message with public key`
	`public_key = RSA.import_key(self.pub_key)`
	`cipher = PKCS1_OAEP.new(public_key)`
	`message_encrypted = cipher.encrypt(message)`
	
	`# Combining the HMAC with the message using secret key`
	`hmac_msg = hmac.new(self.secret_key, message_encrypted, hashlib.sha256).digest()`

(3) - Encrypted message proceeds to be decrypted from ciphertext only if the initial HMAC is identical to the newly generated HMAC to ensure message integrity. - (RemoteSession.java, lines 122-142)	

```java
// Compare new HMAC with received HMAC
if (!Arrays.equals(hmac, hmacCiphertextBytes)) {
    System.out.println(x:"HMAC is not valid, skipping...");
    return "";

} else {
    // System.out.println("HMAC is valid");
    // Decrypt the ciphertext
    try {
        Cipher cipher = Cipher.getInstance("RSA/ECB/OAEPWithSHA-1AndMGF1Padding");
        cipher.init(Cipher.DECRYPT_MODE, privateKey);
        byte[] decryptedBytes = cipher.doFinal(message);
        System.out.println("Decrypted message: " + new String(decryptedBytes));
        return new String(decryptedBytes);
    }
    catch (Exception e) {
        System.out.println(x:"Decryption didn't work!");
        e.printStackTrace();
        return "";
    }
}
```

Security Analysis

Confidentiality

The encryption process of EtM which first encrypts the plaintext then generates the MAC based on the resulting ciphertext, rendering it confidential and unintelligible without the decryption key. The MAC function is created by the keyed Hash Message Authentication Code (HMAC), which uses a secret key and the encrypted ciphertext as input. EtM uses distinct keys meaning only the intended recipient who has access to the secret key and decryption key can generate the MAC and decrypt the message.

Integrity

Utilising the EtM approach, the integrity and authenticity of the encrypted message are protected. In EtM the role of HMAC is generated using a CHF which is a single direction work, which makes it extremely difficult to reverse to reconstruct the data that was used to produce it. A reliable CHF should also be able to survive all known types of attacks on the hashing equation itself. The MAC verifies the ciphertext's integrity by identifying any unauthorised changes or altering attempts. It also guarantees the ciphertext's authenticity by confirming that it was created by a reliable source.

Security Trade-offs

EtM is a well-balanced method to ensure security and confidentiality however it can still sometimes be susceptible to attacks other than "man in the middle".

1. Efficiency: It may introduce some computational overhead due to the resources used for generating the MAC and the data required for computation. Therefore, the processing times and resource utilisation would generally increase compared to just using encryption or a Mac alone.
2. Key management: EtM requires the management of two keys: an encryption key and a MAC key. This requires and adds some more complexity to the key management process to ensure integrity and confidentiality of the system.
3. Timing Attacks: When implementing the MAC generation or verification process, timing attacks may be a problem. An attacker could learn details about the MAC or the key by watching the timing of MAC verification activities. To avoid timing-based side-channel attacks, care must be made to ensure constant-time activities during MAC verification.
4. Replay attacks, in which an attacker intercepts and retransmits previously valid ciphertexts and MACs, are not prevented by encryption-then-MAC alone. Replay attacks can be lessened by adding extra defences like a date or a distinctive identifier to the message or by using a nonce.

REFERENCE

(1958) *Should we mac-then-encrypt or encrypt-then-mac?* Available at: https://crypto.stackexchange.com/questions/202/should-we-mac-then-encrypt-or-encrypt-then-mac/205#205 (Accessed: 31 May 2023).

(2004) *Encrypt-then-MAC*. Available at: https://www.daemonology.net/blog/2009-06-24-encrypt-then-mac.html (Accessed: 31 May 2023).

Bellare, M. and Namprempre, C. (2000) *Authenticated encryption: Relations among notions and ... - springer*. Available at: https://link.springer.com/content/pdf/10.1007/3-540-44448-3_41.pdf (Accessed: 31 May 2023).

cdjk (2012) *Could you explain why 'encrypt then mac' is the right option please? it isn't ob...: Hacker news*. Available at: https://news.ycombinator.com/item?id=4779212 (Accessed: 31 May 2023).

Guttman, P. (no date) *Encrypt-then-MAC for Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS), " RFC editor*. Available at: https://www.rfc-editor.org/rfc/rfc7366.txt (Accessed: 31 May 2023).

(2014) *Why is plain-hash-then-encrypt not a secure MAC?* Available at: https://crypto.stackexchange.com/questions/16428/why-is-plain-hash-then-encrypt-not-a-secure-mac (Accessed 30 May 2023)

(2010) *What is the Difference between a Hash and MAC (Message Authentication code)?* Available at: https://stackoverflow.com/questions/2836100/what-is-the-difference-between-a-hash-and-mac-message-authentication-code (Accessed 31 May 2023)

Block cipher (no date) *Block Cipher - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/computer-science/block-cipher (Accessed: 09 June 2023).

Paul, J. (no date) Department of Computer Science | Authenticated Encryption. Available at: https://www.cs.jhu.edu/~astubble/dss/ae.pdf (Accessed: 01 June 2023).

Message authentication code (no date) *Message Authentication Code - an overview | ScienceDirect Topics*. Available at: https://www.sciencedirect.com/topics/computer-science/message-authentication-code (Accessed: 09 June 2023).

Security encyclopedia (no date) *What is a Cryptographic Hash Function (CHF)?* Available at: https://www.hypr.com/security-encyclopedia/cryptographic-hash-function#:~:text=A%20cryptographic%20hash%20function%20(CHF)%20is%20an%20equation%20used%20to,size%20numerical%20string%20%E2%80%94%20the%20hash. (Accessed: 09 June 2023).

(No date) *A beginner's guide to constant-time cryptography*. Available at: https://www.chosenplaintext.ca/articles/beginners-guide-constant-time-cryptography.html (Accessed: 09 June 2023).

Puneet (2023) *What is RSA? how does an RSA work?*, *Encryption Consulting*. Available at: https://www.encryptionconsulting.com/education-center/what-is-rsa#:~:text=The%20Rivest%2DShamir%2DAdleman%20(,rypt%20and%20decrypt%20data. (Accessed: 09 June 2023).