An Analysis of Client/Server Security Techniques

Security Assignment Report

Figure 1 . Cybersecurity concept collage design | AI-generated image

Lecturer: Paul Jacob

Student A00277326: Ivan Lapickij

Date: 21/03/2025

Table of contents

Word Count: 1250

1. Introduction (short and possibly including some definitions)

This is my 4th year software security assignment report where I compare and contrast different cryptographic methods to ensure authentication, integrity and confidentiality in client/server settings.

For a long time people have been trying to pass secret messages to each other, since then till now the ways and technology had advanced to the point where not everyone that uses security services these days actually understands how it works.

Terms and definitions:

SSH - stands for “Secure Shell” network protocol that gives users secure way to access computer over an unsecured network.

HMAC - is a standardized way that describes how a message and a key should be combined to generate a digest.

HTTPS - HyperText Transfer Protocol Secure. This protocol defines encrypted communication between web browser and website, making communication safe.

Hash- ensures message integrity by creating a fingerprint of message(digest).

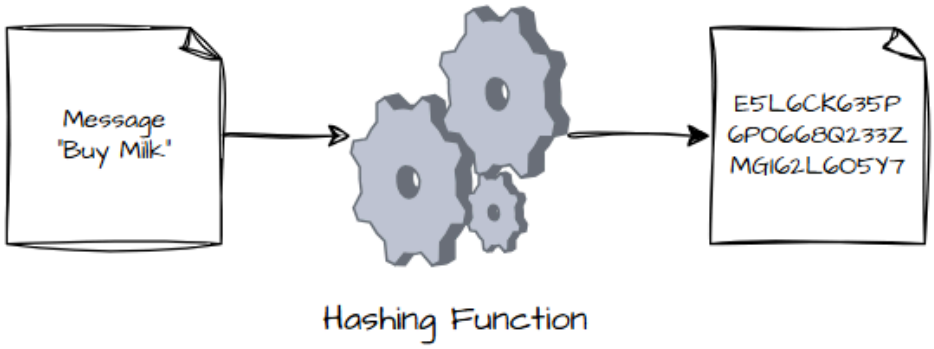

Figure 2. Illustrating Messages Hashing process.

It's very important to clarify understanding by underlining approaches like symmetric and asymmetric. Where both could contain key exchange and encryption or be part of a broader field of cryptography. That means that these terms exchange, encryption, cryptography could be symmetric and asymmetric.

There is three main aspects of security:

Authentication - this aspect is important because received information could influence decision making and our understanding. If this aspect would be exposed we could be adversely affected to take most probably unfavorable decisions.

Integrity - in security stands for assurance of messages original version, elsewhere if messages were not altered.

Confidentiality - some messages hold sensitive data that if exposed could harm etc. financially. The process of securing these messages is called cryptography. The goal of this process is to hide the content of the message so that only the receiver could read it. Over time technology kept evolving so had cryptography because it became easier with better/faster technology to find vulnerabilities and get crack encryption.

Further I'm gonna compare and contrast different techniques in terms of these 3 aspects.

2. Client Accesses a Public API (HMAC)

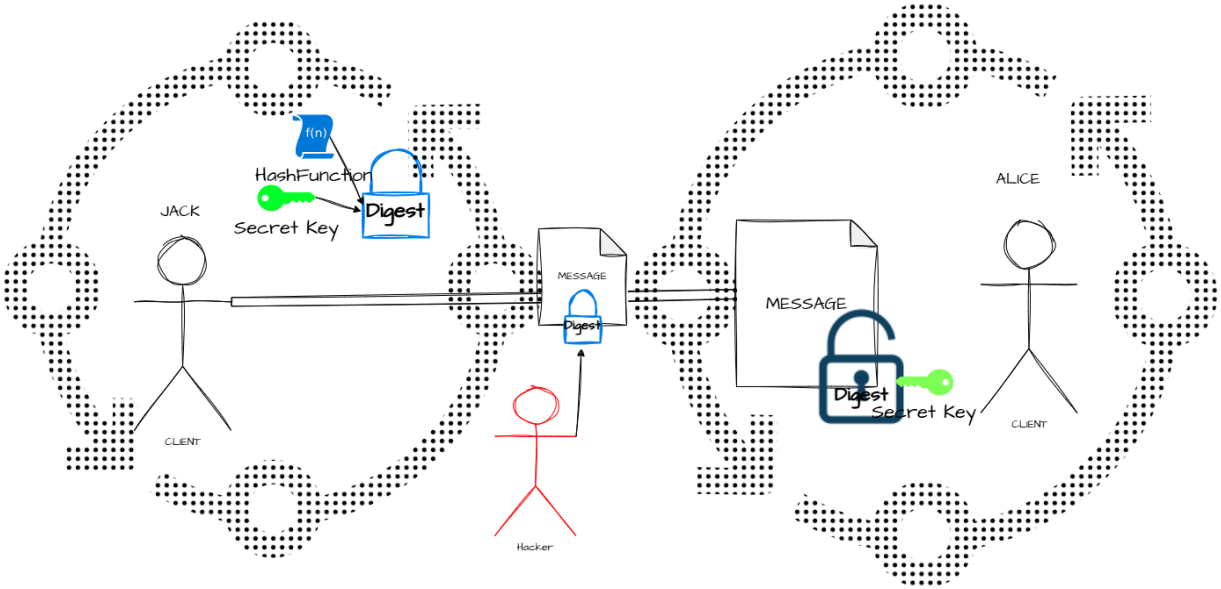

Figure 3. Illustrating Symmetric usage of secret keys with HMAC.

Symmetric cryptography is the approach of accessing public api by verifying authentication & integrity with two ways of key exchange:

Symmetric key exchange involves shared secret keys that are used in combination with hash function to produce a digest that reveals if the message was altered(*integrity*) and that it comes from a person that has a secret key(*authentication*). It is not safe since both parties use the same secret key on which they have to agree and exchange before proceeding further.This approach faces the dilemma of how secret keys should be shared in the first place and that's why asymmetric encryption appeared.

Asymmetric key exchange on the other hand is more safe. The shared key may be established through an asymmetric key exchange protocol, such as Diffie-Helman, where only public keys are exchanged to derive the secret key. Then it uses modular exponentiation to make discrete logarithms.

Then the message is sent across the wire without disclosing the secret key. In this case the attacker can see the content of the message since HMAC doesn't ensure confidentiality on its own, therefore it ensures the message can't be altered without a shared secret key. Therefore the server receives a message and has the same shared secret key that will be used to combine with the message to calculate a digest and verify if message has been changed in transit by comparing the resulting digest with received one. In this case only users that have a secret key can create an acceptable message or verify authenticity.

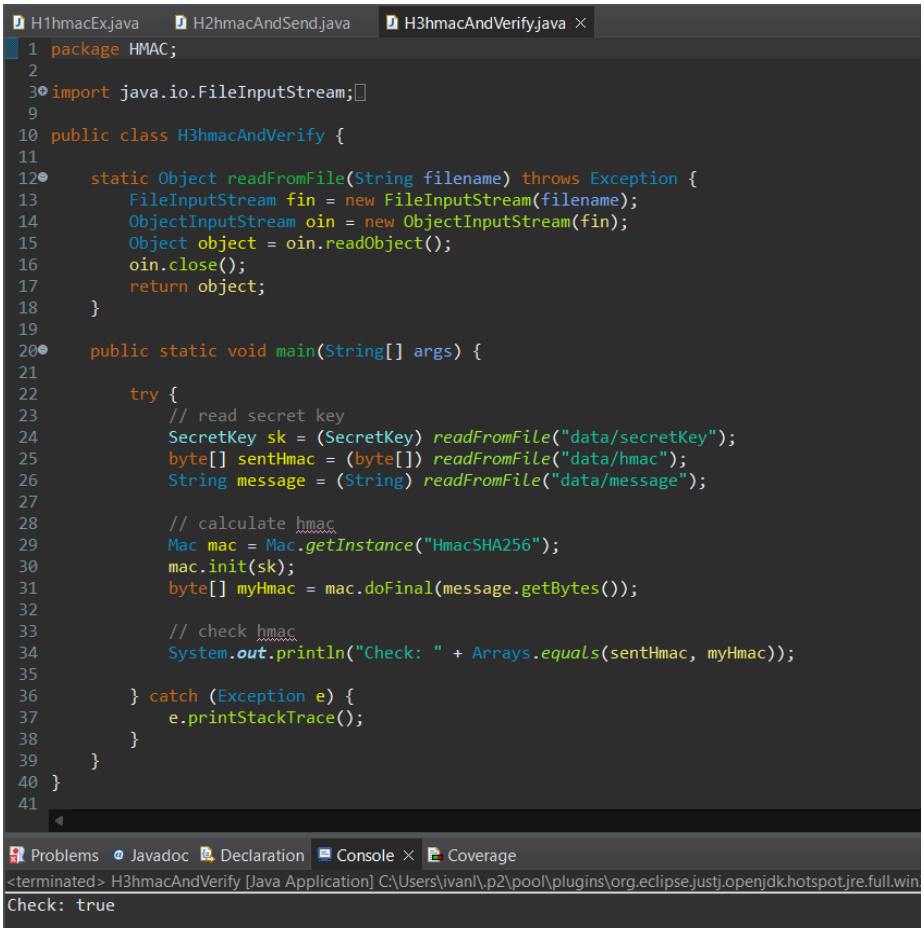

Figure 4. Illustrating Symmetric HMAC.

As in “Figure 4.” 3 classes:

H1hmacEx - generates & verifies digest with HmacSHA256.

H2hmacAndSend - generates and writes secret key, message, digest to files.

H3hmacAndVerify - reads and verifies secret key, message, digest from files.

Diffie-Hellman Key Exchange: How to Share a Secret

3. Secure Access to Web Applications (HTTPS)

Asymmetric way of accessing public api where one key kept private and one made public. Confidentiality comes first, primarily concern is authentication on the server when only a private key can decrypt a message. As shown in “Figure 5.” Below Jack used Alice's public key to encrypt a message with the RSA algorithm , then only Alice can decrypt it. Therefore even if the attacker will get Jack's private key , he won't be able to decrypt messages that Alice has.

Asymmetric Encryption - Simply explained

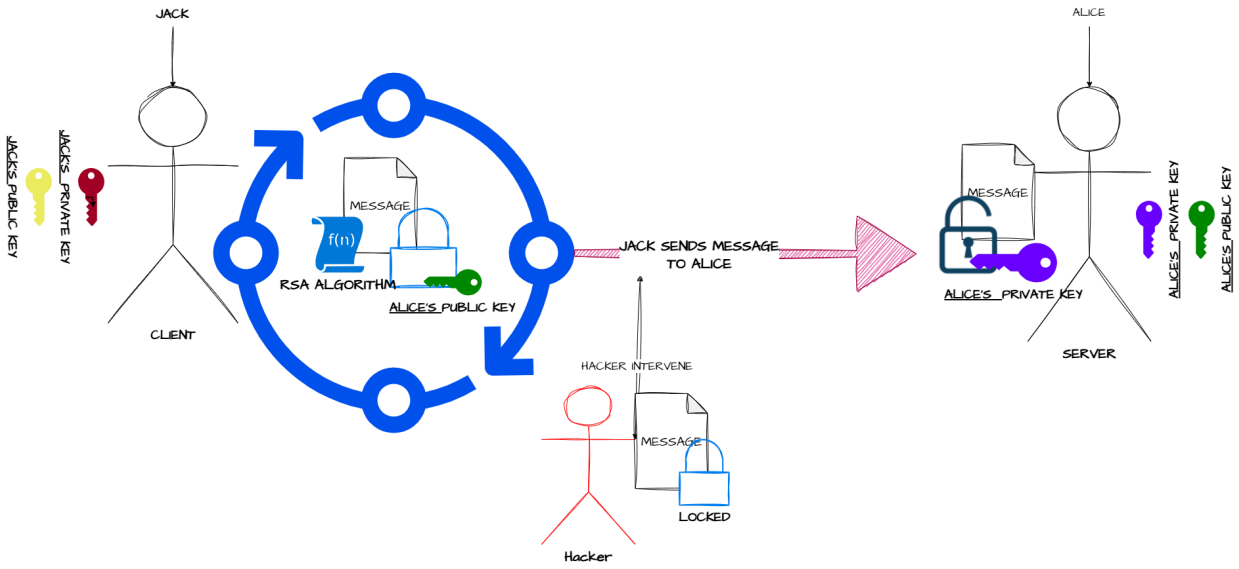

Figure 5. Illustrating Asymmetric encryption.

How Encryption Works - and How It Can Be Bypassed

7 Cryptography Concepts EVERY Developer Should Know

F1Authenticate.java F3SignAndSend.java F4ReceiveAndVerify.java

```
1  package DigitalSignatures;
2  import java.io.FileOutputStream;
9
10 public class F3SignAndSend {
11   static void writeToFile(String filename, Object object) throws Exceptic
12     FileOutputStream fout = new FileOutputStream(filename);
13     ObjectOutputStream oout = new ObjectOutputStream(fout);
14     oout.writeObject(object);
15     oout.close();
16   }
17
18   public static void main(String[] args) {
19
20     try {
21         KeyPairGenerator keyGen = KeyPairGenerator.getInstance("DSA");
22         KeyPair pair = keyGen.generateKeyPair();
23         PrivateKey privateKey = pair.getPrivate();
24         PublicKey publicKey = pair.getPublic();
25
26         String message = "This is the data I am sending";
27
28         Signature dsa = Signature.getInstance("SHA256withDSA");
29         dsa.initSign(privateKey);
30         dsa.update(message.getBytes());
31         byte[] sig = dsa.sign();
32
33         writeToFile("data/message", message);
34         writeToFile("data/signature", sig);
35         writeToFile("data/publicKey", publicKey);
36     } catch (Exception e) {
37         e.printStackTrace();
38     }
39   }
```

Problems Javadoc Declaration Console Coverage

<terminated> F4ReceiveAndVerify [Java Application] C:\Users\ivanl\.p2\pool\plugins\org.eclipse.justj.ope

signature verifies: true

Figure 6. Illustrating 3 classes and verified signatures.

As in “Figure 6.” 3 classes:

F1Authenticate: Generates a DSA key pair, signing a message with the private key, and then verifying the signature with the corresponding public key.

F3SignAndSend: Generates a DSA key pair, signs a message with the private key, and writes the message, signature, and public key to files for transmission.

F4ReceiveAndVerify: Reads the public key, signature, and message from files and verifies the message's authenticity and integrity using DSA.

4. Secure Shell (SSH)

SSH allows secure log in to machines, could be virtual or physical, allows remotely and safely log in to machines. It needs to do 2 things: authentication and confidentiality. Authentication - client side(virtual machine that client wants to log in to) - it's about a person who wants to be logged in. SSH works by having ssh server, and ssh client running on desktop.

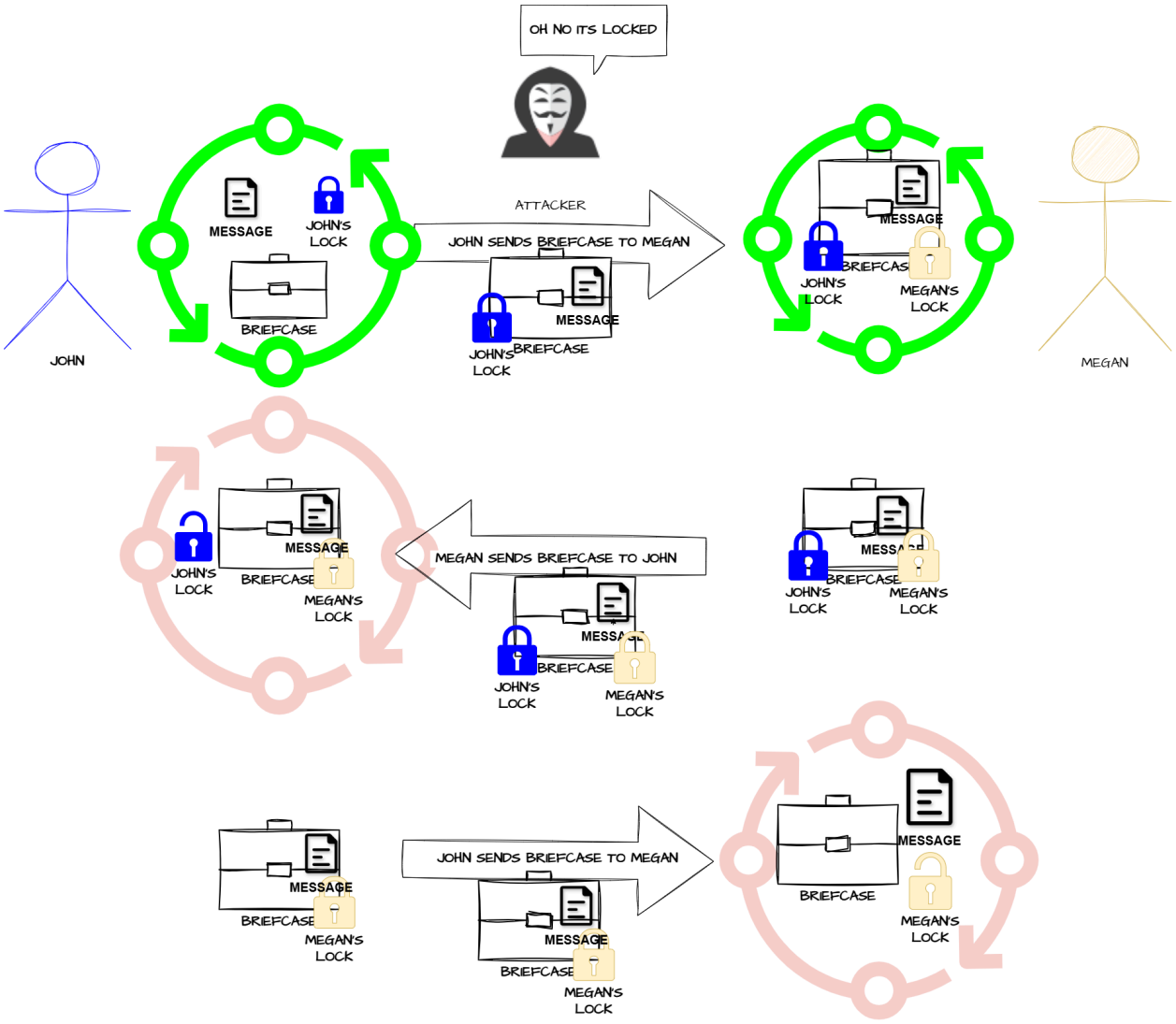

Figure 7. Illustrating usage of SSH.

Beginners Guide To SSH

There are 2 ways of authentication : password or key. Password because users have an option for password based, confidentiality has to be sent first- this is different from https. In https confidentiality first so order is different.
SSH keys - both sides have public and private keys.
Confidentiality - algorithm and look more to encryption sessions where client connects to server. Sets temporar public key, client randomly chooses session key. Same as for https. You take the session key and encrypt with both.

Why we using two keys? Because perfect forward secrecy - session key can never be generate again. If we use one key if the server is compromised then if someone obtains one of the keys they would be able to decrypt. Session key thrown away once finished. You need temp public and private key and perfect secrecy means throwing stuff away. Then it ensures the session key is not possible to regenerate.

5. Overview and Conclusion (including comparison)

- **HMAC**: Provides integrity and authentication but does not ensure confidentiality.
- **HTTPS**: Prioritizes confidentiality by using asymmetric encryption for server authentication, ensuring that only the intended recipient can decrypt the message.
- **SSH**: Emphasizes client authentication while also ensuring session confidentiality through temporary keys and perfect forward secrecy.

❖ **Authentication** can be achieved through password-based or key-based methods, with key-based methods using public/private key pairs.
❖ **Symmetric key encryption**, which uses the same key for both parties, is faster but requires secure key distribution.
❖ **Asymmetric encryption** uses distinct keys for encryption and decryption, offering higher security but at the cost of performance.

This report has examined these cryptographic methods to understand how they protect against unauthorized access and ensure secure communication in client/server environments.