Homework 2

Gabriel Alexander Pástor Villacís - 00211253

Exercise 1:

1. **Suppose a password is chosen as a concatenation of seven lower-case dictionary words. Each word is selected uniformly at random from a dictionary of size 50,000. An example of such a password is "mothercathousefivenextcrossroom". How many bits of entropy does this have?**

   Consider this entropy concept, where the implementation requires of the possible cases:

   $$E = \log_2(possible\ cases)$$

   As we select some words from a dictionary to create a password, containing there just seven words, then we apply the next formula:

   $$possible\ cases = (total\ words)^{words\ used}$$

   Finally, we replace values and get the next result:

   $$possible\ cases = (50000)^7 = 7.8125 \cdot 10^{32}$$

   $$E = \log_2(7.8125 \cdot 10^{32}) = 7 \cdot \log_2(50000)$$

   $$E = 7 \cdot 15.6096 \approx 109.2672$$

   Then, we can get 109 bits of entropy as a result.

2. **Consider an alternative scheme where a password is chosen as a sequence of 10 random alphanumeric characters (including both lower-case and upper-case letters). An example is "dA3mG67Rrs". How many bits of entropy does this have?**

   Based on the password, for an alphanumeric case character, then we have to take in count each alphabet character, as capital letter and as lower case (26 cases for each one). In addition, all numeric digits enter here (10 values). Then we have the next condition:

$$total\ values = 26 + 26 + 10 = 62$$

It occurs that there are 10 random characters, and with that we have the next elements, applying entropy:

$$E = \log_2(62^{10}) \approx 59.542$$

Then, we get for this case 59 bits of entropy.

3. **Which password is better, the one from 1. or 2.?**

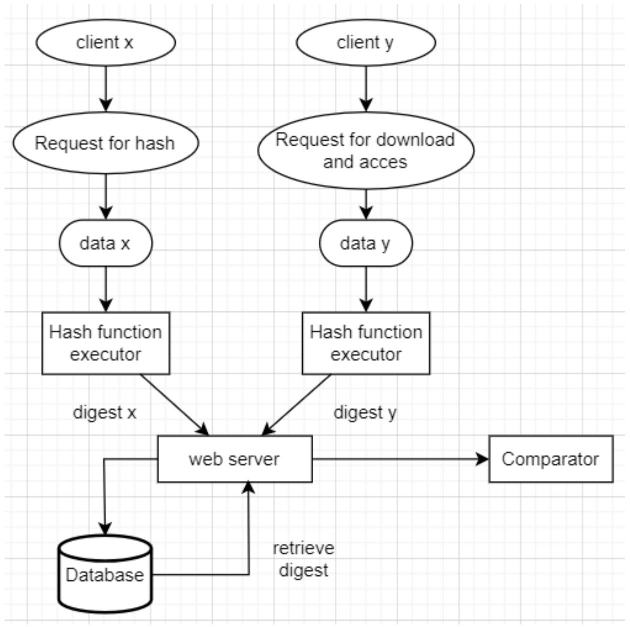
   Based on the entropy values, then we can consider the one with higher entropy as higher. In that way, the first password is better.

Exercise 2:

1. **Design a data verification system using hash functions. Explain the steps involved in the process.**

   The structure could depend on the implementation, but if we consider a specific kind of system, where the hash input applies over an input, then we have to consider two different parts of it: hash function to create a digest and hash function to verify digests.

   In this case, although verification is the main point, there should be a section where digest creation is shown and another where digest is verified.



Then we can consider the next elements:

Input: this part implies the client and its data, which is sent to the web server to be processed. It depends on the goal of each activity: one to save data and another for downloading data or access to a website mainly.

Hash function: this component is for hashing, applying a hash function to modify data, then the type of function could be anyone, such as MD5, and getting a digest value.

Web server: this the main part of all the system, where the hashing values are managed to be stored on a database or to be sent for being compared.

Comparator: this section mainly is just a section apart of the web server, where two digests are analyzed to know if they are equal, being then this part for data verification mainly.

Database: part where digest is stored, these values are applied after for data verification, giving a digest value if a data is requested.

Considering that, then we can get the next steps for this system:

1. Input: This perhaps is the most straightforward step, first, you need to have the data that you want to verify or secure. This data could be any form of digital information, from passwords and usernames to crucial files or documents.

2. Hash Function (Hashing): Following that, the “hash function” is the must-have component of this system. This function processes the input data and generates a fixed-size string of characters, which is the hash code. This hash function can be any well-known algorithm such as SHA-256 or MD5, depending on your specific requirements and security levels. The obtained hashed output should have the properties of pre-image resistance, second pre-image resistance, and collision resistance for ensured security.

3. Hash Comparison: If you are verifying data, you need something to compare the hash with. Typically, you would have a previously stored or transmitted hash, which you compare with the new hash. This step either takes place at the recipient's side for the purpose of integrity verification or on your side to verify hash values at regular intervals.

4. Hash Storage: In the system, a dedicated portion is kept reserved for storing these hashes, it could be a database or any secure form of storage. In case of a change in data, the hash changes and you can easily spot any inconsistencies or tampering with the data.

5. Output/Verification Result: The final step is the result of whether the data is verified or not. If the newly generated hash matches the stored hash, the data is verified and hasn't been tampered with. If the hashes don't match, it points towards possible data alteration and raises a flag within the system.

2. Discuss the advantages and disadvantages of using hash functions for data verification.

In this case, it is essential to consider hash functions in relation to other cryptographic methods that are applicable to data integrity.

<u>Advantages:</u>

1. Data Integrity Assurance: Hash functions are effective at detecting alterations in data by comparing the digest values of both the original and obtained data. They identify even the slightest changes in content. This means that if the values do not match, they are discarded, ensuring greater confidence in the complete data match and its integrity.
2. Unique Hash Values: It is highly unlikely for two or more different data inputs to produce the same digest value (generated by hash functions). This means that a hash collision is not an immediate concern, allowing for precise data verification, as each digest corresponds to specific data.
3. Fast and Consistent Processing: It is considered computationally efficient. Additionally, a function can perform search and modification operations in constant time, denoted as O(1). This is in comparison to other methods like MAC or digital signatures.
4. Space Efficiency: Hash functions are more efficient when compared to trees or linked lists, especially when storing distinct values. They allocate data into a fixed-length format, regardless of the original data's length.
5. Simplicity: Compared to MAC, hash functions do not require additional keys, making them easier to manage.

<u>Disadvantages:</u>

1. Data Irretrievability: Hash functions are unidirectional, although the possibility of a reverse process is speculative. Despite this, their one-way nature prevents the retrieval of original data from the generated hash. They simply guarantee data integrity but not data recovery.
2. Collision Risk: Hash functions are not collision-proof, and this could potentially lead to false positives in data verification. Analyzing the digest in each case may be necessary.
3. Dependency on Hash Function: While it may seem advantageous, in reality, a data verification system heavily depends on the type of hash function used. A weak function implies vulnerability to attacks.
4. Lack of Order: The order of elements is not preserved, which can be problematic if order is crucial. This is a significant consideration when determining whether to use hash functions when data sequence is important.
5. Limited Capacity: Storage capacity is not extensive, as hash tables have limited space and can become full at any time. Additionally, the additional cost associated with resizing storage to accommodate more information is a factor.
6. Increased Complexity: Although hash functions improve data verification, they also increase the complexity of data structures by requiring management of potential collisions.
7. Insecurity: In comparison to methods like MAC and digital signatures, hash functions can be insecure due to collision vulnerabilities, lack of order preservation, and limited storage capacity.

3. **Provide an example of a real-world application where a data verification system using hash functions is used.**

   The implementation of data verification using hash functions can be found in various scenarios, considering software distribution and version control.

   In the case of software distribution, software providers apply hash values, generated using cryptographic hash functions like SHA-256, giving an additional layer of security. Users who download the software receive both the software itself and its respective hash value. After that, users calculate the digest (hash value) of the downloaded file and compare it with the provided hash value. The matching of these values indicates with high probability that the downloaded software is genuine and hasn't been altered during transmission, ensuring the integrity of the software.

For the case of Git, hash values are fundamental for tracking and managing versions of files and projects. Each time a change or update is made to the source code, Git generates a unique digest, known as the "commit hash," using the SHA-1 hash function. This hash value becomes a unique identifier for that specific version of the content, helping for the management of different versions and changes within a software development project.

The importance of this is for ensuring data integrity. When files or data are retrieved from a Git repository, Git's software recalculates the hash value of the downloaded files and compares it to the stored hash value. If both values match, it can be confidently stated that the data hasn't been altered, preserving data integrity throughout the process.

Exercise 3:

1. **Define what a Message Authentication Code (MAC) is and how it is used in cryptography.**

   A Message Authentication Code (MAC), also known as a tag, is a piece of information used to authenticate the origin and integrity of a message. In cryptography, the MAC plays a crucial role in confirming the authenticity of data transmitted over a network or shared with users. Essentially, it involves protection against message tampering, ensuring that the message is authentic and from the correct source (providing authenticity), remains unaltered during its transmission (preserving integrity), and is valid.

   The MAC process begins by creating a secure communication channel between the sender and the receiver. To secure a message, an algorithm is used, incorporating an additional key and the message for transmission. Subsequently, this MAC algorithm processes the message, generating fixed-length authentication tags, constituting the MAC of the message. It is attached to the message and sent to the receiver. Upon receipt, the recipient applies the same algorithm to calculate the MAC. If the calculated MAC matches the one sent by the sender, the message is authenticated as genuine, legitimate, and unaltered.

   The methods of implementation vary, including One-time MAC, VMAC (involving the use of two shared keys with random hash functions), and HMAC (applying hashing combined with a secret key and using XOR). Among them, HMAC, KECCAK Message Authentication Code (KMAC), and Cipher-based Method

Authentication Code (CMAC) stand out as generally approved and versatile MAC algorithms.

It's crucial to emphasize that, although MAC functions share similarities with hash functions, they have different security requirements. For a MAC function to be considered secure, it must effectively withstand existential forgery under chosen-message attacks, ensuring robust protection against fraudulent or malicious activities.

2. **Explain the process of generating and verifying a MAC.**

   MAC Creation and Verification Process:

   1. Secret Key Generation: The initial step in generating a Message Authentication Code (MAC) involves creating a confidential key. This key remains exclusive to both the message sender and receiver. The security of the entire MAC system heavily relies on the secrecy of this key.
   2. MAC Generation: After generating the secret key, it is utilized in combination with a cryptographic algorithm to produce the MAC itself. The most common approach is to employ a block cipher such as DES or AES, or a cryptographic hash function like SHA-256. Both the secret key and the message serve as input to this algorithm. For instance, when using a hash function like H, the MAC generation might appear as follows:
      Python: MAC = H(key || message). Here, "||" signifies concatenation.
   3. Appending MAC to Message: Subsequently, the generated MAC is affixed to the end of the message. This amalgamated 'message + MAC' is then dispatched to the recipient.
   4. MAC Verification: Upon receiving the 'message + MAC,' the receiver applies the same secret key and cryptographic algorithm to calculate the MAC for the received message. The calculated MAC is subsequently compared with the received MAC.
   5. If both MACs correspond, it signifies the preservation of integrity and authenticity. In case of a mismatch, it indicates that either the message was altered during transmission or that the identity of the sender could not be verified.
   6. It's essential to underscore that while MAC offers integrity and authenticity, it does not ensure confidentiality. If message confidentiality is a requisite, the inclusion of encryption alongside MAC should be considered.

7. Additionally, it's vital to keep in mind that key management, which encompasses secure key generation, distribution, storage, and retirement, is a critical component in any cryptographic system.

3. **Discuss the importance of using MACs in secure communication systems.**
   Message Authentication Codes (MACs) play a crucial role in secure communication systems. Consider the next points for it:
   1. Preservation of data integrity: MACs play an important role maintaining data integrity for secure communication. This applies to ensure that data has not suffer unauthorized changes during its transmission. This is achieved by including a MAC code with the message. By calculating an identical MAC using the shared key, the recipient can compare it with the message's MAC. A match indicates data integrity, while a lack of match suggests potential alterations.
   2. Confirmation of authenticity: It involves verifying that a message originates from the legitimate source, not from an attacker or an intermediate. MACs confirm this by enabling the recipient to verify that the message comes from the authorized sender. This is particularly relevant for thwarting impersonation, an attack where an attacker pretends to be a legitimate sender. MACs' ability to confirm authenticity plays a crucial role in establishing trust in online communications.
   3. Protection Against Malware: MACs also offer defense against malware, virus and other forms of malicious code. MACs can detect attempts to introduce unauthorized code into the system. This is important for an environment where malware can cause substantial damage to systems and networks.
   4. Prevention of replays: Replay attacks are a common issue in online communication. In these attacks, a user intercepts and replays a valid data transmission at a later time. MACs can defend against this by including unique message information in the MAC calculation. If a message is replayed, the MAC will not match the original, allowing for the detection and prevention of replay attacks.
   5. Non-Repudiation: In specific contexts, such as financial or legal transactions, the ability to prove that a message was sent by a specific party is essential. MACs can provide non-repudiation by uniquely linking a message to the sender through the use of the shared key. This prevents the sender from denying sending the message, which is critical in situations where authenticity and accountability are paramount.

Exercise 4:

Given the values of p = 17 and q = 23, generate a pair of keys for RSA.

The, we have the next process, for n value:

$$\mathrm{n} = \mathrm{p} \cdot \mathrm{q} = 17 \cdot 23 = 391$$

Then, we calculate the $\varphi(\mathrm{n})$ value, considering them as prime numbers:

$$\varphi(\mathrm{n}) = (\mathrm{p} - 1) \cdot (\mathrm{q} - 1) = (17 - 1) \cdot (23 - 1) = 16 \cdot 22 = 352$$

For that, we have then to choose e, a positive integer such that 1 < e < 352 and gcd(e, 352) = 1.

For this case, we can consider a common value like 3, then we define the next process, for e=3:

$$\mathrm{d} \equiv \mathrm{e}^{-1} \bmod \varphi(\mathrm{n})$$

$$\mathrm{d} \equiv 3^{-1} \bmod 352$$

Calculating the modular inverse of 3 mod 352, considering and intermediate value a q:

$$q = \frac{352}{3} \cong 117$$

Then, consider the Extended Euclidean Algorithm:

$$r = 352 - (117 \cdot 3) = 1$$

$$d = 0 - (117 \cdot 1) = -117$$

As we get those values, we could say that the gcd is 1, but then for d the difference between -117 and 352 could help to define a positive value.

$$d = -117 + 352 = 235$$

Then, we have the pairs for this case:

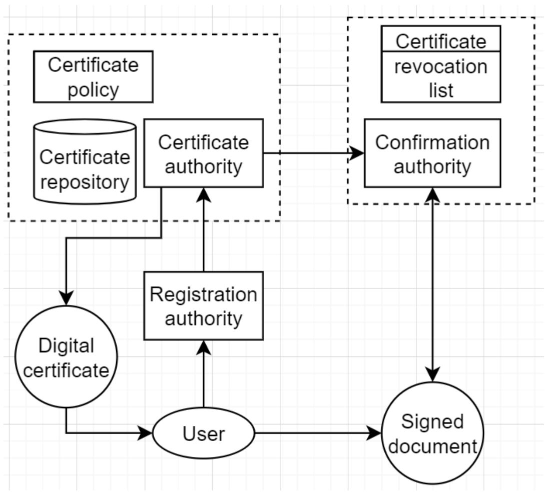
Public Key: (n, e) = (391, 3)

Private Key: (n, d) = (391, 235)

Exercise 5:

1. **Design a public key infrastructure (PKI) system. Explain the components and their roles in the system.**

   As PKI implies a system that is not simply programmed and stablished by software, it could occur that some other elements could be part of it, and not just hardware, also company rules and other requisites could be part of the system. Then we have the next elements.



   If that could be part of the system design, then we have the next components:

   Certificate Authority (CA): The CA is the core component in a PKI system. It issues and manages digital certificates, which bind public keys to entities (e.g., individuals, organizations).

   Responsibilities:

   - Verifying the identity of certificate applicants.
   - Creating and signing digital certificates.
   - Revoking certificates if compromised or expired.

   Registration Authority (RA): The RA acts as an intermediary between the user and the CA.

   Responsibilities:

   - Verifying the identity of certificate applicants.

- Submitting certificate requests to the CA.
- Handling administrative tasks related to user registration.

Certificate Repository: This is a central repository that stores issued certificates and makes them available to the public.

Responsibilities:

- Providing a way for users to access and retrieve public keys.
- Ensuring that certificates are up-to-date and accessible.

End Entities (Users and Devices): End entities include individuals, organizations, and devices that use digital certificates for secure communication.

Responsibilities:

- Generating key pairs (public and private keys).
- Requesting digital certificates from the CA.
- Using their private keys to encrypt and sign data.

Public Key Infrastructure (PKI) Policy and Practice Statements: These are documents that define the rules, practices, and standards that the PKI follows.

Responsibilities:

- Setting out the criteria for certificate issuance.
- Describing the security measures in place.
- Defining roles and responsibilities within the PKI.

Certificate Revocation List (CRL): The CRL is a regularly updated list of revoked certificates.

Responsibilities:

- Providing information about certificates that are no longer valid.
- Allowing relying parties to check the status of certificates before trusting them.

Confirmation Authority: This part implies verification of certificates and the analysis of them to define how trustful it is.

Responsibilities:

- Verifying the validity of a digital certificate.
- Providing guarantee that the certificate was issued by a trusted authority certification.

Then, for this system we can define the next points.

1. An End Entity (User or Device) generates a key pair and requests a digital certificate from the Registration Authority (RA).

2. The RA verifies the entity's identity and forwards the certificate request to the Certificate Authority (CA).

3. The CA verifies the request and issues a digital certificate, signing it with the CA's private key.

4. The issued certificate is stored in the Certificate Repository, making it accessible to the public.

5. Relying parties (e.g., users, devices) can retrieve the public keys from the Certificate Repository to establish secure communication.

6. The CA also maintains a Certificate Revocation List (CRL) to keep track of revoked certificates, allowing users to check their validity.

7. The PKI Policy and Practice Statements provide guidance and rules for the entire PKI system.

8. Also, the Confirmation Authority can define the validation of a signed document, ensuring that a digital certificate was issued by a trustful authority.

2. **Discuss the advantages and challenges of implementing a PKI system.**

   Advantages:

   <u>Flexibility and Control:</u> Having complete control over the certificate issuance process and unrestricted access to cloud resources are advantages of its implementation. Furthermore, it allows for customized solutions based on the organization's requirements and capabilities.

   <u>Autonomy:</u> This gives the organization the ability to make internal configurations according to its own needs and schedule, considering it is crucial for maintaining important data within the system.

Enhanced Security: It is more secure than a password-based system. Public Key Infrastructure (PKI) is a solid choice for a high level of security in communications due to its simpler management compared to passwords, which have known vulnerabilities. Multifactor authentication (MFA) is one of the security techniques used in the context of PKI.

Disadvantages:

Hidden Costs: Taking into account the hidden costs associated with on-premises PKI, which go beyond the initial hardware investment. These additional expenses cover backup and disaster recovery procedures, certificate lifecycle management, software acquisition and maintenance, and staff training. When implementing on-site PKI, these additional costs can accumulate, especially in the context of security and compliance audits.

Ongoing Maintenance: Companies that choose on-premises solutions require continuous maintenance. This involves ongoing management of physical space, power usage, and server hardware. The PKI system, including cryptographic key management, must be continuously administered to ensure effective and secure operation.

Company Responsibility for Data Security: Emphasizing organizational responsibility when assessing data security, especially for businesses subject to regulations like HIPAA. This is why companies need to manage security and access policies to maintain compliance with laws and constant control over where data is stored.

Need for Backup in Case of Data Loss: The need for a reliable backup system is highlighted. An organization runs the risk of suffering permanent data loss if it does not have a dependable backup system. This entails implementing robust data recovery procedures, such as security incident management.

In conclusion, the text provides an in-depth analysis of the benefits and challenges of establishing an on-premises public key infrastructure, with a special focus on terms related to cybersecurity and cryptographic key management.

3. **Provide an example of a real-world application where a PKI system is used.**

PKI involves security in public internet traffic, which is why it's already integrated with all web browsers. This allows for authenticating and securing communication between the web browser and web server.

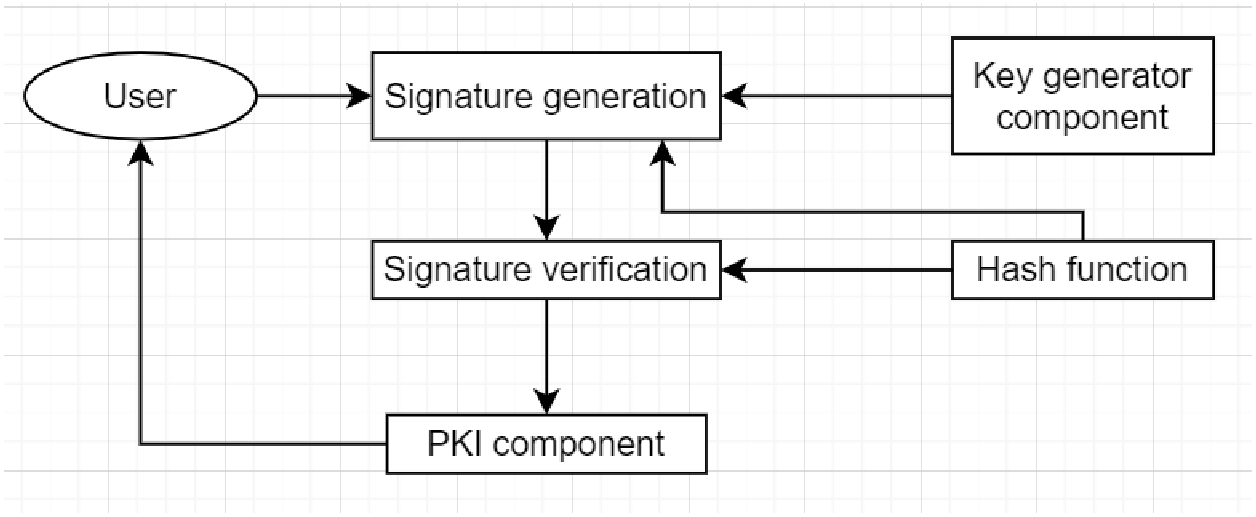
Here's how it works:

1. Digital Certificates: A digital certificate cryptographically links a public key with the device or user who owns it. This helps authenticate users and devices and ensures secure digital communication. Digital certificates are issued by a trusted source, a Certificate Authority (CA), and serve as a kind of digital passport to verify that the sender is who they claim to be.
2. Public and Private Keys: It involves both a private key and a public key. The public key is available to anyone who needs it and is used to encrypt a message sent to you. The private key is used to decrypt the message after you receive it. These keys are linked by a complex mathematical equation. Despite their relationship, it's extremely challenging to determine the private key using data from the public key.
3. Certificate Authority (CA): The CA is a trusted entity that issues, stores, and signs the digital certificate. The CA signs the digital certificate with its private key and then publishes the public key, which can be accessed upon request.
4. Registration Authority (RA): The RA verifies the identity of the user or device requesting the digital certificate. This can be a third party, or the CA can also act as the RA.
5. Certificate Database: This database stores the digital certificate and its metadata, including its validity period.
6. Central Directory: This is the secure location where cryptographic keys are indexed and stored.
7. Certificate Management System: This system is used to manage the issuance of certificates and control access to them.

This process ensures that when you connect to a website, your browser can verify that it's connecting to the correct server, not an impostor. It also ensures that any data you send to that server is encrypted during transmission, protecting it from potential eavesdroppers.

Exercise 6:

Design a system for digital signatures based on public-key cryptography. Explain the steps involved in the process and the role of each component.



A digital signature system based on public-key cryptography involves several components working together to ensure the integrity and authenticity of electronic documents. Here are the main steps involved in the process and the role of each component:

1. Key Generation:

    - Public Key: A public-private key pair is generated by the signer. The public key is shared with others and used to verify the digital signature.
    - Private Key: The private key is kept securely and used by the signer to create the digital signature.

2. Signature Generation:

    - Hash Function: A cryptographic hash function is applied to the message or document to create a fixed-size hash value.
    - Private Key Operation: The private key is used to encrypt the hash value, creating the digital signature.

3. Signature Verification:

    - Hash Function: The same hash function used in the signature generation is applied to the received message or document to produce a hash value.
    - Public Key Operation: The digital signature received is decrypted using the signer's public key, yielding the original hash value.
    - Comparison: The computed hash value and the decrypted hash value are compared. If they match, the signature is verified to be authentic and the message has not been tampered with.

Now, let's describe the components and their roles more deeply:

1. Key Generation Component:

    - Purpose: This component generates the public-private key pair.
    - Responsibilities: It generates a strong random private key and derives the corresponding public key.

2. Signature Generation Component:

    - Purpose: This component creates the digital signature for a given message or document.
    - Responsibilities: It applies a hash function to the message or document to generate the hash value, and uses the private key to encrypt the hash value, creating the digital signature.

3. Signature Verification Component:

    - Purpose: This component verifies the authenticity of a digital signature and ensures the integrity of the message or document.
    - Responsibilities: It applies the same hash function to the received message or document to produce the hash value. Then, it uses the signer's public key to decrypt the received digital signature, obtaining the original hash value. Finally, it compares the computed hash value with the decrypted hash value to determine if the signature is valid.

4. Hash Function:

    - Purpose: This is a one-way function that takes an input and produces a fixed-size hash value.
    - Responsibilities: It ensures that even a small change in the input will result in a significantly different hash value. The hash function used must be secure and resistant to collisions.

5. Public Key Infrastructure (PKI):

    - Purpose: PKI is a system that manages and verifies the public-key digital certificates used in the digital signature process.
    - Responsibilities: It issues digital certificates to users, binds their public keys to their identities, and provides a trusted framework for verifying the authenticity of public keys.

The key generation component generates the key pair, the signature generation component creates the digital signature, and the signature verification component verifies the signature's authenticity. The hash function ensures data integrity, and the PKI provides a trusted infrastructure for managing and verifying public keys. This system is based on the topics mentioned in previous exercises.

Bibliography:

¿Qué es una pki? Infórmese sobre ESTA tecnología aquí. Entrust. (n.d.). https://www.entrust.com/es/resources/certificate-solutions/learn/what-is-pki

Academic.oup.com. (n.d.-a). https://academic.oup.com/book/26672/chapter-abstract/195441533?redirectedFrom=fulltext&login=false

Academic.oup.com. (n.d.). https://academic.oup.com/cybersecurity/article/7/1/tyab025/6470936?login=false

Alvaro. (2021, August 25). *Funciones Hash Criptográficas Y HMAC*. Just Cryptography. https://justcryptography.com/funciones-hash-criptograficas-y-hmac/

Alvinashcraft. (n.d.). *Mac, hash y firmas - UWP applications*. UWP applications | Microsoft Learn. https://learn.microsoft.com/es-es/windows/uwp/security/macs-hashes-and-signatures

Blanton, M. (1970, January 1). *Message authentication codes*. SpringerLink. https://link.springer.com/referenceworkentry/10.1007/978-0-387-39940-9_1483

Hash functions: Theory, attacks, and Applications - Stanford University. (n.d.). https://crypto.stanford.edu/~mironov/papers/hash_survey.pdf

Krawczyk H., Bellare M., and Canetti R. HMAC: Keyed-hashing for message authentication, RFC 2104. Internet Engineering Task Force (IETF), 1997.

Paar, C., & Pelzl, J. (1970, January 1). *Message authentication codes (macs)*. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-642-04101-3_12

Public-key cryptography standards: PKCS - arxiv.org. (n.d.-b). https://arxiv.org/pdf/1207.5446v1.pdf

Springer International Publishing. (n.d.). *Public-key cryptography – PKC 2022*. SpringerLink. https://link.springer.com/book/10.1007/978-3-030-97121-2

Wu, Y., Zhang, X., Teng, Y., Liu, Z., Huang, L., Lin, J., & Bao, X. (1970, January 1). *Recent advances in the web PKI and the technical challenges in SCMS*. SpringerLink. https://link.springer.com/chapter/10.1007/978-3-030-80851-8_11