# TASK 1.1

A screenshot of a computer

Description automatically generated

# TASK 1.2

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer error message

Description automatically generated

A screenshot of a computer screen

Description automatically generated

A screenshot of a computer error message

Description automatically generated

# TASK 1.3

The time required for an attack increases exponentially with the length of the key. Longer key lengths result in a significant increase in the time needed for the attack to succeed.

TASK 1.4  
Yes, the time required for the collision search task (attack) is influenced by the choice of the hash function. Longer times are observed for MD2 and SHA, slightly shorter for SHA-1, and the shortest for MD4 and MD5.

# TASK 1.5

When dealing with a single document, finding a collision involves searching for a specific hash generated by the hash function, making the probability of success extremely low. Conversely, when working with two documents, the task becomes simpler as the goal is to find two identical hashes without specifying which particular hash is sought. The focus is on achieving the same hash result for both documents rather than targeting a specific hash, making the process more feasible

TASK 1.6  
Although hash functions are prone to collisions (they are robust until a collision occurs), their probability of occurrence is low enough to ensure security. However, certain functions, such as MD2, MD5, and SHA-1, have raised security concerns and are advised against. Currently, hash functions from the SHA-2 family (SHA-224, SHA-256, SHA-384, SHA-512) or SHA-3 (Keccak) are considered secure. While they don't guarantee the absence of collisions—generating a unique 256- or 512-bit string for all possible inputs is impossible—increasing the hash length significantly reduces the probability of collisions, making them resilient against most attacks.

TASK 1.7  
Popular hash functions like MD5 and SHA have faced significant issues primarily due to the short length of the generated hashes. MD5 produces 128-bit hashes, while SHA-1 generates 160-bit hashes, making them susceptible to collisions and potential content forgery. By 2005, efficient methods were developed to find collisions within a minute for these functions. In 2008, a vulnerability in MD5 allowed the forging of SSL certificates accepted by all browsers. SHA-1 experienced successful attacks in 2004, and in 2017, Google demonstrated the generation of two different PDF files with the same SHA-1 hash.

The brevity of these hashes also facilitates relatively fast ordinary brute force attacks. To address these vulnerabilities, it is now recommended to use more secure hash functions such as SHA-2 or SHA-3 (Keccak). The longer hashes produced by these functions significantly enhance resistance to both brute force and collision attacks.

# TASK 2

A screenshot of a computer error

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer screen

Description automatically generated

It is evident here that the shared secrets are identical.

# TASK 2.1

Presently, the Diffie-Hellman algorithm is deemed more secure, albeit under the condition of utilizing keys of equivalent length for comparison. In the realm of key lengths, 1024-bit keys in the DH algorithm exhibit greater robustness than RSA keys of matching length. Notably, RSA key generation is substantially more time-consuming compared to DH, and RSA displays lower resilience against attacks.

Nevertheless, an inherent limitation of the DH algorithm lies in its inability to authenticate the authors of a message, rendering it unsuitable for electronic signatures. Nonetheless, the DH algorithm can be synergistically employed with other algorithms, such as DSS, to establish the authenticity and integrity of a message. This collaborative approach circumvents the need for the time-intensive RSA method. Consequently, a definitive answer to the question of which algorithm guarantees higher security—RSA or DH—is elusive. The security of each algorithm hinges on the specific context in which it is applied.

TASK 2.2  
As previously stated, the DH protocol lacks a mechanism for authenticating the parties engaged in the exchange. This vulnerability opens the door to potential man-in-the-middle attacks, where an adversary can effectively intercept and substitute messages between the parties without their awareness. A prevalent form of such an attack involves providing the sender with their own key surreptitiously. The inability to confirm the key's origin empowers the intruder to clandestinely access all transmitted data without the need to compromise the cipher.

# TASK 2.3

In addition to the methods mentioned earlier, various alternatives exist for establishing a common cryptographic key. Elliptic Curve Cryptography is utilized in protocols such as ECDH (Elliptic Curve Diffie-Hellman) and ECDHE (Elliptic Curve Diffie-Hellman Ephemeral). Another viable method involves employing a trust network, where each participant in the network signs the keys of individuals they have personally verified. This signature serves as an endorsement, signifying the signer's confidence in the key's authenticity and its rightful ownership by the declared individual.

Furthermore, a password-authenticated key agreement is another avenue, where one of the communicating parties possesses partial knowledge of a shared password. A noteworthy recent development is quantum key distribution, which leverages the principles of quantum mechanics. In this method, the act of observing or measuring the state of a photon induces detectable perturbations, facilitating the identification of potential man-in-the-middle attacks.  
  
TASK 2.4  
The DH protocol safeguards security by not transmitting a crucial element between clients: the secret key of each party engaged in communication. These secret keys play a pivotal role in generating public keys. The ultimate shared key results from intertwining the private keys with the respective public keys and is identical for both parties. The protocol's reliance on large prime numbers for key generation adds a layer of security by rendering the reversal of key generation operations practically impossible. Efficient forms of number factorization are currently non-existent, contributing significantly to the robustness of the transmission. Consequently, even if a potential intruder manages to eavesdrop on the communication channel, reconstructing the key remains an insurmountable challenge.  
  
TASK 2.5  
As shown earlier, it was possible to establish the same key for both clients.

A screenshot of a computer screen

Description automatically generated