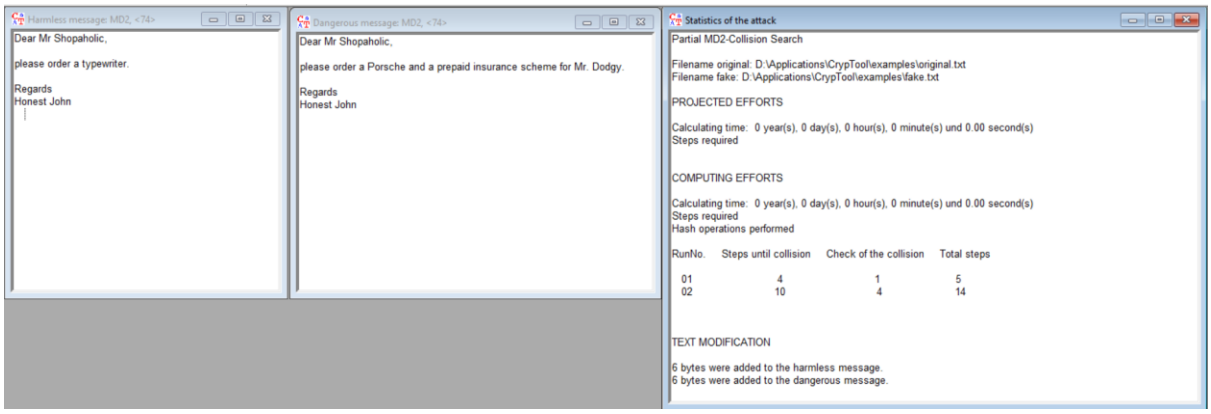
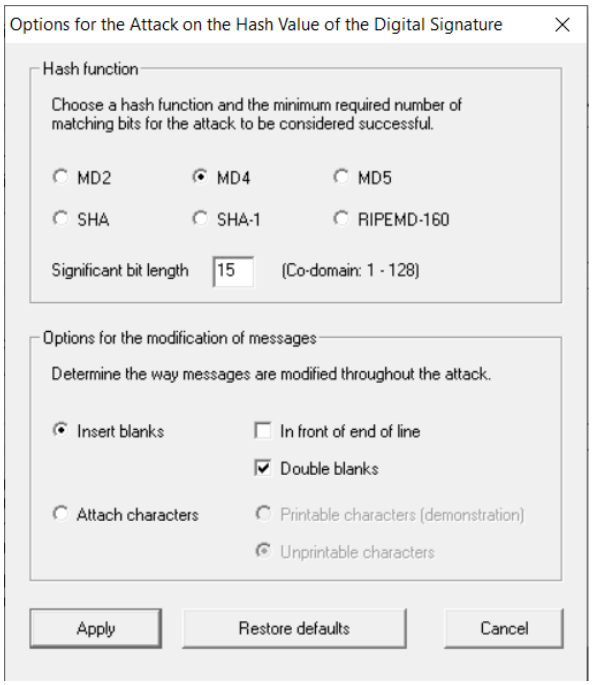
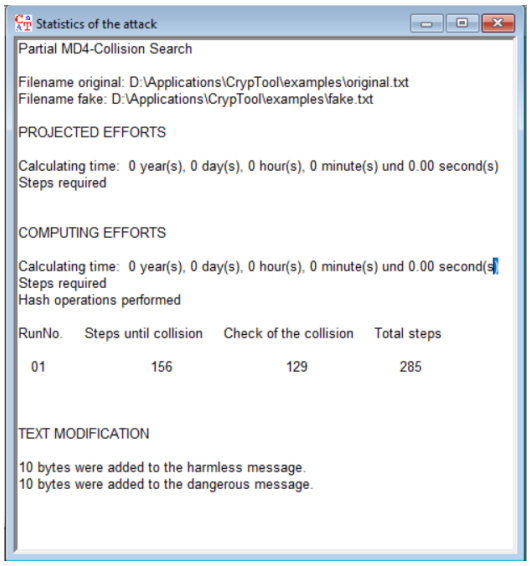
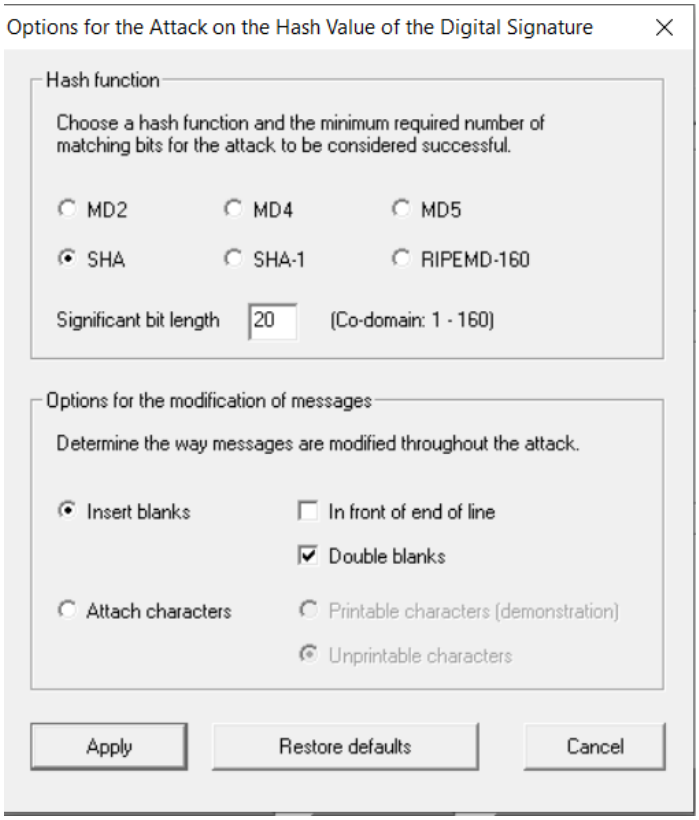
TASK 1.1

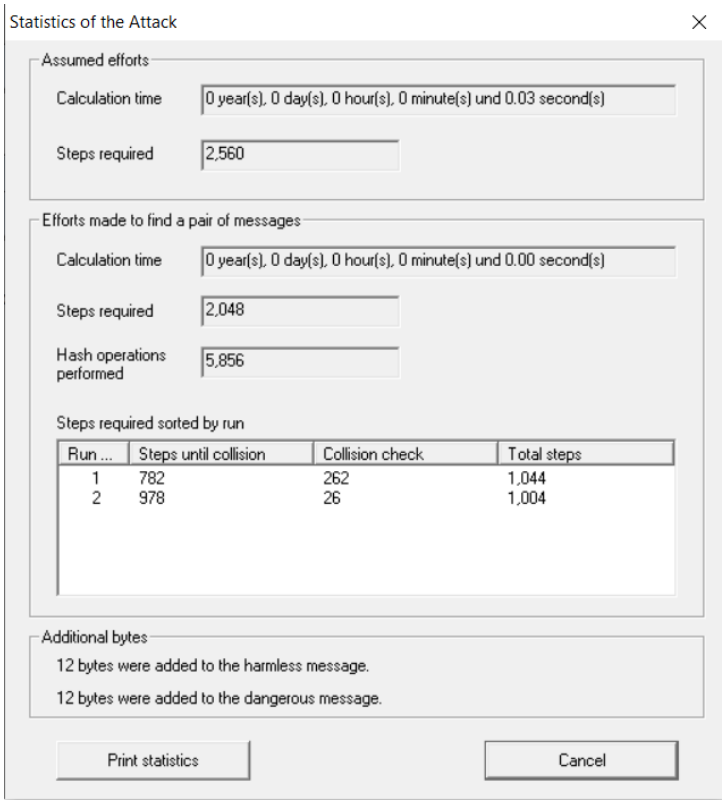


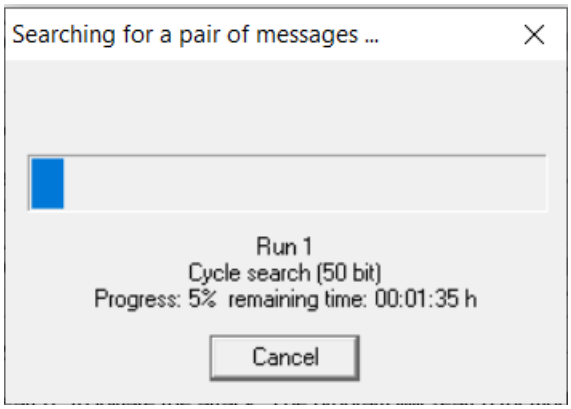
TASK 1.2

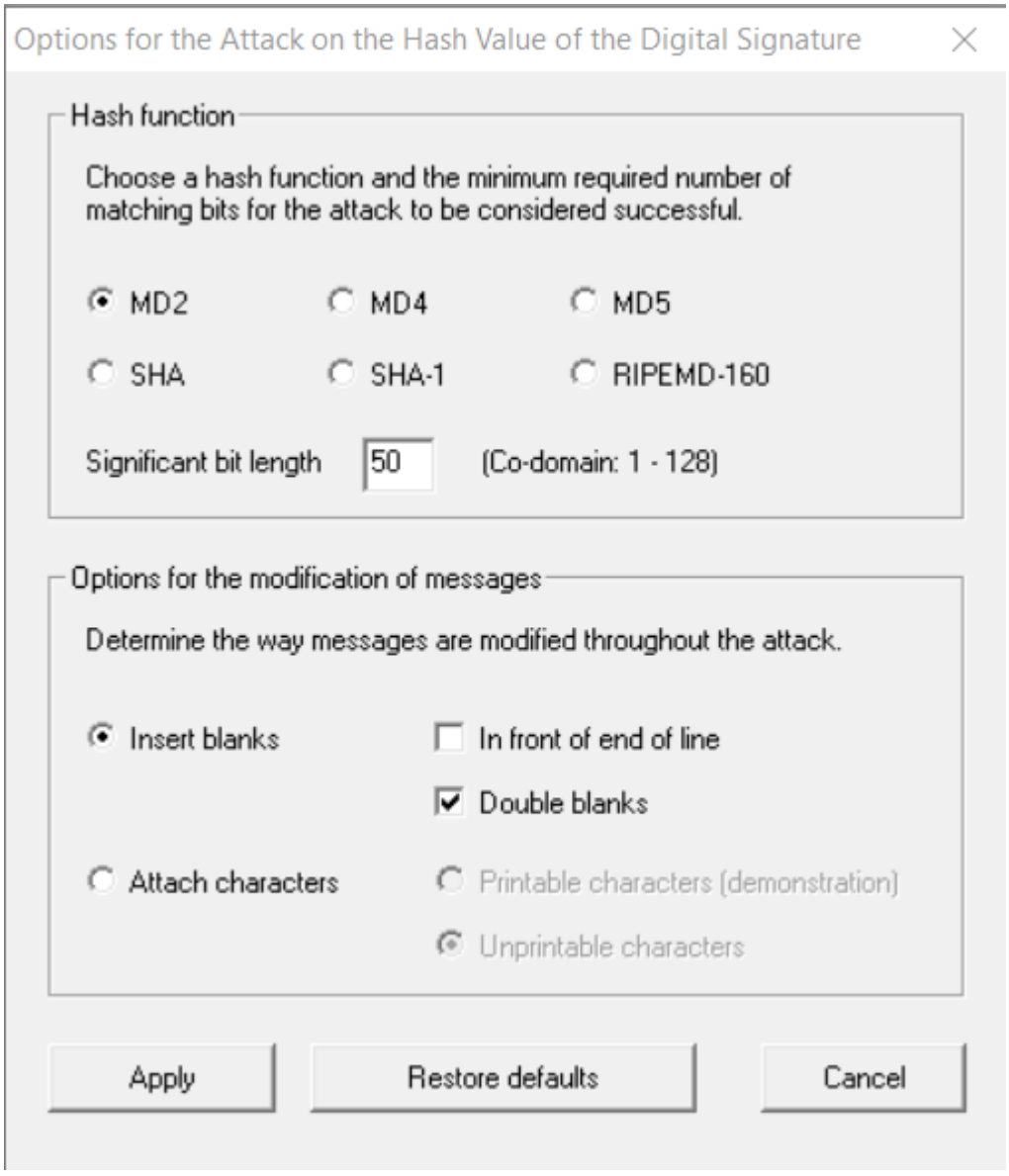


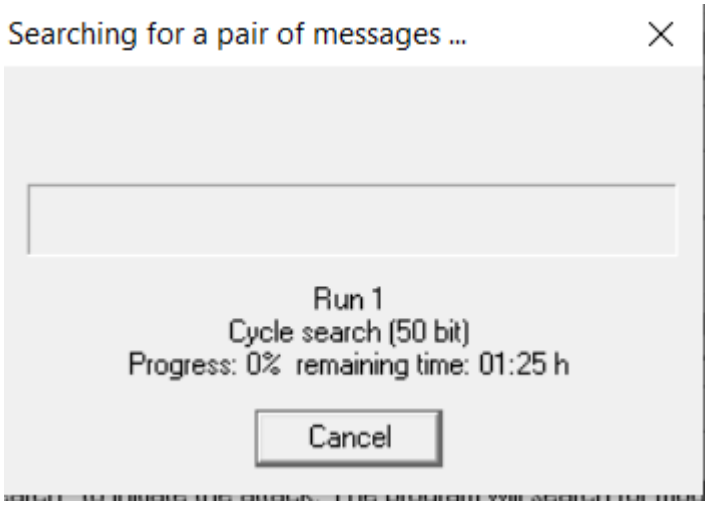



Options for the Attack on the Hash Value of the Digital Signature
Hash function
Choose a hash function and the minimum required number of matching bits for the attack to be considered successful.
MD2
MD4
MD5
SHA
SHA-1
RIPEMD-160
Significant bit length
20
(Co-domain: 1 - 160)
Options for the modification of messages
Determine the way messages are modified throughout the attack.
Insert blanks
In front of end of line
Double blanks
Attach characters
Printable characters (demonstration)
Unprintable characters
Apply
Restore defaults
Cancel


Statistics of the Attack
Assumed efforts
Calculation time
0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.03 second(s)
Steps required
2,560
Efforts made to find a pair of messages
Calculation time
0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.00 second(s)
Steps required
2,048
Hash operations performed
5,856
Steps required sorted by run
Run ...
Steps until collision
Collision check
Total steps
1
782
262
1,044
2
978
26
1,004
Additional bytes
12 bytes were added to the harmless message.
12 bytes were added to the dangerous message.
Print statistics
Cancel


Searching for a pair of messages ...
Run 1
Cycle search (50 bit)
Progress: 5% remaining time: 00:01:35 h
Cancel


Options for the Attack on the Hash Value of the Digital Signature
Hash function
Choose a hash function and the minimum required number of matching bits for the attack to be considered successful.
MD2
MD4
MD5
SHA
SHA-1
RIPEMD-160
Significant bit length
50
(Co-domain: 1 - 128)
Options for the modification of messages
Determine the way messages are modified throughout the attack.
Insert blanks
In front of end of line
Double blanks
Attach characters
Printable characters (demonstration)
Unprintable characters
Apply
Restore defaults
Cancel



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

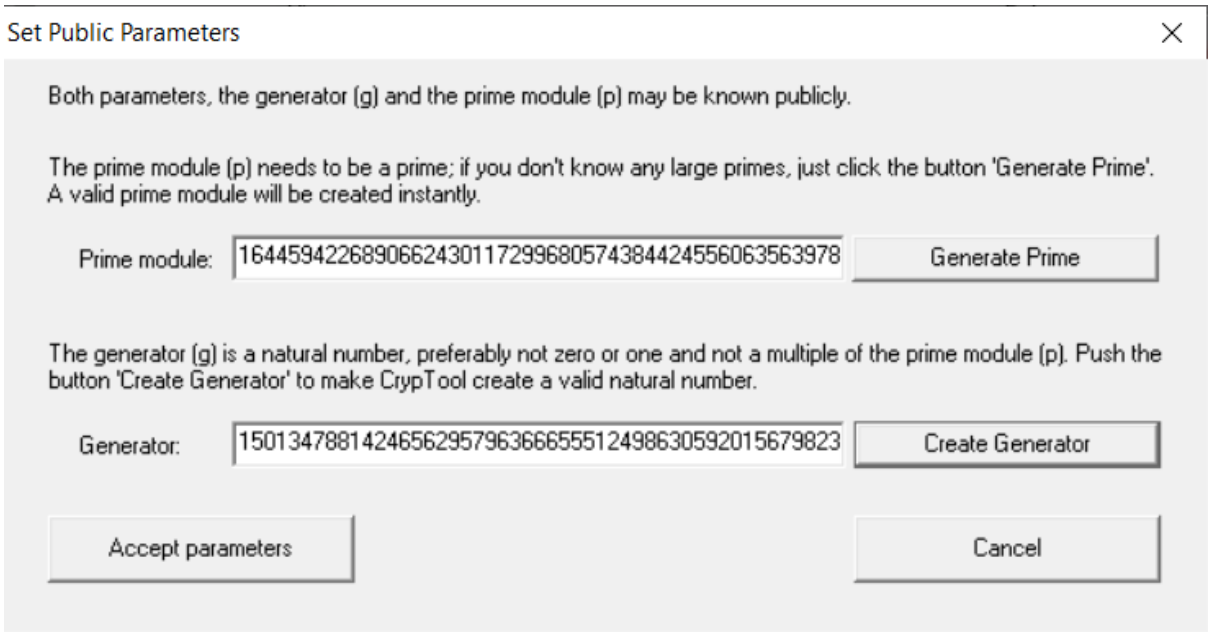
Although hash functions are prone to collisions (they are robust until a collision occurs), their probability of occurrence is low enough to ensure security. However, certain functions, such as MD2, MD5, and SHA-1, have raised security concerns and are advised against. Currently, hash functions from 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.

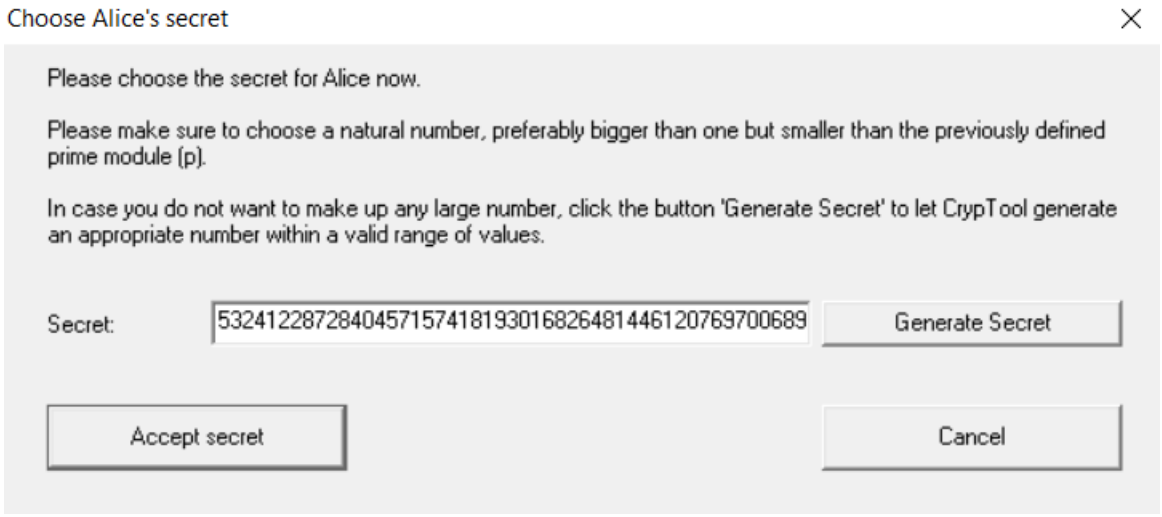
TASK 1.7

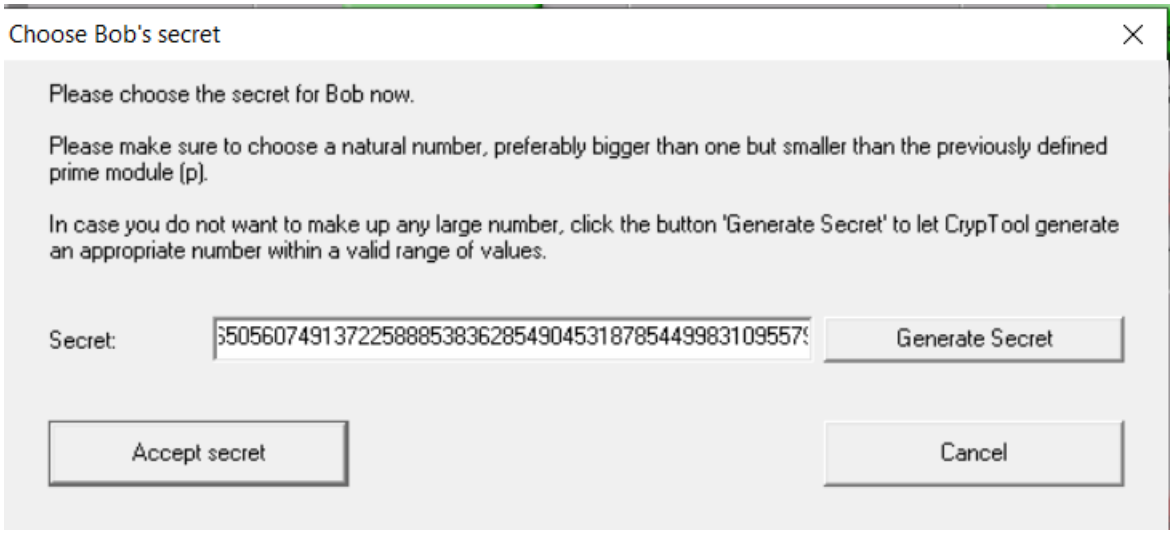
Popular hash functions like MD5 and SHA have faced significant issues primarily due to the short length of the generated hashes. MD5 produces 128-bit hashes, while SHA-1 generates 160-bit hashes, making them susceptible to collisions and potential content forgery. By 2005, efficient methods were developed 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.

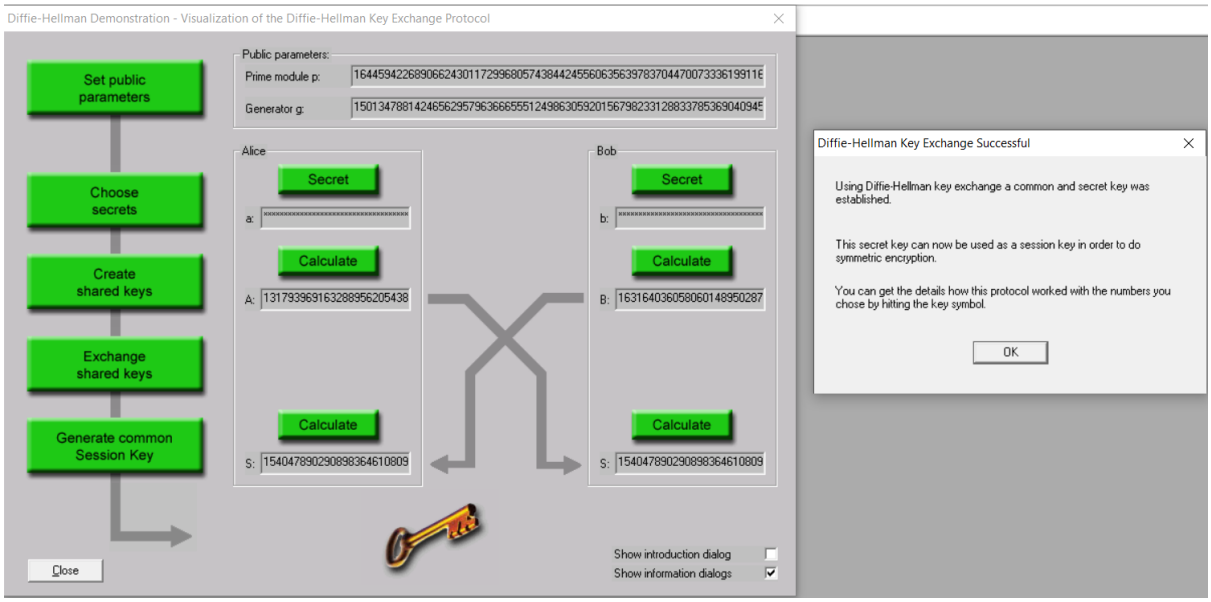
The brevity of these hashes also facilitates relatively fast ordinary brute force attacks. To address these vulnerabilities, it is now recommended to use more secure hash functions such as SHA-2 or SHA-3 (Keccak). The longer hashes produced by these functions significantly enhance resistance to both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

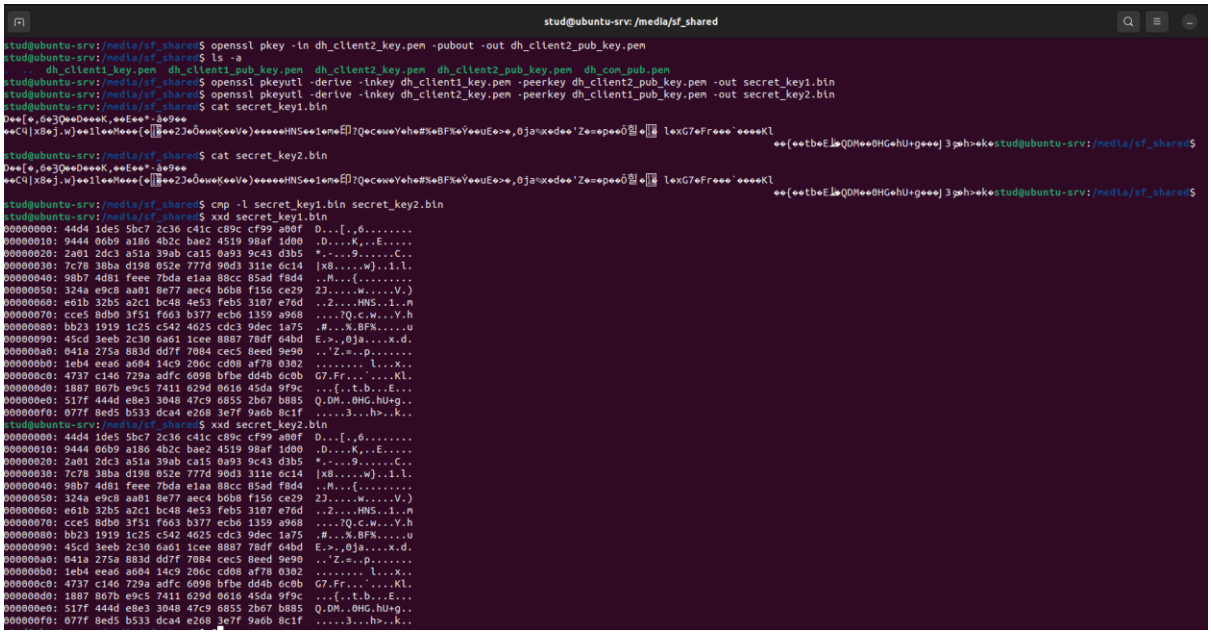
TASK 2




Choose Alice's secret
Please choose the secret for Alice now.
Please make sure to choose a natural number, preferably bigger than one but smaller than the previously defined prime module (p).
In case you do not want to make up any large number, click the button 'Generate Secret' to let CrypTool generate an appropriate number within a valid range of values.
Secret:
532412287284045715741819301682648144612076970068
Generate Secret
Accept secret
Cancel


Choose Bob's secret
Please choose the secret for Bob now.
Please make sure to choose a natural number, preferably bigger than one but smaller than the previously defined prime module (p).
In case you do not want to make up any large number, click the button 'Generate Secret' to let CrypTool generate an appropriate number within a valid range of values.
Secret:
Generate Secret
Accept secret
Cancel


Diffie-Hellman Demonstration - Visualization of the Diffie-Hellman Key Exchange Protocol
Set public parameters
Choose secrets
Create shared keys
Exchange shared keys
Generate common Session Key
Close
Public parameters:
Prime module p:
Generator g:
Alice
Secret
a:
Calculate
A:
Calculate
S:
Bob
Secret
b:
Calculate
B:
Calculate
S:
Show introduction dialog
Show information dialogs
Diffie-Hellman Key Exchange Successful
Using Diffie-Hellman key exchange a common and secret key was established.
This secret key can now be used as a session key in order to do symmetric encryption.
You can get the details how this protocol worked with the numbers you chose by hitting the key symbol.
OK



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.

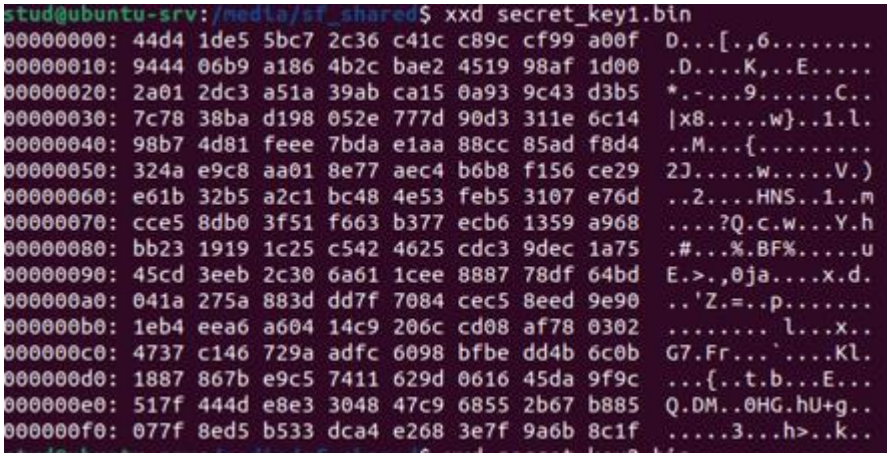
Furthermore, a password-authenticated key agreement is another avenue, where one of the communicating parties possesses partial knowledge of a shared password. A noteworthy recent development is quantum key distribution, which leverages the principles of quantum mechanics. In this method, the act of 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.


stud@ubuntu-srv:/media/sf_shared$ xxd secret_key1.bin
00000000: 44d4 1de5 5bc7 2c36 c41c c89c cf99 a00f D...[.,6........
00000010: 9444 06b9 a186 4b2c bae2 4519 98af 1d00 .D....K,..E.....
00000020: 2a01 2dc3 a51a 39ab ca15 0a93 9c43 d3b5 *.-...9......C..
00000030: 7c78 38ba d198 052e 777d 90d3 311e 6c14 |x8.....w}..1.l.
00000040: 98b7 4d81 feee 7bda e1aa 88cc 85ad f8d4 ..M...{.........
00000050: 324a e9c8 aa01 8e77 aec4 b6b8 f156 ce29 2J.....w.....V.)
00000060: e61b 32b5 a2c1 bc48 4e53 feb5 3107 e76d ..2....HNS..1..m
00000070: cce5 8db0 3f51 f663 b377 ecb6 1359 a968 ....?Q.c.w...Y.h
00000080: bb23 1919 1c25 c542 4625 cdc3 9dec 1a75 .#...%.BF%.....u
00000090: 45cd 3eeb 2c30 6a61 1cee 8887 78df 64bd E.>.,0ja....x.d.
000000a0: 041a 275a 883d dd7f 7084 cec5 8eed 9e90 ..'Z.=..p.......
000000b0: 1eb4 eea6 a604 14c9 206c cd08 af78 0302 .... l...x..
000000c0: 4737 c146 729a adfc 6098 bfbe dd4b 6c0b G7.Fr...`....Kl.
000000d0: 1887 867b e9c5 7411 629d 0616 45da 9f9c ...{..t.b...E...
000000e0: 517f 444d e8e3 3048 47c9 6855 2b67 b885 Q.DM..0HG.hU+g..
000000f0: 077f 8ed5 b533 dca4 e268 3e7f 9a6b 8c1f .....3...h>..k..