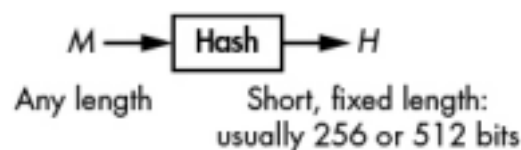


## CYBERSECURITY

## LAB 6

## 1. Introduction

The hash function is an important element of the message authentication technique. It gets various size inputs and produces a fixed-size hash value. The hash function uses a compression function repetitively to generate  $n$ -bit output. In the digital signature procedure, the hash value uses private and public keys for processing. Message authentication deals with the protection of messages with integrity. It checks the identity of the message sender and non-repudiation of the origin. It checks whether the received messages originated from the original sender. It ensures that the content of the message is not modified or altered. It also verifies the sequence and timing of the messages. A digital signature is an authentication technique that is used to check the repudiation from the sender's side or the receiver's side. The authentication of the digital signature is done on two levels. The sender sends a signed message to the receiver. The receiver compares the computed hash codes with the hash code he got. If both hashes match, he/she can view the message. The most popular hash function is SHA (Secure Hash Algorithm). SHAs are cryptographic algorithms that provide data integrity and authentication. TLS, SSL, SSH, and PGP applications use SHA. Unlike stream ciphers, which create a long output from a short one, hash functions take a long input and produce a short output, called a hash value or digest.



*Figure 1 A hash function's input and output*

Whitefield Diffie and Martin Hellman were the first to publish a public-key algorithm that solved the problem of key agreement or key exchange. The algorithm is generally referred to as Diffie-Hellman key exchange. The main purpose of the algorithm is to enable two users to securely exchange a key which may then be used for subsequent encryption of messages. The algorithm is limited only to the exchange of keys and is not used for encryption or decryption of messages.

The Diffie-Hellman Key Exchange is a cryptographic protocol exchanging cryptographic keys via a public network. It provides high security of key exchange between entities in an insecure channel. The exchanged keys are used later for symmetric communication using a popular symmetric cipher such as AES, Blowfish, ChaCha20, etc. DHKE is also called a key agreement protocol. The Diffie-Hellman Key Exchange protocol can be implemented using discrete logarithms (classic DHKE algorithm) or elliptical curve cryptography (ECDH algorithm). Below is a visualization of key exchange between users.

## LAB 6

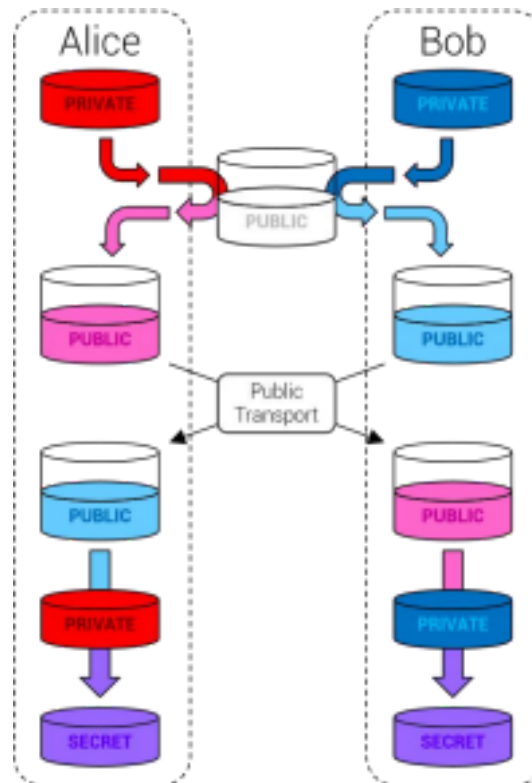


Figure 2 Diffie-Hellman key exchange protocol - source <http://crypto.mdc.io/2012/10/13/public-key-cryptography/>

## 2. Prerequisites

In this laboratory, the digest functions, the introduction to digital signatures, and key exchange algorithms are analyzed. During the execution of tasks, tools such as Cryptool and OpenSSL library are used. Get acquainted with the following elements: • In Cryptool look at:

- Digital signatures/PKI -> Signature Demonstration,
- Indiv. Procedures -> Hash -> Hash Demonstration,
- Indiv. Procedures -> Protocols -> Diffie-Hellman Demonstration,
- Diffie-Hellman key exchange,
- Hash function,

## 3. Environments

Virtual machines to use in this lab:

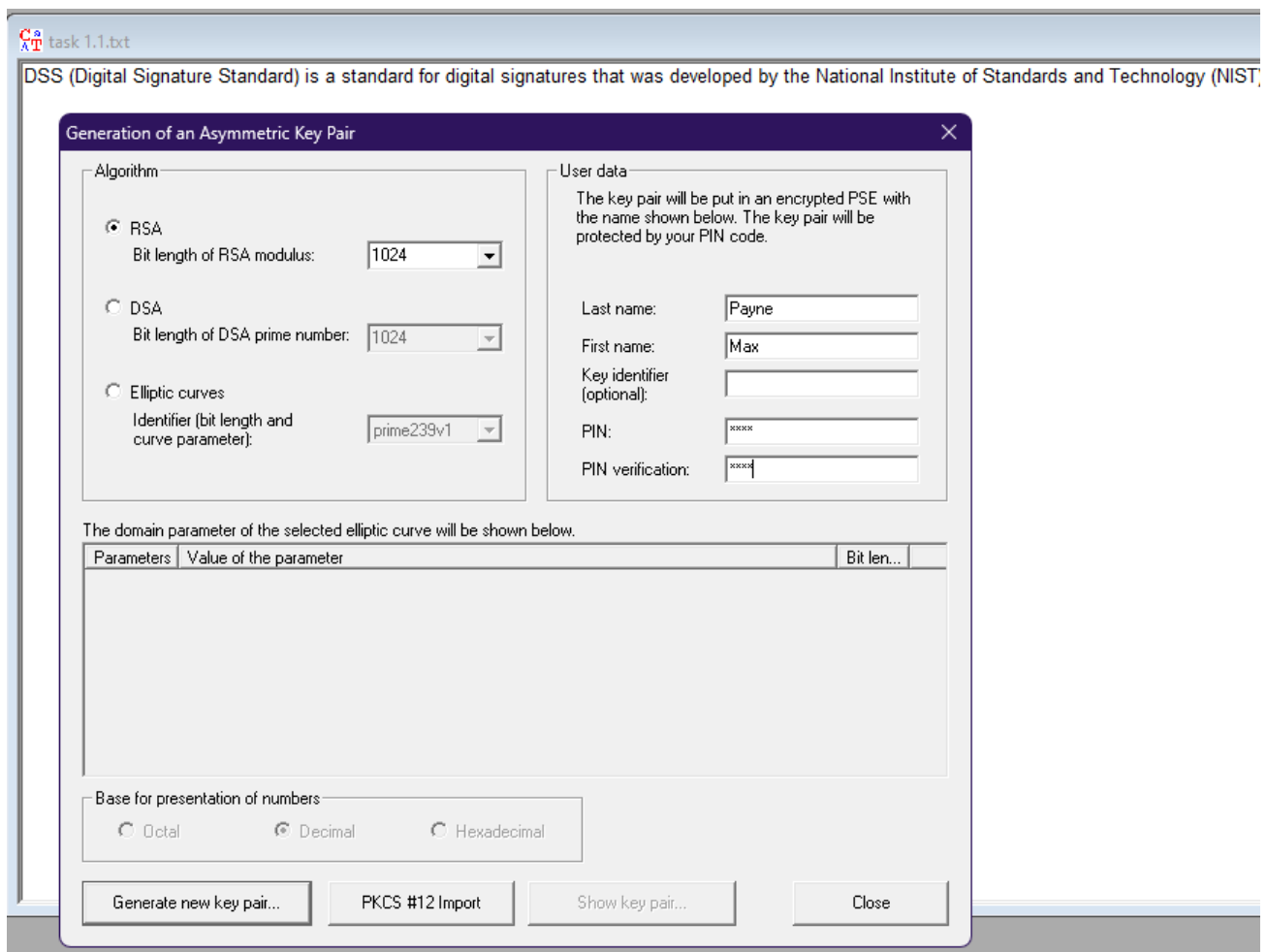
- Ubuntu Server,
- Kali,

## 4. Digital signature

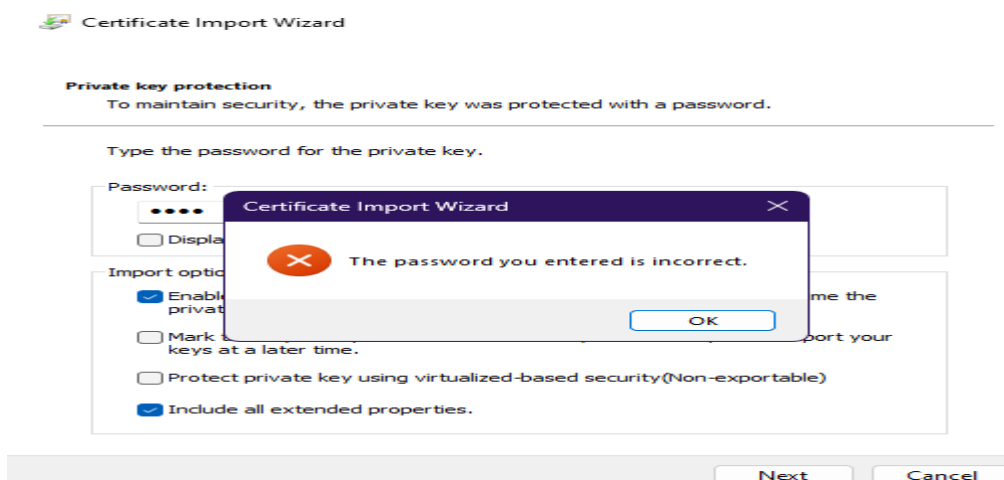
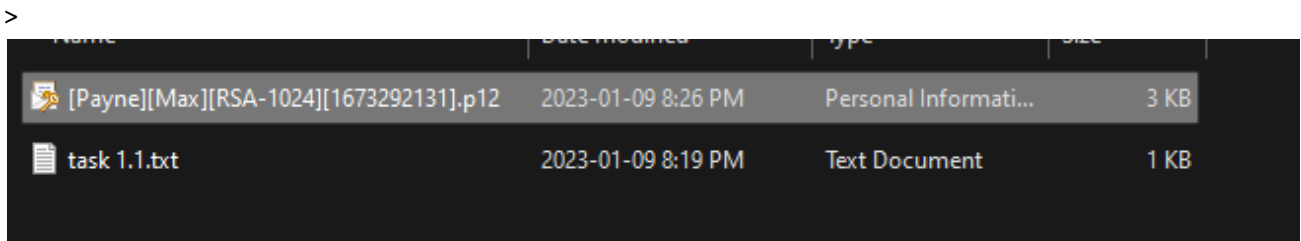
Tool: Cryptool

### I. Tasks:

1. Check the list of available keys and then generate your own key (RSA or DSS algorithm), and include its certificate in the report. Tab: Digital signatures/PKI -> PKI -> Generate Keys



2. Export your own certificate and then import it into your web browser.



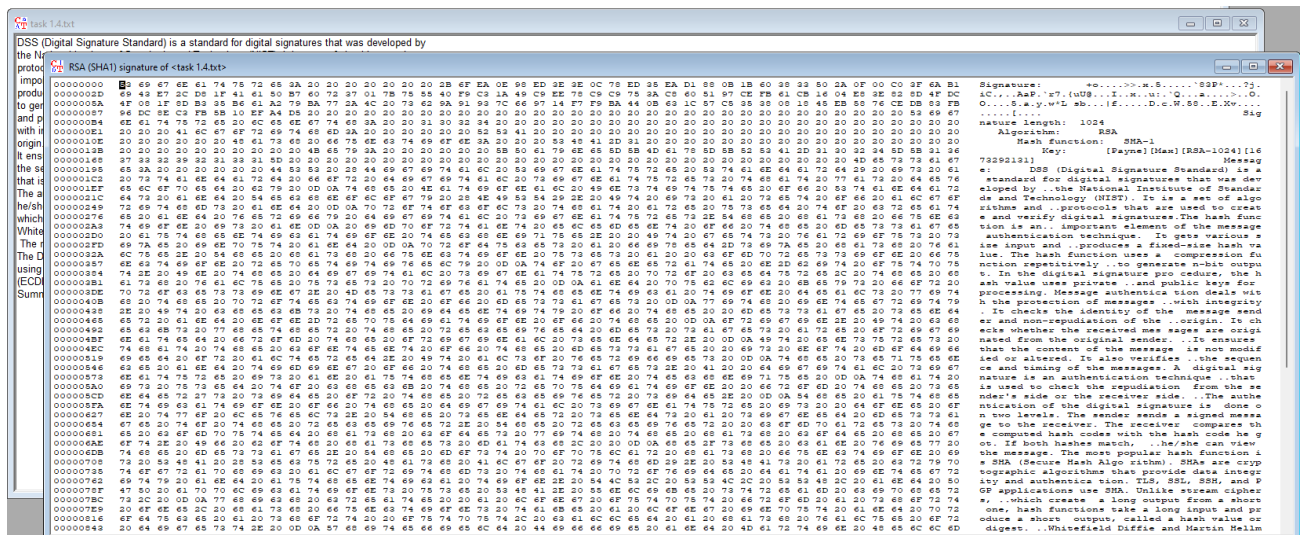
(i do not know why the same pin is not working)

3. Watch the demonstration showing the signing process.

> did not understand, please note on eportal for re-inspection / re-do !

#### 4. Sign any document with your own key.

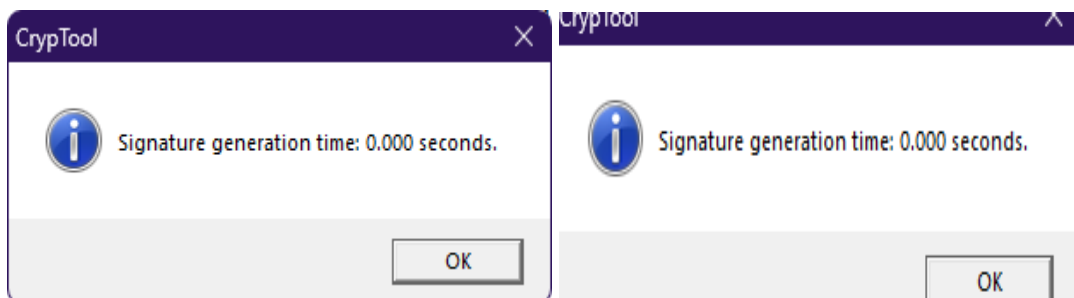
>



Signed a paragraph using created key using RSA, SHA-1 with 160 KB

#### 5. Compare the time of signing documents of different sizes using different keys.

>



From the two text files I worked with - I did not find any time difference.

### II. Questions:

#### 1. What are the elements of public-key certificates?

> The elements of public-key certificates are as follows;

- The name of the certificate owner
- The public key of the certificate owner
- The name of the certificate issuer
- The serial number of the certificate
- The validity period of the certificate
- The digital signature of the issuer, which is used to verify the authenticity

#### 2. What is a digital signature? What are the elements of a digital signature?

> A digital signature is a type of electronic signature that uses cryptography to verify the authenticity and integrity of a message or document. The elements of a digital signature typically include:

- The message or document being signed
- The private key of the signer, which is used to generate the signature
- The public key of the signer, which is used to verify the signature

- The signature itself, which is generated by applying a cryptographic hash function to the message or document and encrypting the resulting hash value with the private key

3. How does the time of execution of a digital signature of a document change and what does it depend on?

> The time of execution of a digital signature on a document does not change. The signature remains valid for as long as the certificate is valid, unless the certificate is explicitly revoked or the document is modified in a way that invalidates the signature.

4. Check which other certificates you can find in your web browser and include some examples in the report.

> In a web browser, you can find various types of certificates, including SSL/TLS certificates, which are used to secure web connections, and code signing certificates, which are used to verify the authenticity and integrity of software programs.

5. What will change when a new certificate is added to the list of trusted certificates in the browser?

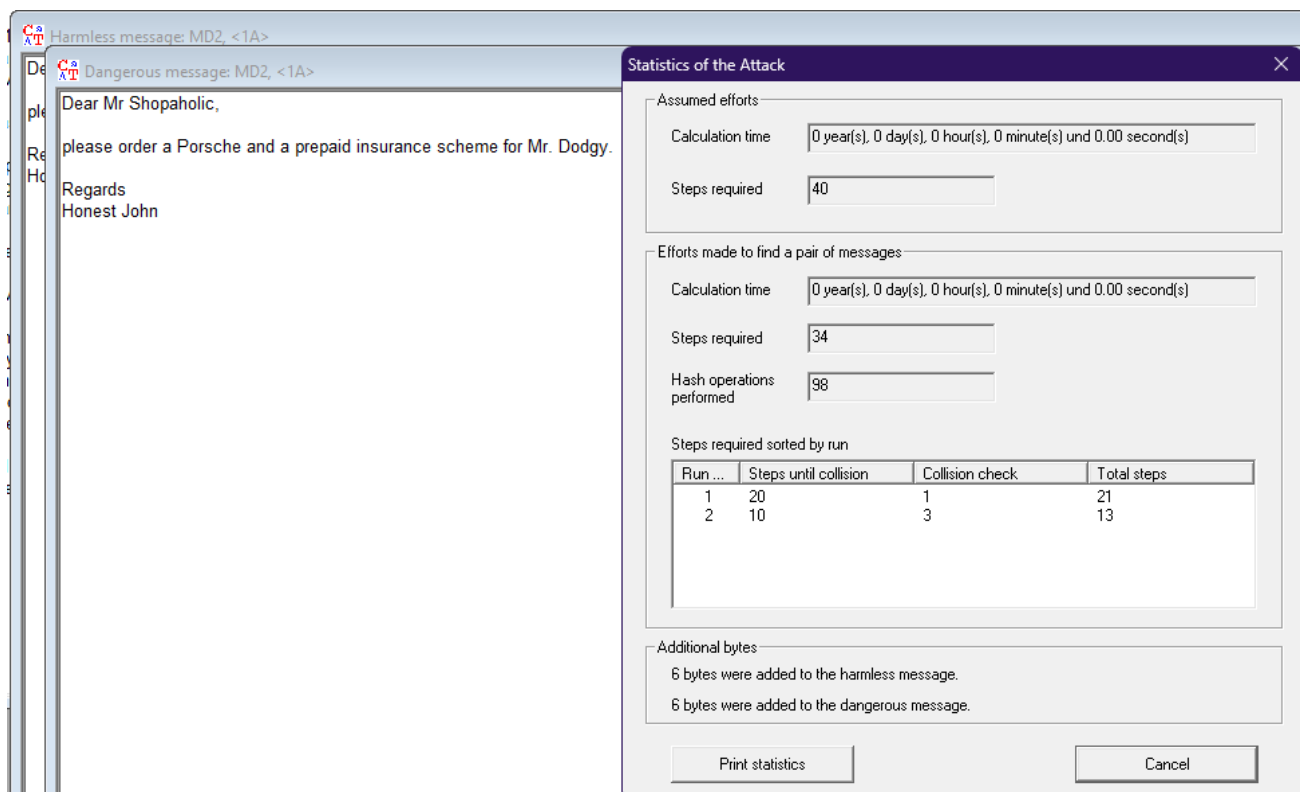
> It means that the browser will trust websites and applications that present that certificate. This can be useful in situations where you want to trust a certificate that is not issued by a well-known CA, or if you want to temporarily trust a certificate for testing purposes.

## 5. Hash function

### I. Tasks:

1. Perform the attack on the hash function based on the default files.  
(Analysis -> Hash -> Attack on the hash value of the digital signature)

>



The screenshot shows a software interface with an email window on the left and a 'Statistics of the Attack' dialog box on the right. The email window displays two messages: 'Harmless message: MD2, <1A>' and 'Dangerous message: MD2, <1A>'. The 'Dangerous message' content is: 'Dear Mr Shopaholic, please order a Porsche and a prepaid insurance scheme for Mr. Dodge. Regards Honest John'. The 'Statistics of the Attack' dialog box provides detailed information about the attack process.

**Statistics of the Attack**

Assumed efforts

Calculation time: 0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.00 second(s)

Steps required: 40

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: 34

Hash operations performed: 98

Steps required sorted by run

| Run ... | Steps until collision | Collision check | Total steps |
|---------|-----------------------|-----------------|-------------|
| 1       | 20                    | 1               | 21          |
| 2       | 10                    | 3               | 13          |

Additional bytes

6 bytes were added to the harmless message.

6 bytes were added to the dangerous message.

Buttons: Print statistics, Cancel

2. Perform the attack again, but change the hash function and the significant bit length(at least two others).

> with, SHA & 32 significant bits,

Statistics of the Attack

Assumed efforts

Calculation time0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 1.72 second(s)

Steps required163,840

Efforts made to find a pair of messages

Calculation time0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.14 second(s)

Steps required194,341

Hash operations performed513,884

Steps required sorted by run

| Run ... | Steps until collision | Collision check | Total steps |
|---------|-----------------------|-----------------|-------------|
| 1       | 62,628                | 36,325          | 98,953      |
| 2       | 62,574                | 32,814          | 95,388      |

Additional bytes

18 bytes were added to the harmless message.

18 bytes were added to the dangerous message.

Print statistics

Cancel

&

MD5 with 24 bits,

Statistics of the Attack

Assumed efforts

Calculation time

0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.06 second(s)

Steps required

10,240

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

2,232

Hash operations performed

5,616

Steps required sorted by run

| Run ... | Steps until collision | Collision check | Total steps |
|---------|-----------------------|-----------------|-------------|
| 1       | 1,152                 | 1,080           | 2,232       |

Additional bytes

14 bytes were added to the harmless message.

14 bytes were added to the dangerous message.

Print statistics

Cancel

We can see some minor time change with changing lengths and hashes.

3. Perform the attack again using a larger text file (at least a few megabytes).

> For harmless 2mb & dangerous 1mb file we can see -

The screenshot shows a window titled "Statistics of the Attack" with a close button (X) in the top right corner. The window is divided into three main sections: "Assumed efforts", "Efforts made to find a pair of messages", and "Additional bytes".

**Assumed efforts**

- Calculation time: 0 year(s), 0 day(s), 0 hour(s), 0 minute(s) und 0.06 second(s)
- Steps required: 640

**Efforts made to find a pair of messages**

- Calculation time: 0 year(s), 0 day(s), 0 hour(s), 5 minute(s) und 32.88 second(s)
- Steps required: 516
- Hash operations performed: 1,447

**Steps required sorted by run**

| Run ... | Steps until collision | Collision check | Total steps |
|---------|-----------------------|-----------------|-------------|
| 1       | 130                   | 20              | 150         |
| 2       | 285                   | 81              | 366         |

**Additional bytes**

- 10 bytes were added to the harmless message.
- 10 bytes were added to the dangerous message.

At the bottom of the window, there are two buttons: "Print statistics" and "Cancel".

## II. Questions:

1. How does changing the key length affect the time of the attack?

> Increasing the key length of a cryptographic algorithm generally makes it more secure against attacks, but it also typically increases the amount of time and resources required to perform the attack

2. Does the selection of the hash function affect the time of the collision search task (attack)?

> The selection of the hash function can affect the time required to find a collision, as different hash functions have different properties that make them more or less resistant to collision attacks

3. What is the advantage of an attack (finding a collision) whose aim is to modify two documents rather than one?

> An attack that aims to modify two documents rather than one may have the advantage of being able to produce more subtle or sophisticated changes to the documents.



4. Can the hash function be regarded as safe? If so, for which parameters?

> It is generally not possible to say that a hash function is completely "safe" against collision attacks, as all hash functions are theoretically vulnerable to such attacks.

## 6. Key exchange - Diffie-Hellman protocol

Get acquainted with Diffie-Hellman Demonstration available in Cryptool. For demonstration purposes, use values generated in Cryptool. Then the OpenSSL library is used for the practical application of the DH protocol. Use two virtual machines as clients (Kali Linux, Ubuntu) to exchange the keys.

Short description of the next steps:

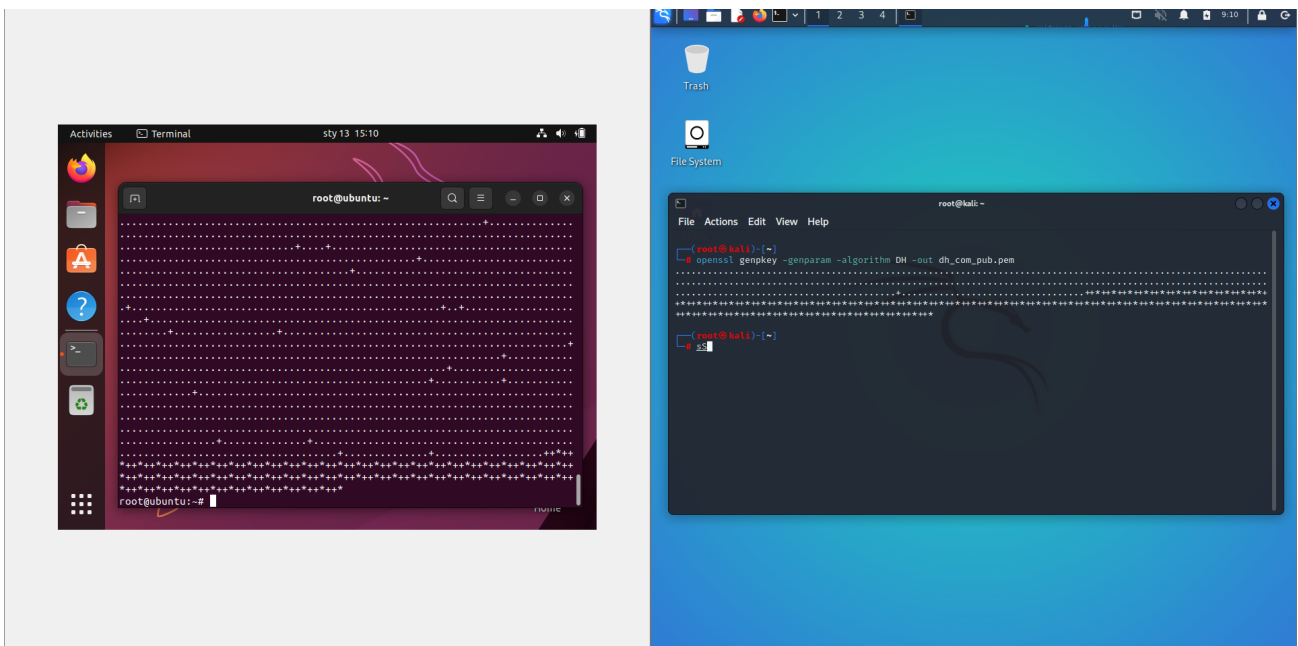
- Generate public DH key for both clients,
- Based on the public DH key, the clients generate public and private keys, -
- The clients must exchange their public keys,
- Based on the public key and private key, the shared secret key is derived.

### I. Tasks:

a. Generate the Diffie-Hellman common public key for both clients:

```
openssl genpkey -genparam -algorithm DH -out dh_com_pub.pem
```

>

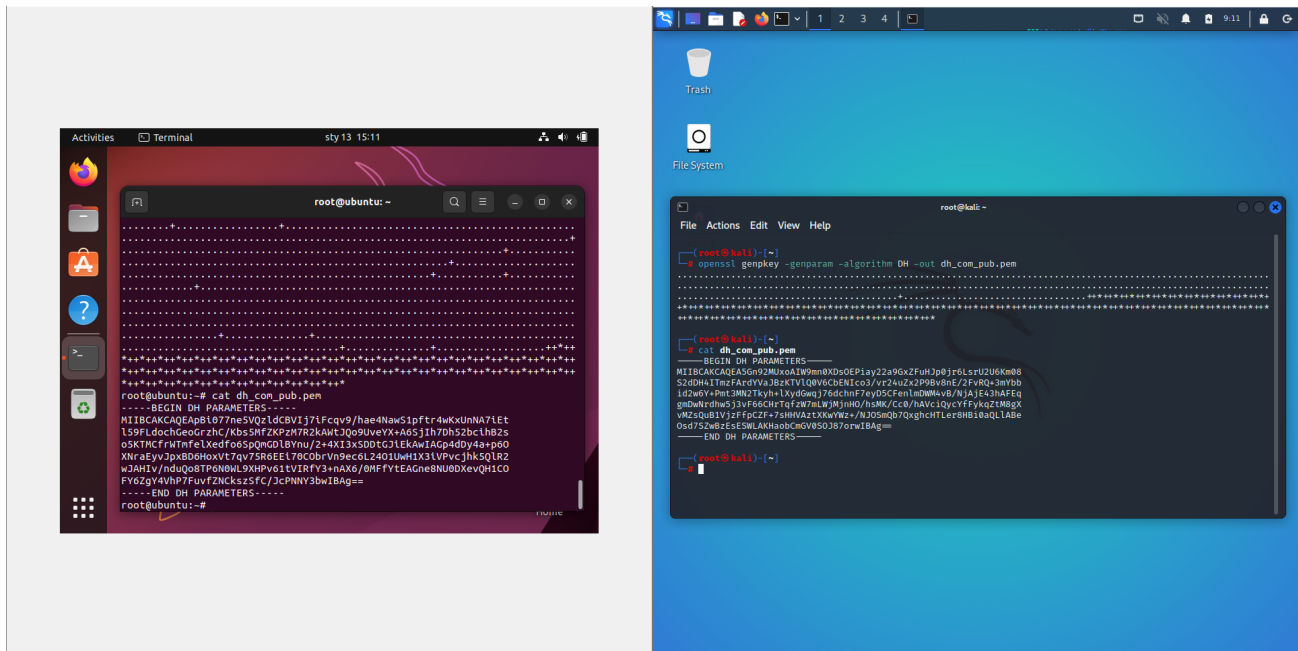


b. To display the common key, use:

a. Encoded form:

```
cat dh_com_pub.pem
```

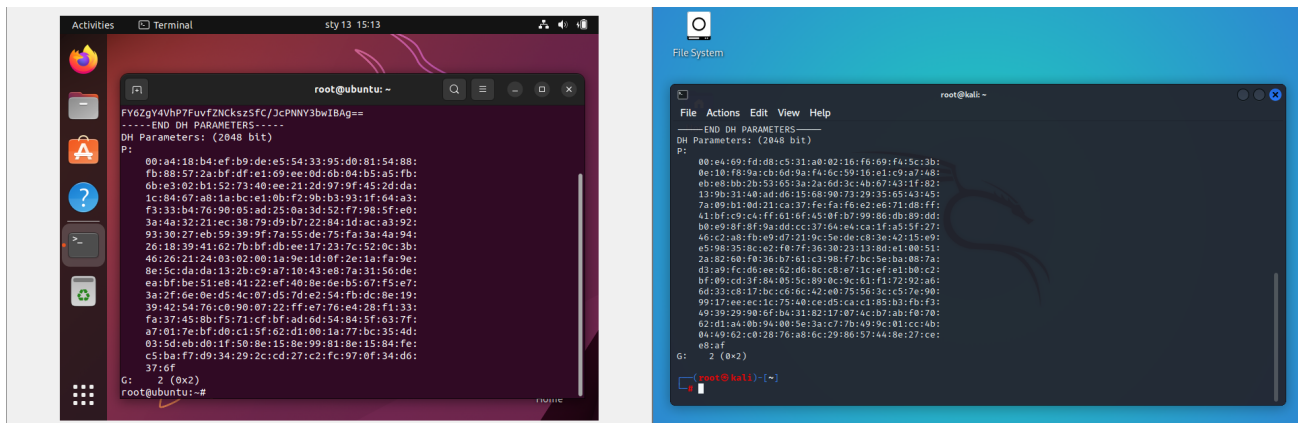
>



## b. Text form:

```
openssl pkeyparam -in dh_com_pub.pem -text
```

>



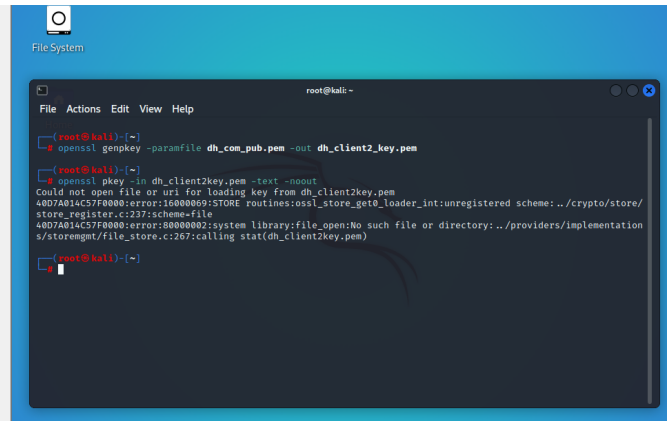
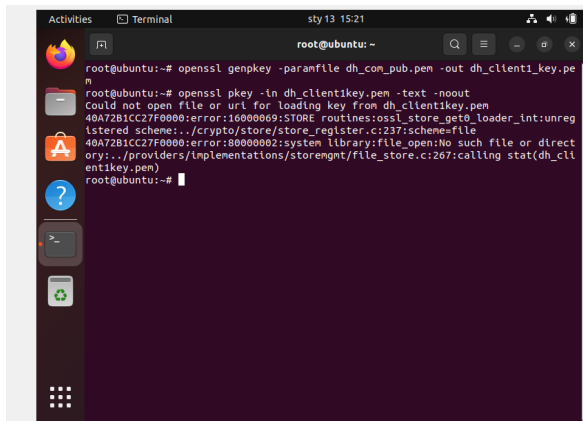
## c. Based on common DH key, each client should generate their private keys:

```
openssl genpkey -paramfile dh_com_pub.pem -out  
dh_client1_key.pem
```

List the key:

```
openssl pkey -in dh_client1_key.pem -text -noout
```

>

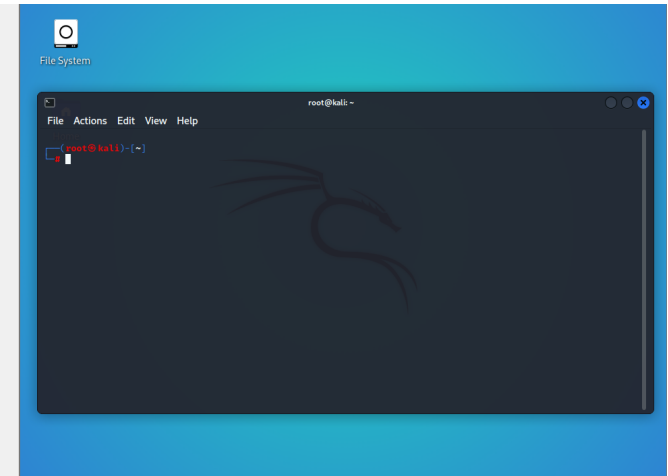
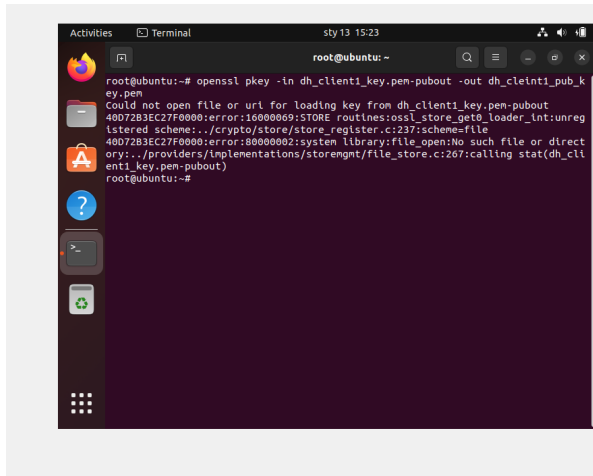


(could not extract)

#### d. Extract the public key based on the private key:

```
openssl pkey -in dh_client1_key.pem -pubout -out
dh_client1_pub_key.pem
```

>



(not working)

#### e. Exchange the public keys of the client between them and generate the shared secret key:

```
OpenSSL pkeyutl -derive -inkey dh_client1_key.pem -peerkey
dh_client2_pub_key.pem -out secret_key1.bin
```

> (did not work)

#### f. Compare the shared secret keys using cmp command.

> not resolved

#### g. Create a hex dump of the shared secrets using xxd command.

> not resolved

(the instructions were not clear enough!! sorry.)

## II. Questions:

1. Which key exchange algorithm guarantees higher security? RSA versus DH. Why?

> In terms of security, DH is considered more secure than RSA because it is based on the discrete logarithm problem, which is considered to be a stronger mathematical problem than factoring large integers, the problem that RSA is based on.

2. What are the risks for entities using DH key exchange protocol?

> The main risk for entities using DH key exchange protocol is the potential for a man-in-the-middle (MitM) attack.

3. What are the other methods to determine a common cryptographic key?

> Other methods to determine a common cryptographic key include symmetric key algorithms such as AES and Triple-DES etc.

4. Which element of the DH protocol is not transferred between clients? How does this affect security issues?

> In the DH protocol, the private key is not transferred between clients. This affects security issues because the private key is used to calculate the shared secret key and it needs to be kept secret. If an attacker intercepts the private key, they can use it to impersonate one of the clients and intercept the communication.

5. Check whether the last point has managed to establish the same key for both clients. Include the content of the agreed secret key in the report.

> It's hard to determine whether the last point has managed to establish the same key for both clients without further context. The secret key is the result of the DH key exchange protocol and it is based on the private key of the two clients and the public key that they exchanged. It is a value that is derived through a mathematical calculation and cannot be determined without the private key.