



ETHICAL HACKING V2 LAB SERIES

Lab 26: Cryptography

Document Version: **2021-05-18**

Material in this Lab Aligns to the Following	
Books/Certifications	Chapters/Modules/Objectives
All-In-One CEH Chapters ISBN-13: 978-1260454550	11: Cryptography 101
EC-Council CEH v10 Domain Modules	20: Cryptography

Copyright © 2021 Network Development Group, Inc.
www.netdevgroup.com

NETLAB+ is a registered trademark of Network Development Group, Inc.

Microsoft® and Windows® are registered trademarks of Microsoft Corporation in the United States and other countries. Google is a registered trademark of Google, LLC.

Contents

Introduction	3
Objectives.....	3
Lab Topology	4
Lab Settings	5
1 Learn How Hash Values Change When Text is Edited	6
2 Encrypting and Decrypting Text.....	12

Introduction

This lab focuses on hashing files and then recognizing changes to the hash value after the file is modified. It will also cover another method to hide text or files within other files using steganography techniques.

Objectives

- Use encrypting/decrypting techniques
- Generate hashes and file checksum

Lab Topology



WinOS

Lab Settings

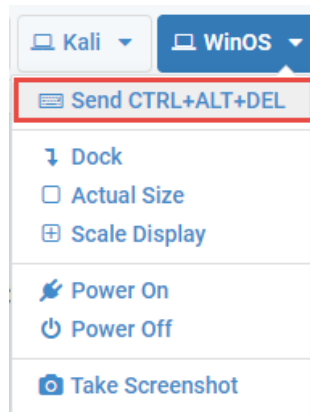
The information in the table below will be needed in order to complete the lab. The task sections below provide details on the use of this information.

Virtual Machine	IP Address / Subnet Mask	Account (if needed)	Password (if needed)
WinOS	192.168.0.20	Administrator	Train1ng\$

1 Learn How Hash Values Change When Text is Edited

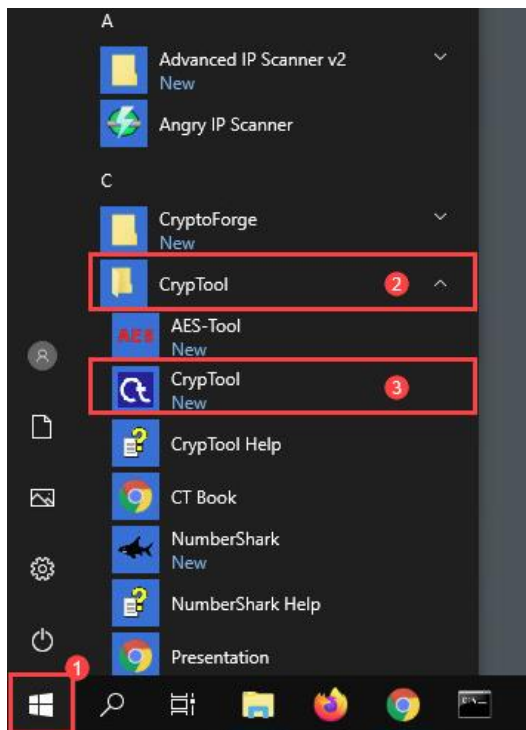
Hashing is the process of generating a unique value using a mathematical algorithm. This hash value can be used to validate digital information. This means a hash value can be used to determine if files have been changed since they have been hashed. The hash value can be considered as a serial number for a file and will be completely different if the slightest change is made to the file. In this exercise, we will hash some data and compare the values to see the verification in action.

1. Launch the **WinOS** virtual machine to access the graphical login screen.
 - 1.1. Select **Send CTRL+ALT+DEL** from the dropdown menu to be prompted with the login screen.



- 1.2. Log in as **Administrator** using the password: **Train1ng\$**

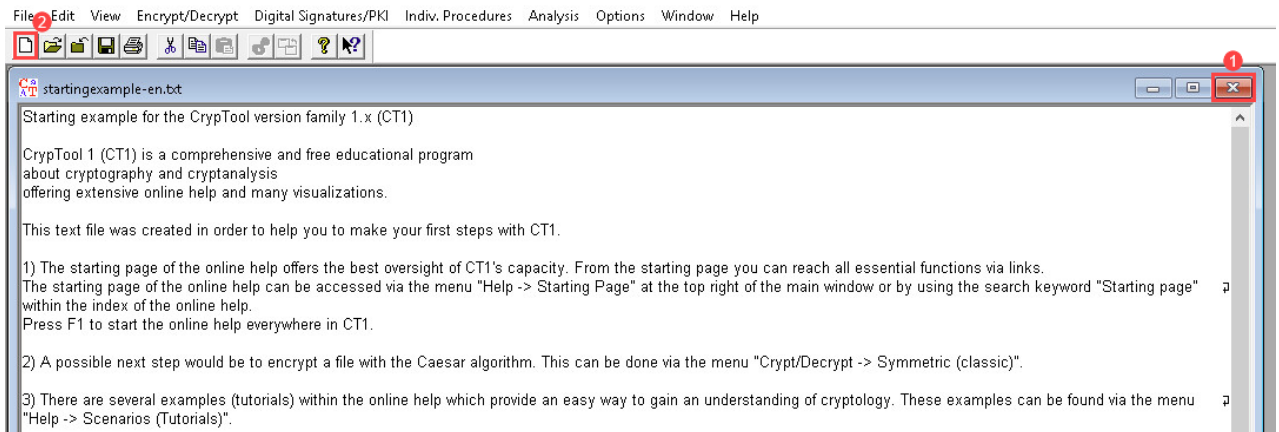
- Let's begin by opening the software called *Cryptool* by clicking the **Start Menu** button seen in *item 1* below. Next, navigate to and click the folder called **Cryptool** and then click the **Cryptool** icon seen in *items 2* and **3** below.



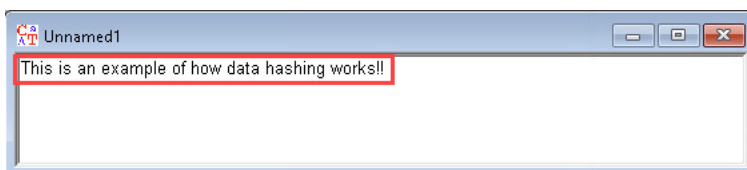
- The *Cryptool* window will appear. This is a free educational tool that is great for learning about the wonders of cryptography. This tool has many features and a detailed help file, so we will not do a familiarization for this one. Instead, we will go directly to the feature we will be using in this lab. Begin by reading the *How to Start* window and then clicking the **Close** button once you are done, as highlighted below.



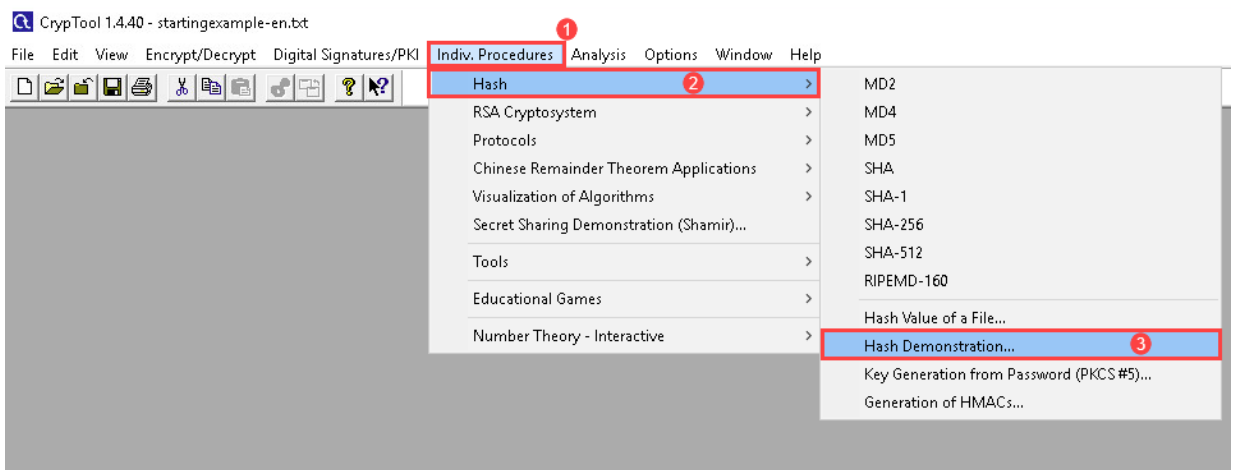
- The main window will display a sample text file, but we will not be using that for this exercise. Instead, click the **X** at the top-right corner of that window to close it, then click the **New** icon from the toolbar to create a new file, as seen in *items 1 and 2* below.



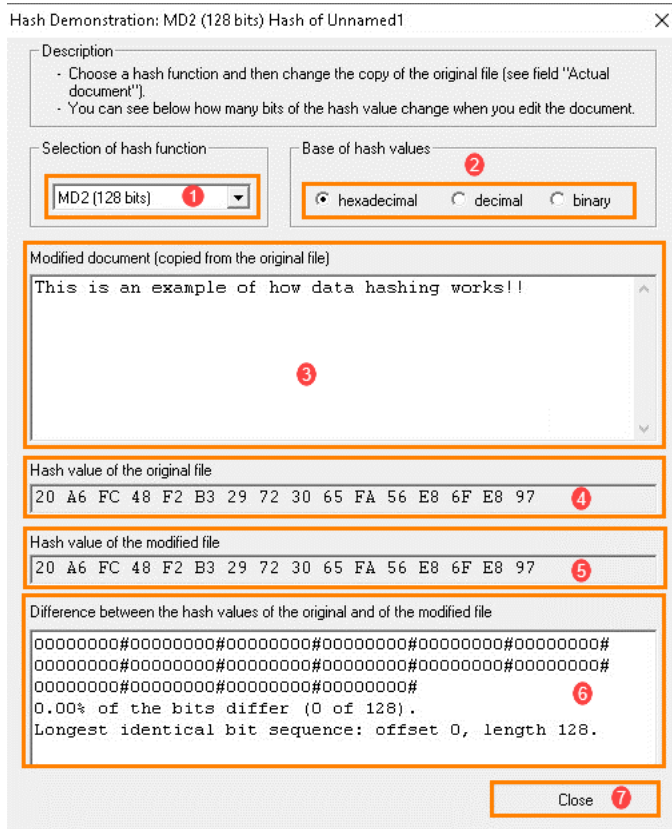
- A new blank file will appear, so let's type some text in this window. Type the following text (Or text of your choice): **This is an example of how data hashing works!!** in the new window as seen below:



- Next, navigate to and click **Indiv. Procedures** from the *Menu bar* as seen in *item 1* below. Next, hover over **Hash** from the dropdown menu that appears and then click **Hash Demonstration...** from the submenu that appears, as seen in *items 2 and 3* below.



7. The *Hash Demonstration* window will appear. In this window, we will be able to see the hexadecimal, decimal, and binary hash values of the data we typed earlier. We can also compare the hash of the data we typed with the hash of the same data after we modify it a little. The table below the following screenshot provides a quick outline of the features we will be using in this exercise.



Hash Demonstration: MD2 (128 bits) Hash of Unnamed1

Description

- Choose a hash function and then change the copy of the original file (see field "Actual document").
- You can see below how many bits of the hash value change when you edit the document.

Selection of hash function

MD2 (128 bits)

Base of hash values

hexadecimal decimal binary

Modified document (copied from the original file)

This is an example of how data hashing works!!

Hash value of the original file

20 A6 FC 48 F2 B3 29 72 30 65 FA 56 E8 6F E8 97

Hash value of the modified file

20 A6 FC 48 F2 B3 29 72 30 65 FA 56 E8 6F E8 97

Difference between the hash values of the original and of the modified file

00000000#00000000#00000000#00000000#00000000#
 00000000#00000000#00000000#00000000#00000000#
 00000000#00000000#00000000#00000000#
 0.00% of the bits differ (0 of 128).
 Longest identical bit sequence: offset 0, length 128.

Close

#	Name	Description
1	<i>Selection of hash function</i>	This option allows you to choose between different hash algorithms
2	<i>Base of hash values</i>	The radio buttons here allow you to choose what base the hash value should be represented in
3	<i>Modified document</i>	This shows the text of the document that is being hashed and allows you to make changes to the document
4	<i>Hash value of the original file</i>	This is the hash value of the data represented in the <i>Modified document</i> field
5	<i>Hash value of the modified file</i>	This is the hash value of the data represented in the <i>Modified document</i> field after you add text or make changes to it
6	<i>Difference between the hash values of the original and the modified file</i>	This display shows the differences between the old file and the new one
7	<i>Close</i>	This is the option to close the window

8. Now that you are a bit more familiar with the interface, let's add some data to the *modified document* field and see what happens. First, pay attention to the data in the *Hash value of the original file* field seen in *item 1* below; this is the MD5 hash value of the data you typed earlier. Now, go back to the *Modified document* field and type the following text (or text of your choice), as seen in *item 2*: **This example shows the differences between the old file and the new one.** Notice as you type, the hash value in the *hash value of the modified file* field, seen in *item 3*, changes simultaneously. The red values in the *Difference between the hash values of the original and the modified file* field, seen in *item 4*, also change simultaneously and represent the new values that cause the hash to be different. When you compare the hashes in *items 1* and *3*, you can see that the difference is significant, even when you add just one character.

Hash Demonstration: MD2 (128 bits) Hash of Unnamed1

Description

- Choose a hash function and then change the copy of the original file (see field "Actual document").
- You can see below how many bits of the hash value change when you edit the document.

Selection of hash function

MD2 (128 bits)

Base of hash values

☒ hexadecimal ☐ decimal ☐ binary

Modified document (copied from the original file)

2 This is an example of how data hashing works!!
This example shows the differences between the old file and the new one.

Hash value of the original file

1 20 A6 FC 48 F2 B3 29 72 30 65 FA 56 E8 6F E8 97

Hash value of the modified file

3 4E DB AA 17 EE 0A 20 AD B5 D6 04 BD 21 F4 FF 5F

Difference between the hash values of the original and of the modified file

4 01101110#01111101#01010110#01011111#00011100#10111001#
00001001#11011111#10000101#10110011#11111110#11101011#
11001001#10011011#00010111#11001000#
58.59% of the bits differ (75 of 128).
Longest identical bit sequence: offset 48, length 4.

Close

9. Now let's take a look at what this data looks like when hashed using a different algorithm. To do this, click the **arrow** to open the *Selection of hash function* dropdown menu, as seen in *item 1*. There are several different options there, but we'll just look at one. Click on **SHA-256** to choose that hash algorithm, as seen in *item 2*. The hash values in the respective fields are now completely different, as seen in *item 3*. They are much longer as well, which means that this algorithm is a lot more secure. Feel free to look at different algorithm outputs and make some changes to the options and text to see how it affects the hash values. When you are done, click the **Close** button seen in *item 4* below.

Hash Demonstration: SHA-256 (256 bits) Hash of Unnamed1

Description

- Choose a hash function and then change the copy of the original file (see field "Actual document").
- You can see below how many bits of the hash value change when you edit the document.

Selection of hash function

SHA-256 (256 bits) 1

SHA (160 bits)

SHA-1 (160 bits)

2 SHA-256 (256 bits)

SHA-512 (512 bits)

TRIPED-160 (160 bits)

Base of hash values

☒ hexadecimal ☐ decimal ☐ binary

original file

how data hashing works!!

This example shows the difference between the old file and the new one.

3

Hash value of the original file

92 8C 4E E8 78 30 36 C9 D8 A9 0E 00 C6 51 8C 0B 7C CA 1F 7

Hash value of the modified file

A3 85 45 59 CB 14 B1 4A DB 95 3F 93 83 A1 66 BB 10 96 37 D

Difference between the hash values of the original and of the modified file

00110001#00001001#00001011#10110001#10110011#00100100#
 10000111#10000011#00000011#00111100#00110001#10010011#
 01000101#11110000#11101010#10110000#01101100#01011100#
 00101000#10100010#01110000#11101011#11011101#00000110#
 11100010#00101011#00000111#11111111#01111000#01110111#
 00010100#01101010#

4

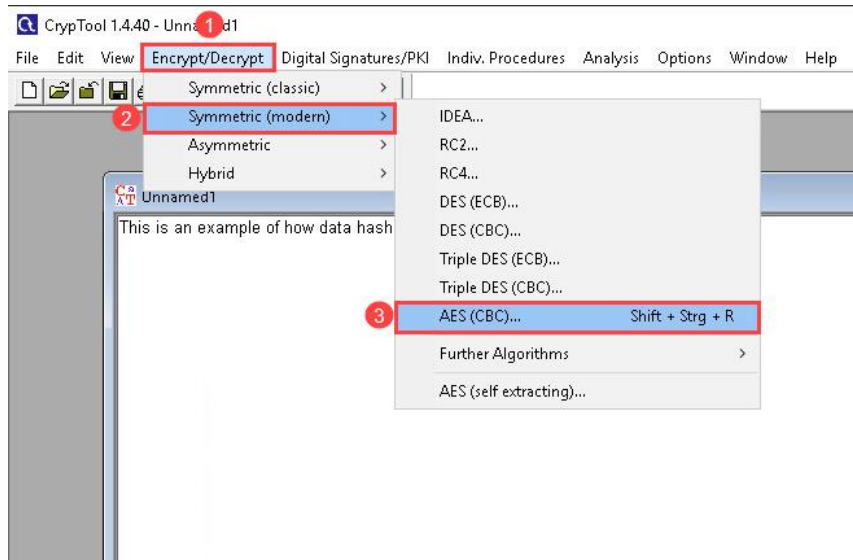
Close

Next, we will use *CrypTool* to see how to encrypt and decrypt text.

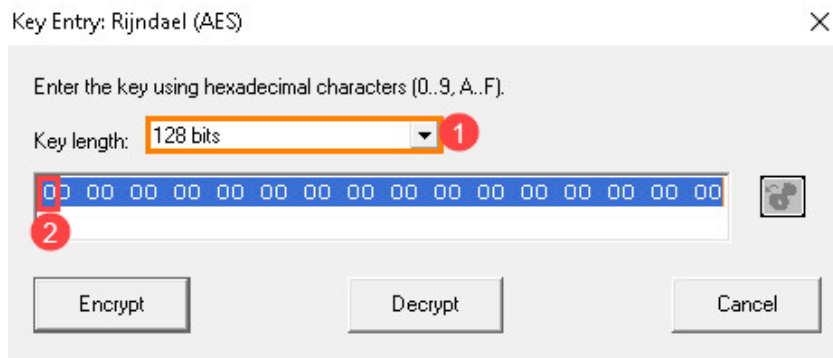
2 Encrypting and Decrypting Text

Since *CrypTool* is still open, let's move directly to the next exercise. In this exercise, we will be quickly encrypting and decrypting the text you typed in the previous exercise.

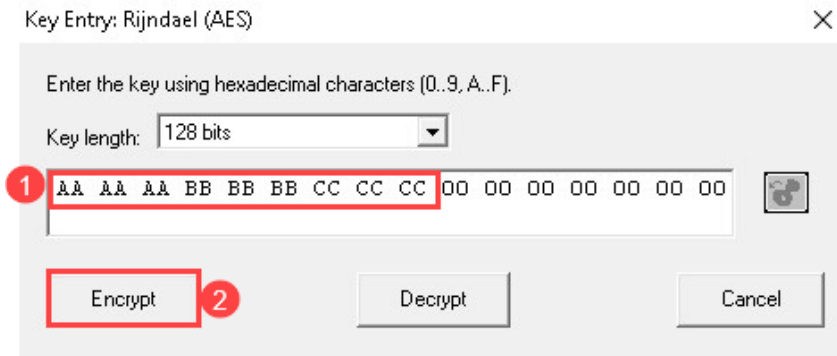
1. To begin, navigate to **Encrypt/Decrypt > Symmetric (modern) > AES (CBC)** as seen in *items 1, 2, and 3* below:



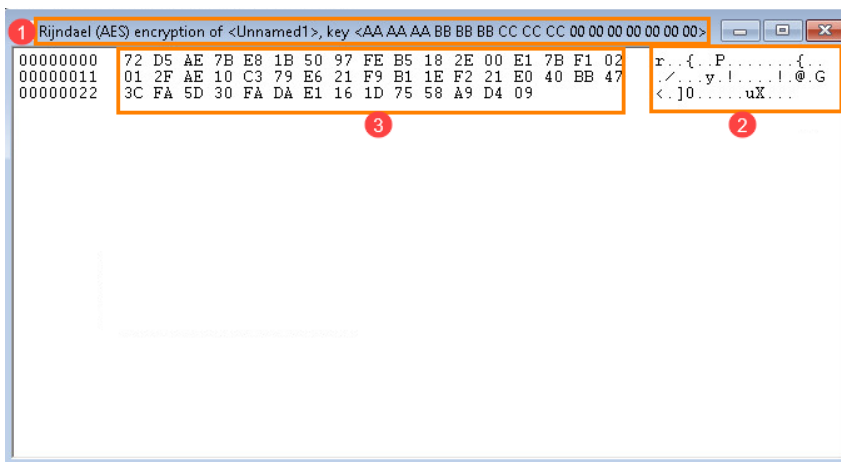
2. The *Key Entry* window will appear, which allows you to set the encryption key and its length. As you can see in *item 1*, the key length is 128bits which means the key will have 32 characters. Let's make a simple encryption key to use for encrypting our text. Begin by clicking on the first character in the text box, as seen in *item 2* below.



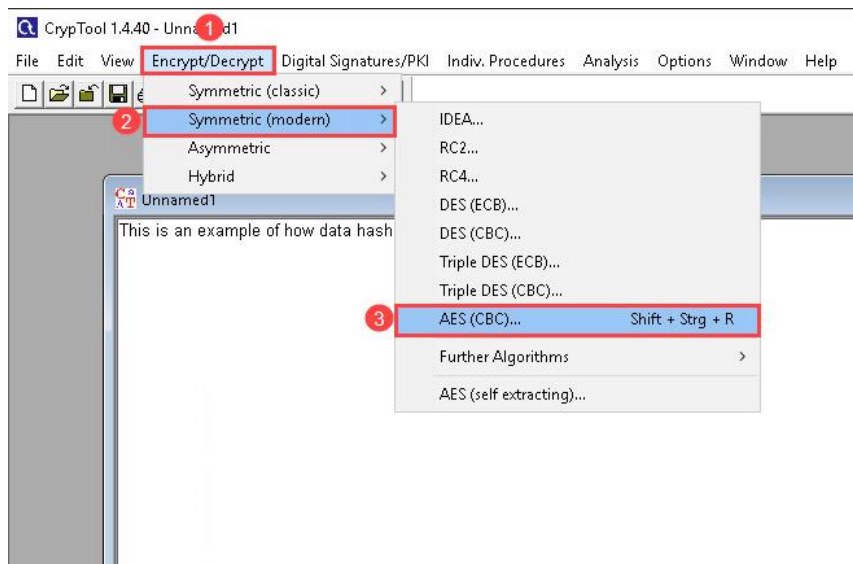
3. Type the following text in the textbox: AA AA AA BB BB BB CC CC CC as seen in *item 1* below. In normal circumstances, you must ensure that you type the key correctly because this key is used to encrypt the document. A mistake here would normally mean that the data will be unrecoverable. Because *CrypTool* is designed for learning, the key is displayed on the window of the encrypted data. Once you have verified that the key is typed correctly, click the **Encrypt** button seen in *item 2* below.



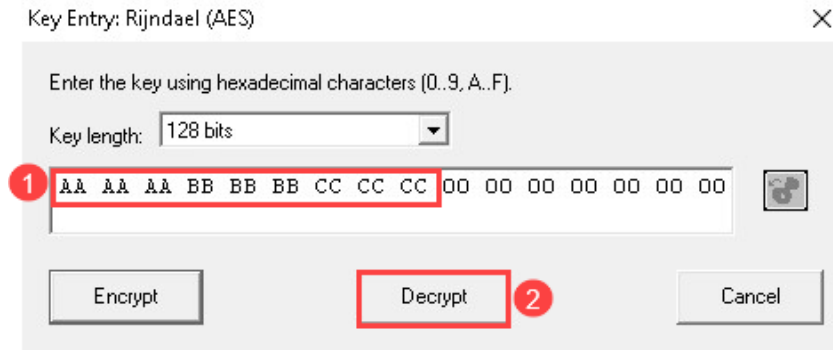
4. Since we're encrypting a very small amount of text, the process will be extremely fast. You should see the following window appear with ciphertext, as seen in *item 1* below. It is unreadable and can only be reverted using the key we typed before. The hexadecimal representation of the data can be seen in *item 2*. *Item 3* shows the encryption algorithm and the key.



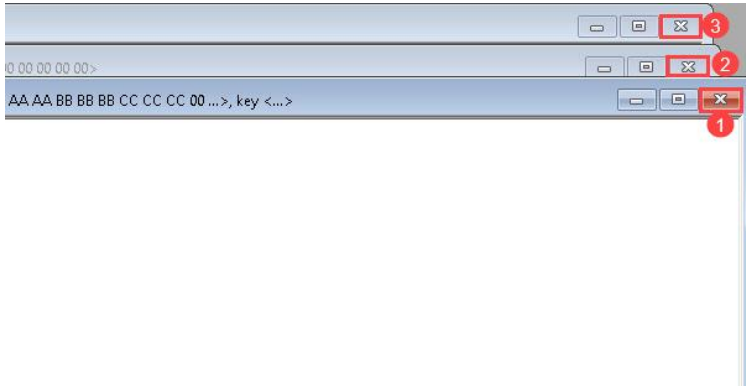
5. Now, let's decrypt this text using our key. To do this, navigate to **Encrypt/Decrypt > Symmetric (modern) > AES (CBC)** as seen in *items 1, 2, and 3* below:



6. The *Key Entry* will appear again. Type the key we created earlier **AA AA AA BB BB BB CC CC CC** as seen in *item 1* below. Once you have verified that the key is correct, click **Decrypt** seen in *item 2* below.



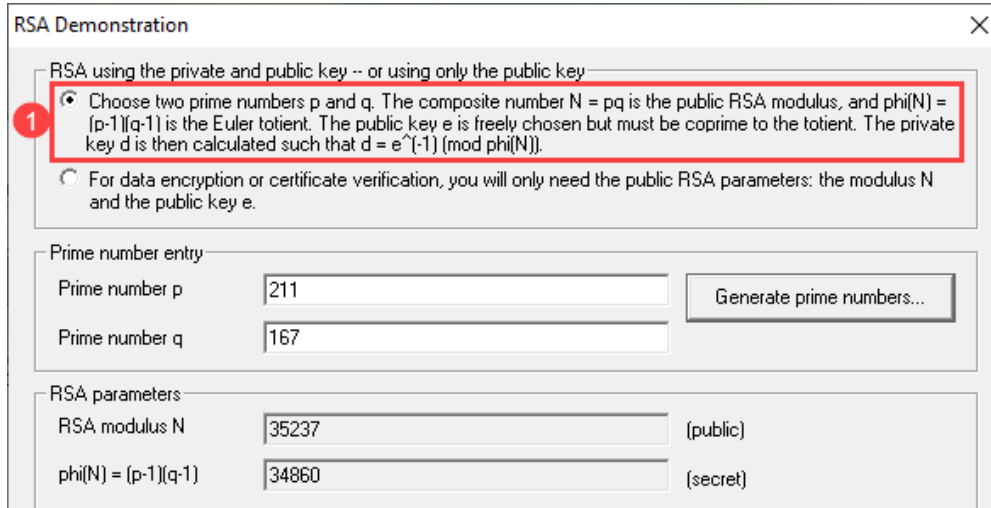
7. You have successfully encrypted and decrypted data using the symmetric algorithm AES. You can compare the contents by switching between the windows. Once you are done, close all the windows open windows seen in *items 1, 2, and 3* below by clicking the **X** at the top-right corner of each one.



8. Now we will encrypt the original text using an Asymmetric algorithm. To begin, navigate to **Encrypt/Decrypt > Asymmetric > RSA Demonstration** as seen in *items 1, 2, and 3* below:



9. Please note, the following process can be very complicated. In this exercise, we will try to stick to simpler explanations to make it easier to understand. The *RSA Demonstration* window should now be open. This window contains several fields and options. We will go through the exercise gradually. The first 2 options are the *RSA using the private and public key or using only the public key* radio buttons. We will leave the first option selected as seen in *item 1*. This indicates that we want to generate both a public and private key.



RSA Demonstration

RSA using the private and public key -- or using only the public key

1 ☒ Choose two prime numbers p and q . The composite number $N = pq$ is the public RSA modulus, and $\phi(N) = (p-1)(q-1)$ is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that $d = e^{-1} \pmod{\phi(N)}$.

☐ For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e .

Prime number entry

Prime number p

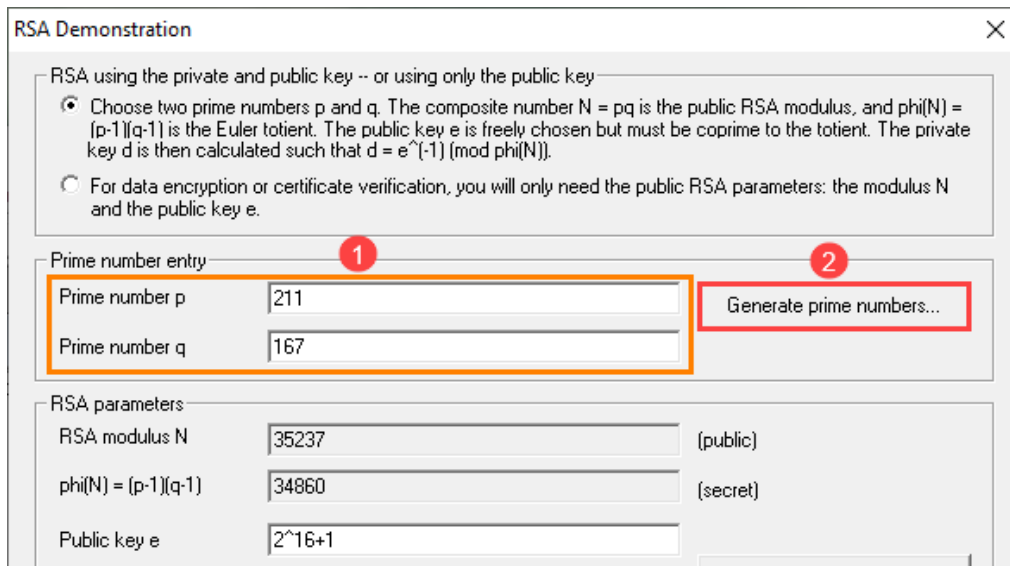
Prime number q

RSA parameters

RSA modulus N (public)

$\phi(N) = (p-1)(q-1)$ (secret)

10. RSA encryption requires the use of prime numbers to help in the calculation of the public and private keys. You can choose 2 prime numbers in the *Prime number entry* fields seen in *item 1*, or you can use the tool to generate prime numbers. In this exercise, we will generate the prime numbers. Do this by clicking the **Generate prime numbers** button seen in *item 2* below.



RSA Demonstration

RSA using the private and public key -- or using only the public key

☒ Choose two prime numbers p and q . The composite number $N = pq$ is the public RSA modulus, and $\phi(N) = (p-1)(q-1)$ is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that $d = e^{-1} \pmod{\phi(N)}$.

☐ For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e .

Prime number entry

1 Prime number p 2

Prime number q

RSA parameters

RSA modulus N (public)

$\phi(N) = (p-1)(q-1)$ (secret)

Public key e

11. The *Private Number Generation* window will appear, and we will use this window to generate our odd numbers. We will leave all the options as they are, but let's take a quick look at their purpose. The *Amount of prime numbers to be generated* radio button seen in *item 1* allows you to choose different options for generating random prime numbers based on a designated range. The *Algorithms for prime number generation* radio buttons seen in *item 2* show you what type of algorithm will be used to generate the prime number. The *Value range of the prime numbers p and q* radio buttons seen in *item 3* allow you to choose between entering unrelated prime numbers or ones that are equal.

Prime Number Generation ✕

Prime numbers play an important role in modern cryptography. Here you can generate primes within a given value range [lower limit, upper limit]. 1

Amount of prime numbers to be generated

☒ Generate two primes randomly from within the value range(s)

☐ Generate all primes within the value range set for p

Separator for the display of the primes:

Algorithms for prime number generation 2

☒ Miller-Rabin Test

☐ Solovay-Strassen Test

☐ Fermat Test

Value range of the prime numbers p and q

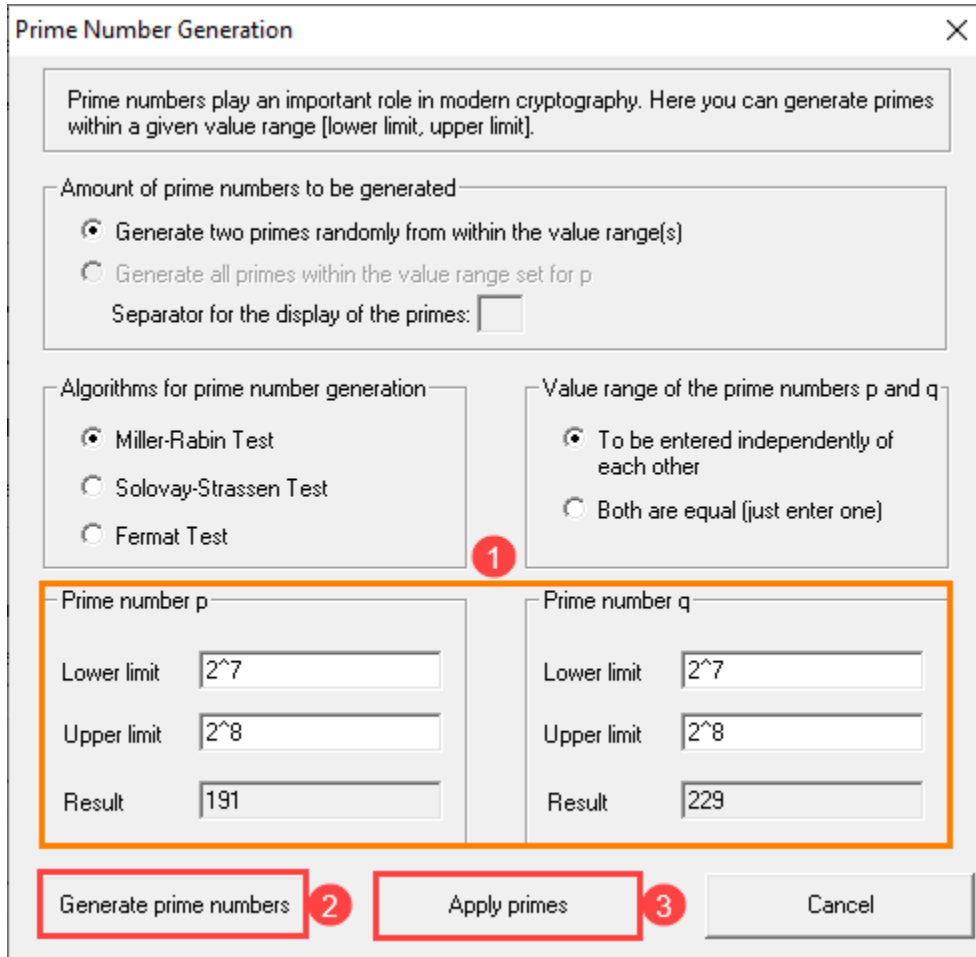
☒ To be entered independently of each other

☐ Both are equal (just enter one)

Prime number p 2

Prime number q 3

12. The *Prime number p* and *Prime number q* fields seen in *item 1* allow you to change the value range and generate larger or smaller prime numbers. We will leave all these options as they are and click the **Generate prime numbers** as seen in *item 2* below. You can press it several times to generate different prime numbers. Once you have 2 prime numbers you want to use, click the **Apply Primes** button seen in *item 3*.

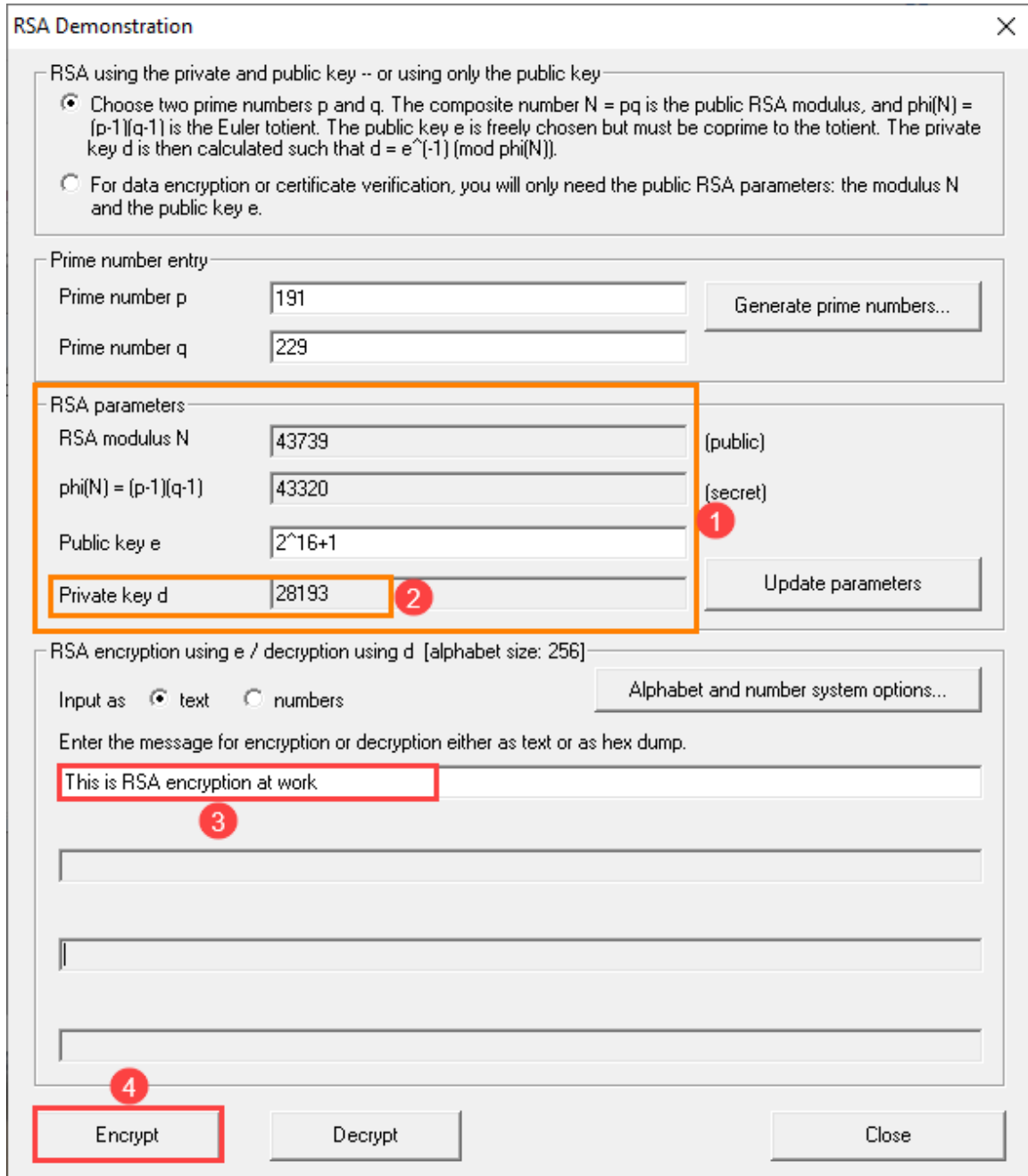


The image shows a 'Prime Number Generation' dialog box with the following sections and controls:

- Prime Number Generation** (Title bar)
- Prime numbers play an important role in modern cryptography. Here you can generate primes within a given value range [lower limit, upper limit].** (Introductory text)
- Amount of prime numbers to be generated**
 - ☒ Generate two primes randomly from within the value range(s)
 - ☐ Generate all primes within the value range set for p
- Separator for the display of the primes:** [] (Text input)
- Algorithms for prime number generation**
 - ☒ Miller-Rabin Test
 - ☐ Solovay-Strassen Test
 - ☐ Fermat Test
- Value range of the prime numbers p and q**
 - ☒ To be entered independently of each other
 - ☐ Both are equal (just enter one)
- Prime number p**
 - Lower limit:
 - Upper limit:
 - Result:
- Prime number q**
 - Lower limit:
 - Upper limit:
 - Result:
- Buttons:**
 - Generate prime numbers** (labeled with a red circle 2)
 - Apply primes** (labeled with a red circle 3)
 - Cancel**

Red annotations in the image include a circle '1' pointing to the 'Algorithms for prime number generation' section, a rectangle '2' around the 'Generate prime numbers' button, and a rectangle '3' around the 'Apply primes' button.

13. You will be taken back to the *RSA Demonstration* window with the new prime numbers entered. The *RSA parameters* fields contain the parameters for the generation of the public and private keys. The private key can be seen in the field seen in *item 2* and is the key that will be needed to decrypt the encrypted data. The *RSA encryption using e / decryption using d* section is where we will enter the data we will encrypt. Let's type the following text as seen in the *Enter the message for encryption or decryption either as text or as hex dump* field seen in *item 3*: **This is RSA encryption at work.** Finally, click the **Encrypt** button seen in *item 4* to see the encryption process.



The screenshot shows the 'RSA Demonstration' window with the following sections and annotations:

- RSA using the private and public key -- or using only the public key**
 - ☒ Choose two prime numbers p and q. The composite number $N = pq$ is the public RSA modulus, and $\phi(N) = (p-1)(q-1)$ is the Euler totient. The public key e is freely chosen but must be coprime to the totient. The private key d is then calculated such that $d = e^{-1} \pmod{\phi(N)}$.
 - ☐ For data encryption or certificate verification, you will only need the public RSA parameters: the modulus N and the public key e.
- Prime number entry**
 - Prime number p: 191
 - Prime number q: 229
 - Generate prime numbers...
- RSA parameters** (highlighted with an orange box)
 - RSA modulus N: 43739 (public)
 - $\phi(N) = (p-1)(q-1)$: 43320 (secret)
 - Public key e: $2^{16}+1$ (annotated with a red circle 1)
 - Private key d: 28193 (annotated with a red circle 2)
 - Update parameters
- RSA encryption using e / decryption using d [alphabet size: 256]**
 - Input as: ☒ text ☐ numbers
 - Alphabet and number system options...
 - Enter the message for encryption or decryption either as text or as hex dump.
 - This is RSA encryption at work (annotated with a red circle 3)
 - Encrypt (annotated with a red circle 4)
 - Decrypt
 - Close

14. The fields below the text that we input reveal the way the data is encrypted. The first field shows the message broken down in blocks, as seen in *item 1* below. In *Item 2*, the *Numbers input in base 10 format* field is a numeric conversion of the letters in the *Input text* field. The last field is the *Encryption into ciphertext* field, and this is the field that shows the ciphertext seen in *item 3*. This ciphertext would be the message that is being transmitted. The recipient would need the private and public keys in order to decrypt this message.

RSA encryption using e / decryption using d [alphabet size: 256]

Input as ☒ text ☐ numbers Alphabet and number system options...

Input text

This is RSA encryption at work.

The Input text will be separated into segments of Size 1 (the symbol '#' is used as separator).

1 T # h # i # s # # i # s # # R # S # A # # e # n # c # r # y # p # t # i # o # n # # a # t # # w # o # r # k

Numbers input in base 10 format.

2 084 # 104 # 105 # 115 # 032 # 105 # 115 # 032 # 082 # 083 # 065 # 032 # 101 # 110 # 099 # 114 # 121 #

Encryption into ciphertext $c[i] = m[i]^e \pmod{N}$

3 35138 # 03462 # 18828 # 22494 # 27493 # 18828 # 22494 # 27493 # 32240 # 13851 # 38081 # 27493 # 1!

Encrypt Decrypt Close

15. Now let's decrypt the ciphertext. To begin, highlight the encrypted data by clicking and sweeping from the beginning of the ciphertext to the end, as seen in *item 1*. Next, right-click on the highlighted text and click the **Copy** from the context menu seen in *items 2 and 3* below.

RSA encryption using e / decryption using d [alphabet size: 256]

Input as ☒ text ☐ numbers Alphabet and number system options...

Input text

This is RSA encryption at work.

The Input text will be separated into segments of Size 1 (the symbol '#' is used as separator).

T # h # i # s # # i # s # # R # S # A # # e # n # c # r # y # p # t # i # o # n # # a # t # # w # o # r # k

Numbers input in base 10 format.

084 # 104 # 105 # 115 # 032 # 105 # 115 # 032 # 082 # 083 # 065 # 032 # 101 # 110 # 099 # 114 # 121 #

Encryption into ciphertext $c[i] = m[i]^e \pmod{N}$

1 35138 # 03462 # 18828 # 22494 # 27493 # 18828 # 22494 # 27493 # 32240 # 13851 # 38081 # 27493 # 1!

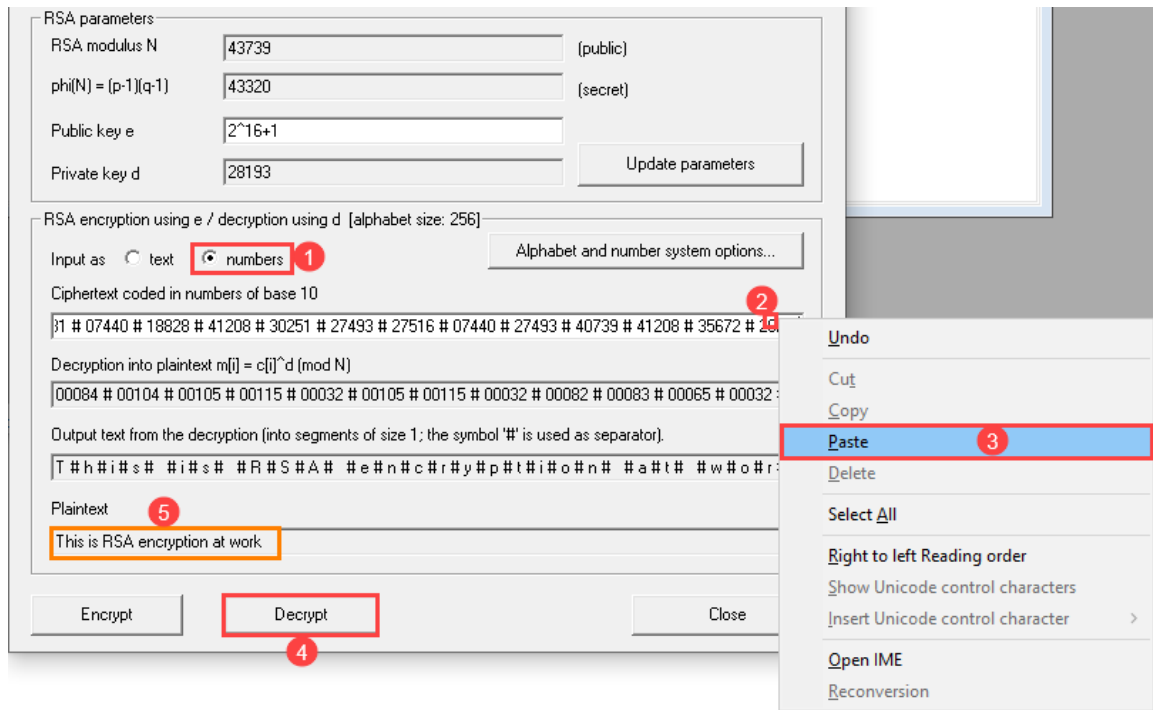
Encrypt Decrypt Close

2

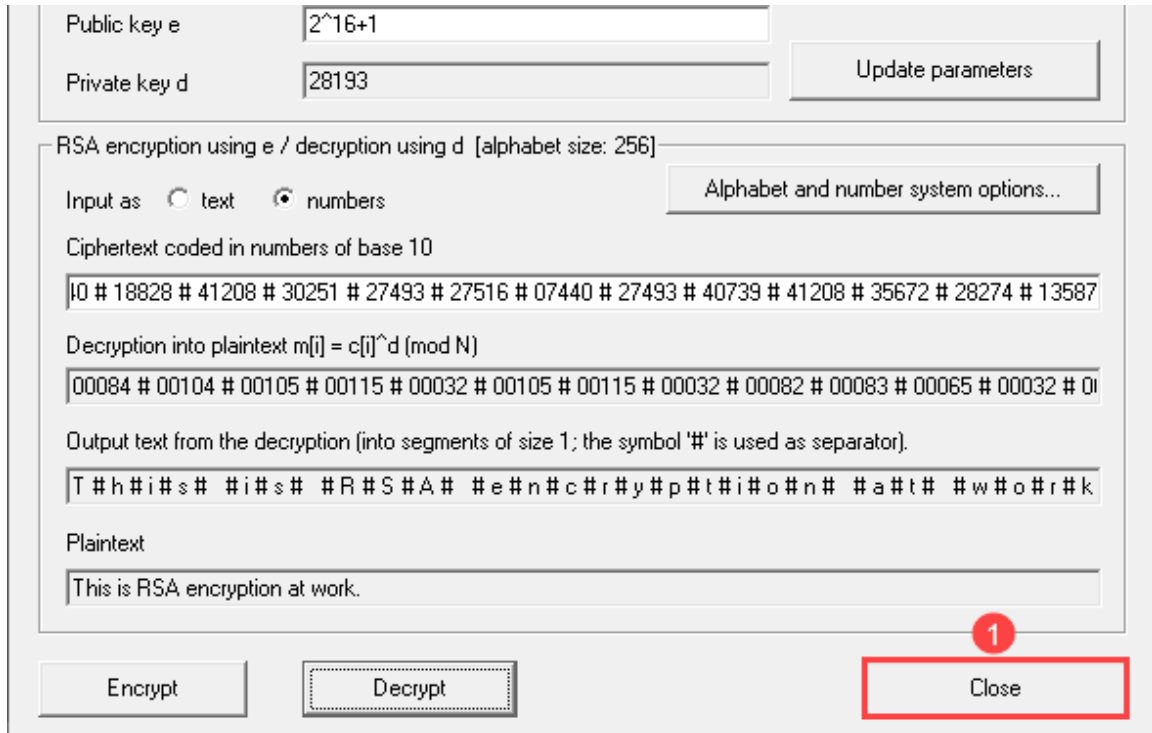
3

Undo
Cut
Copy
Paste
Delete
Select All
Right to left Reading order
Show Unicode control characters
Insert Unicode control character

16. Now that the data is copied to the clipboard, let's delete the data in the *Input text* field by highlighting it and pressing the **Backspace** or **Delete** key. Before pasting the data, click the **numbers** radio button seen in *item 1* to change the type of data we can paste in the *Input text* field. Once you are done, right-click in the *Input text* field and click **Paste** from the context menu, as seen in *items 2* and **3** below. Finally, click the **Decrypt** button seen in *item 4* below. As you can see in the *Plaintext* field in *item 5* below, the message was decrypted.



17. Now that you see how both symmetric and asymmetric encryption works, let's take a look at the hybrid encryption method. Let's start by closing the windows that are currently open by clicking the **Close** button seen in *item 1* below.



Public key e

Private key d

RSA encryption using e / decryption using d [alphabet size: 256]

Input as ☐ text ☒ numbers

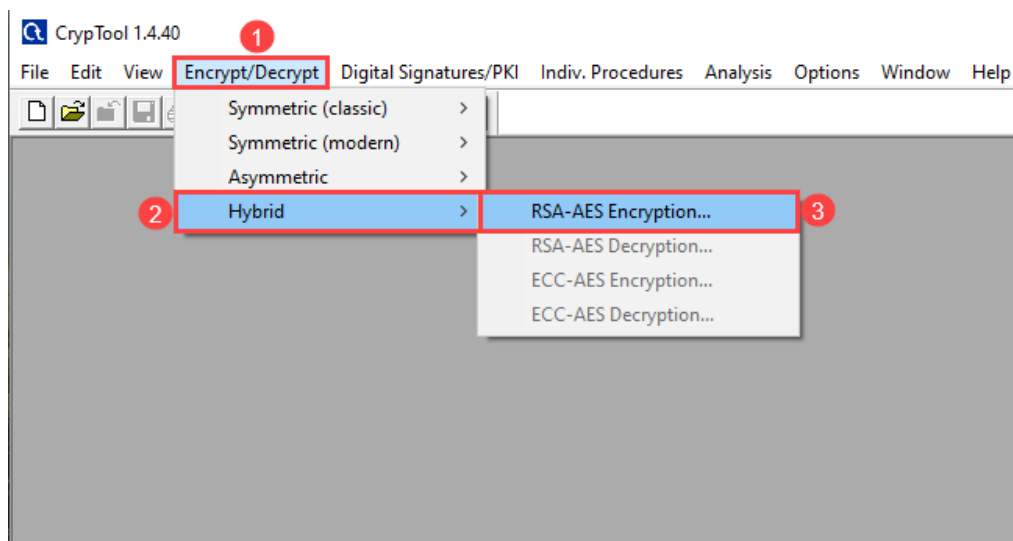
Ciphertext coded in numbers of base 10

Decryption into plaintext $m[i] = c[i]^d \pmod{N}$

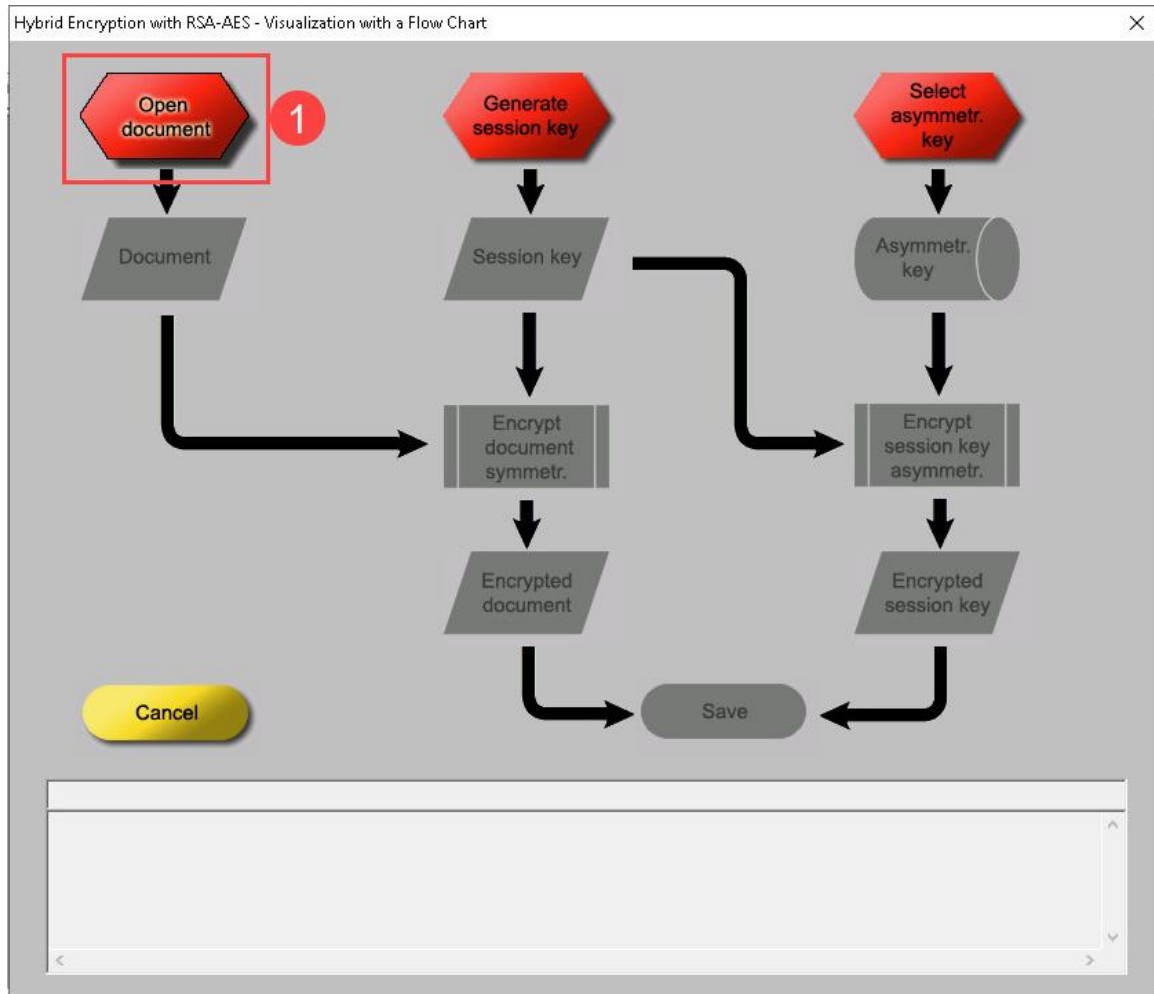
Output text from the decryption (into segments of size 1; the symbol '#' is used as separator).

Plaintext

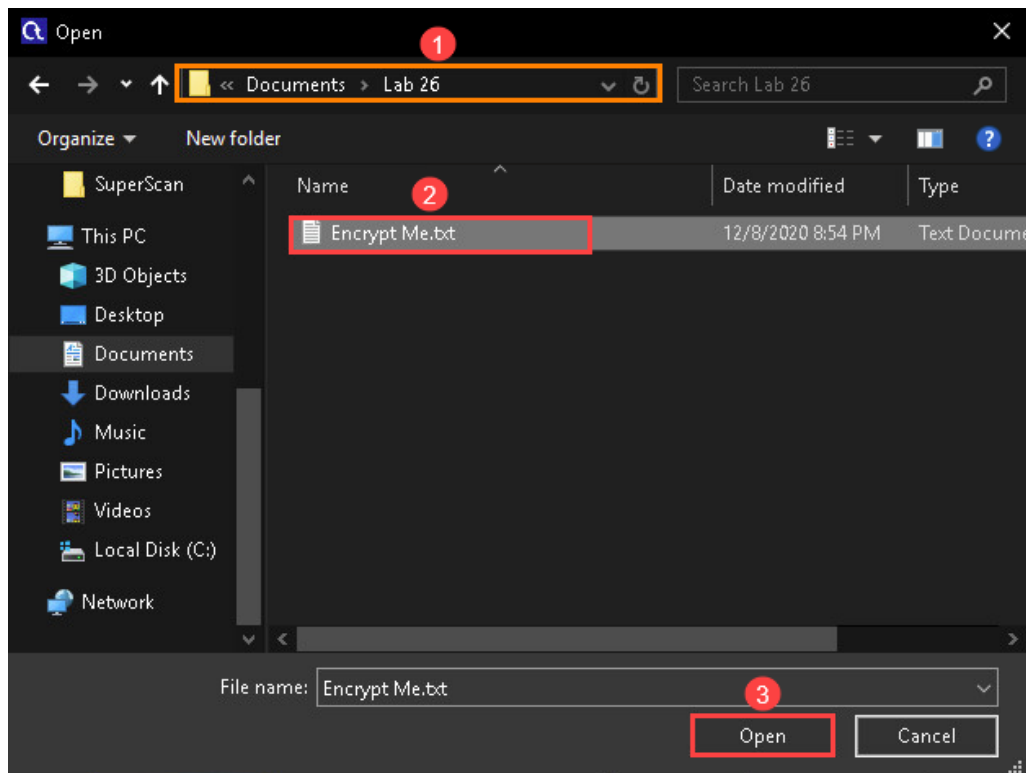
18. Now navigate to **Encrypt/Decrypt > Hybrid > RSA-AES Encryption** as seen in *items 1, 2, and 3* below:



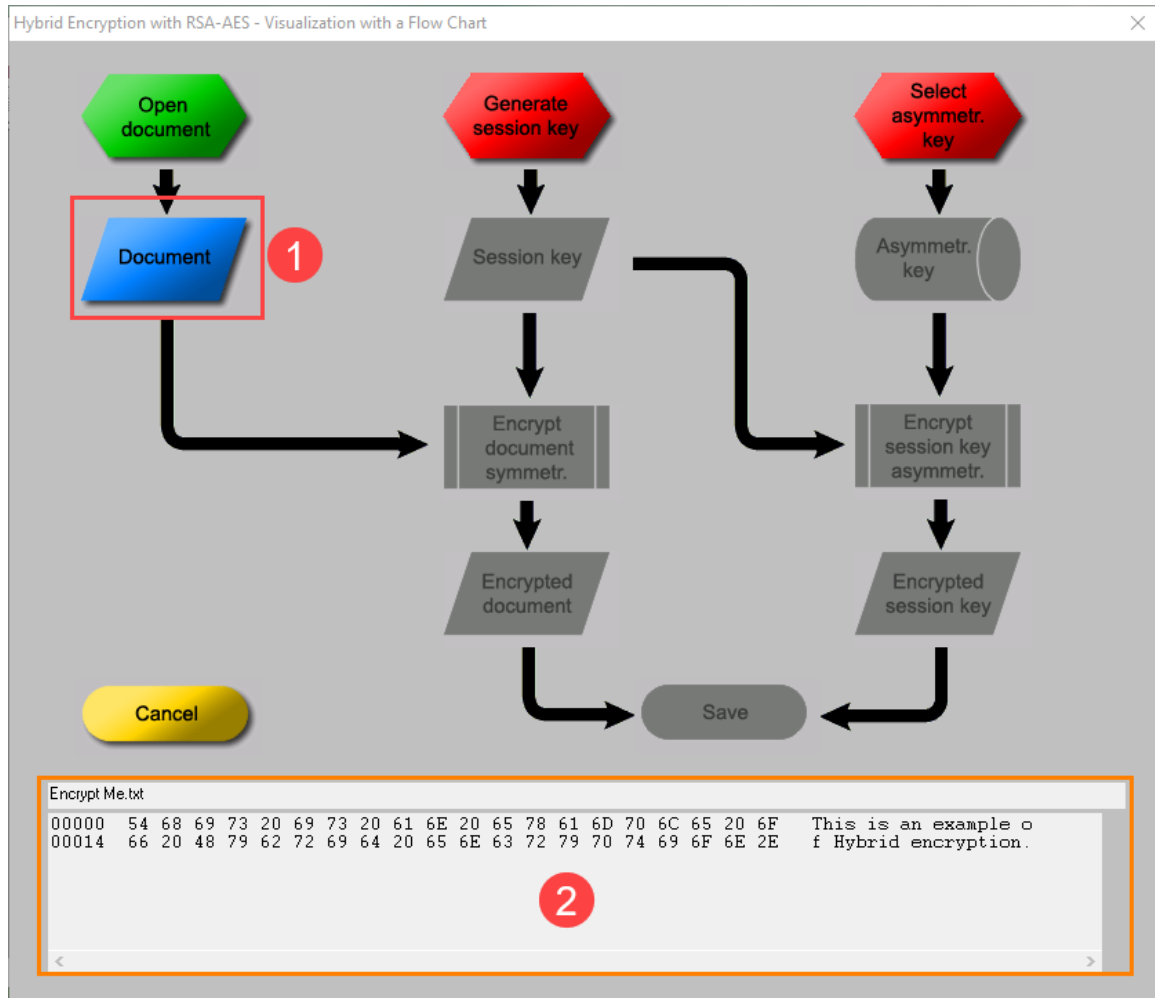
19. The *Hybrid Encryption with RSA-AES - Visualization with a Flow Chart* window will appear. Each step in the flow chart can be clicked to continue the process or reveal data. Let's begin by clicking **Open document** as seen in *item 1* below.



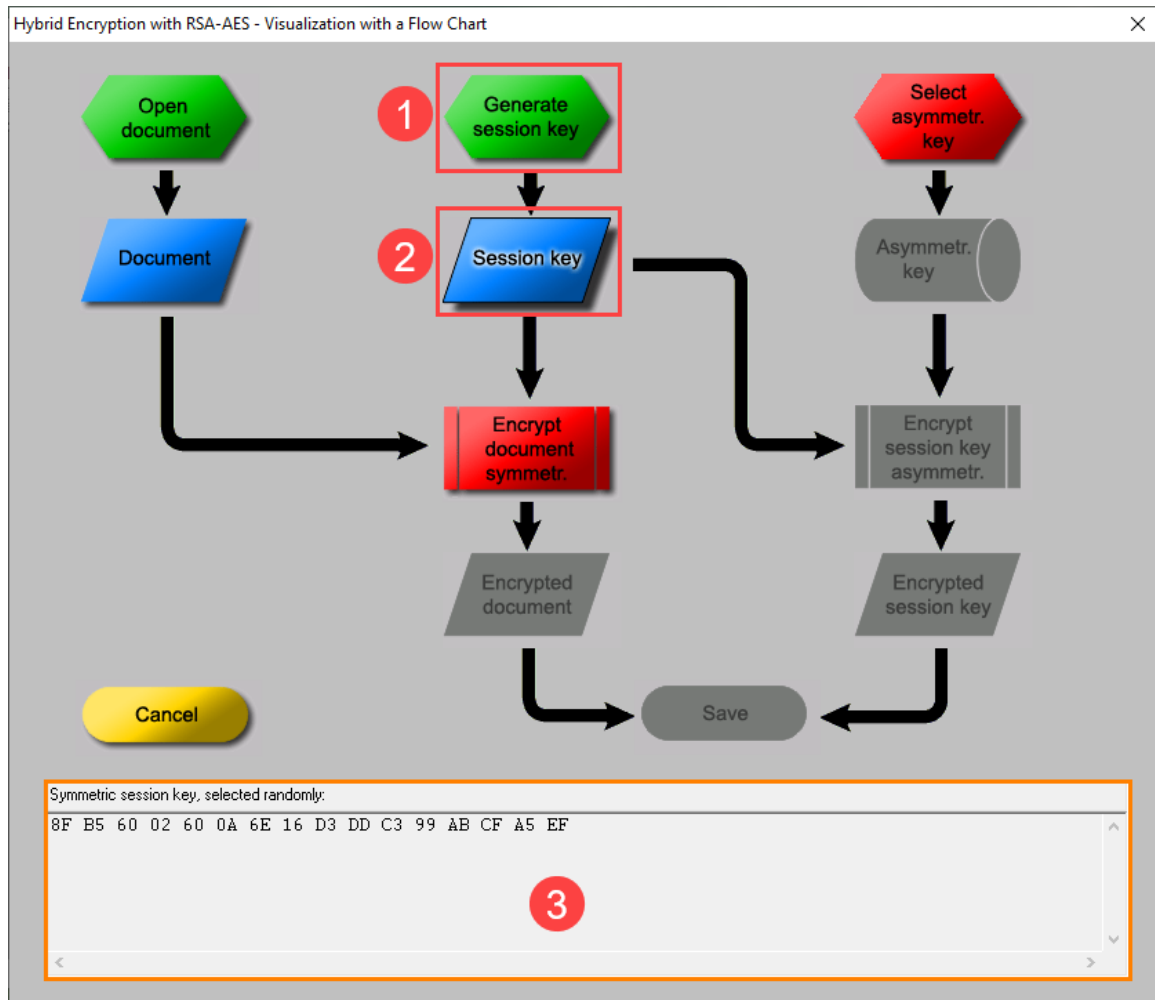
20. The *Open* window will appear; use it to navigate to **Documents > Lab 26** as seen in *item 1* below. Click the single txt file there called **Encrypt Me.txt** and then click the **Open** button seen in *items 2* and *3* below.



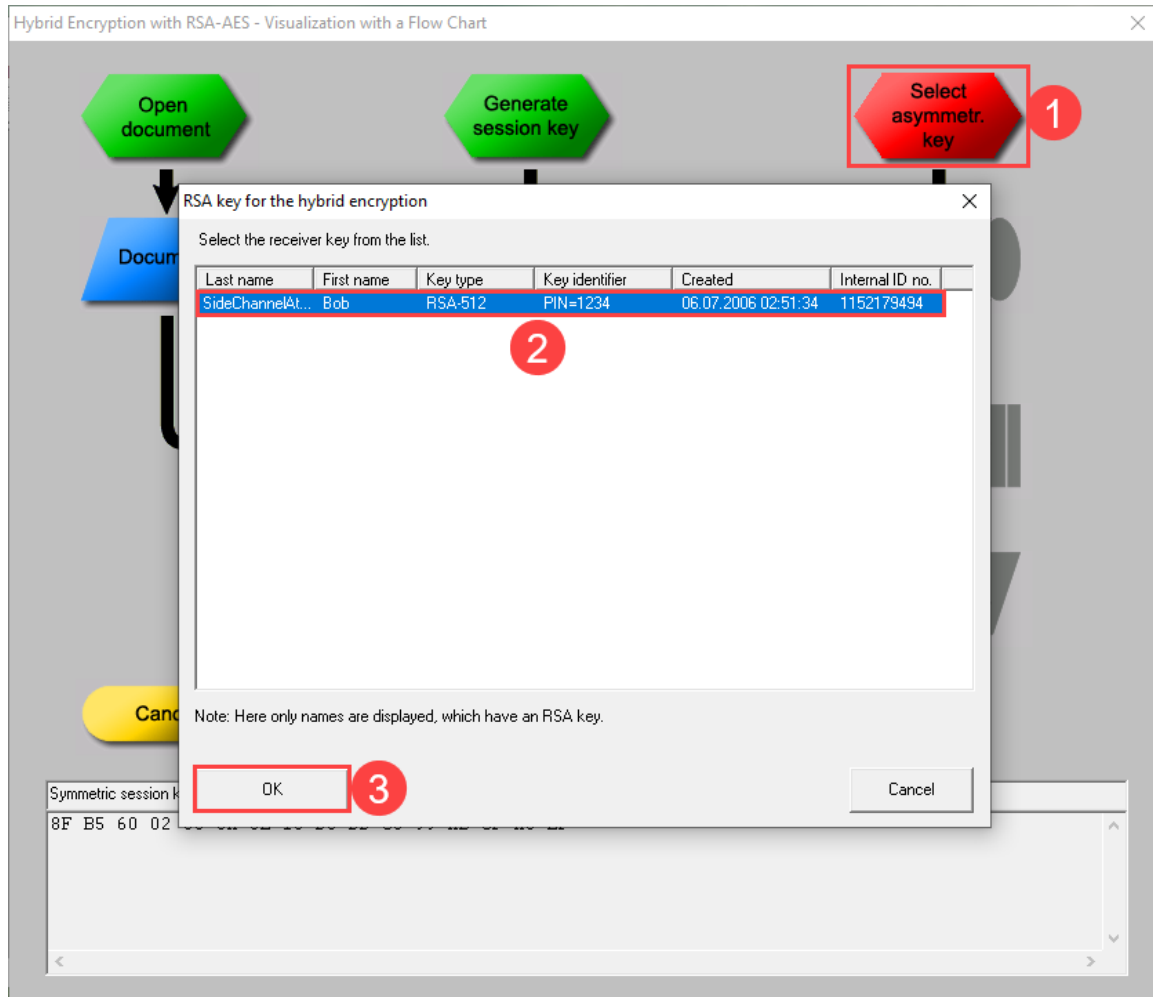
21. You will be taken back to the flow chart, where the *Document* step is now highlighted. Click the **Document** button to reveal the raw unencrypted text in the pane at the bottom, as seen in *items 1* and *2* below.



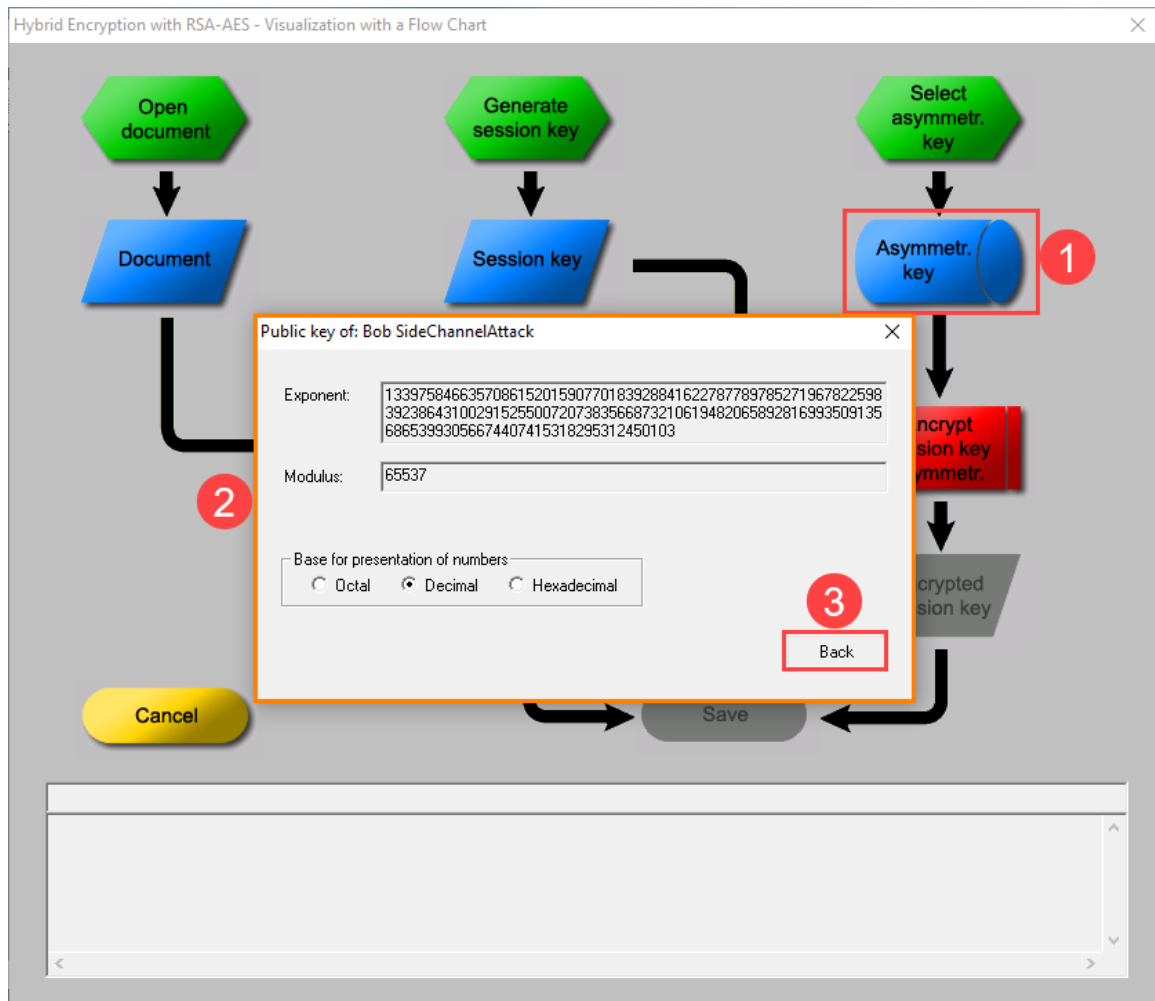
22. The next step is to generate the session key, which will be used to encrypt the document using the AES algorithm. To generate the key, click **Generate session key** button, as seen in *item 1* below. This will generate a random symmetric session key. You will see the *Session key* step highlighted now. Click the **Session key** step to reveal the data in the pane at the bottom of the window seen in *items 2 and 3* below.



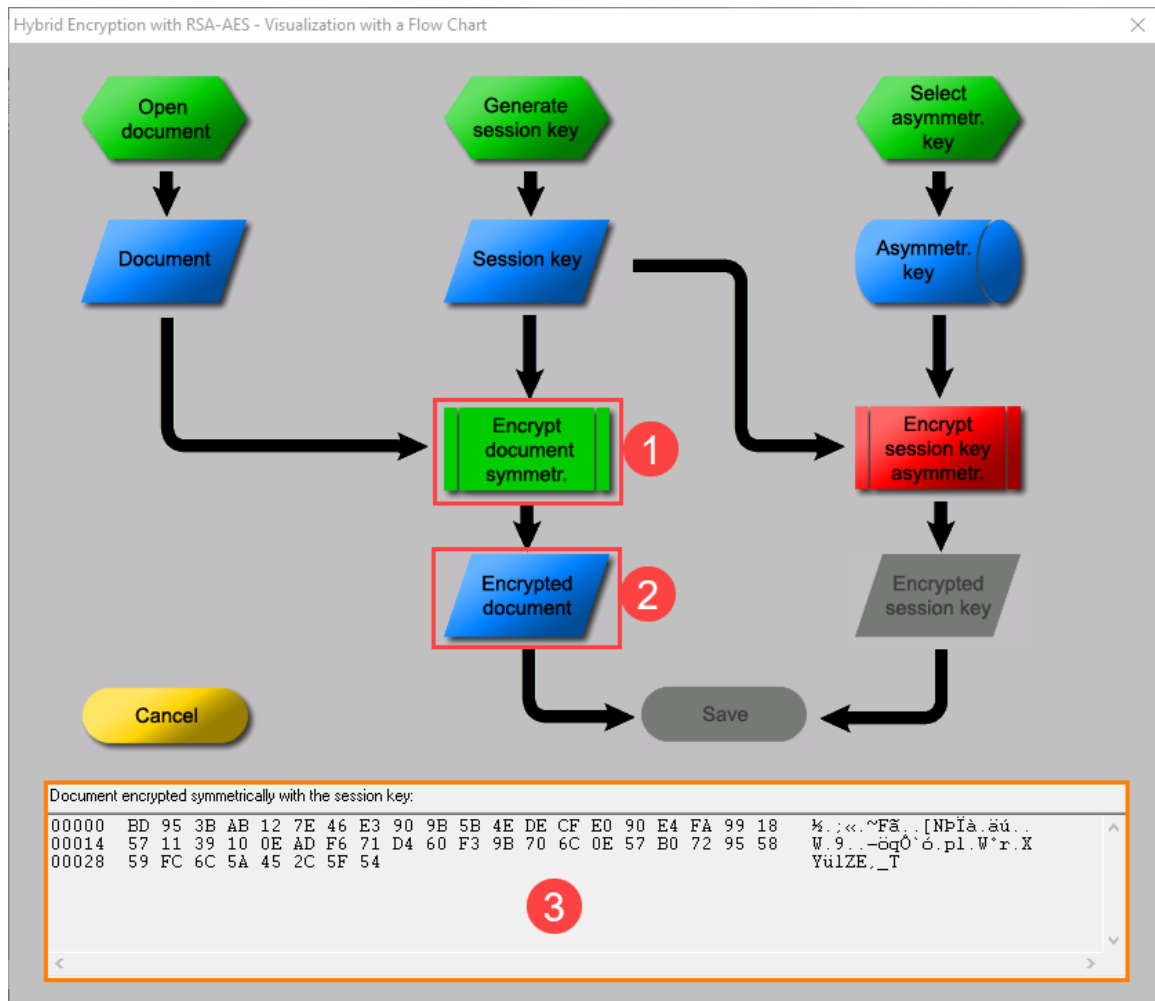
23. Next, let's select an asymmetric key by clicking the **Select asymmetr. key** step, as seen in *item 1*, to open the *RSA key for the hybrid encryption* window. This list contains the public_key of the recipient and is the key that will be used to encrypt the session key we generated in the last step. Click the single entry in this window and click **OK**, as seen in *items 2* and *3* below.



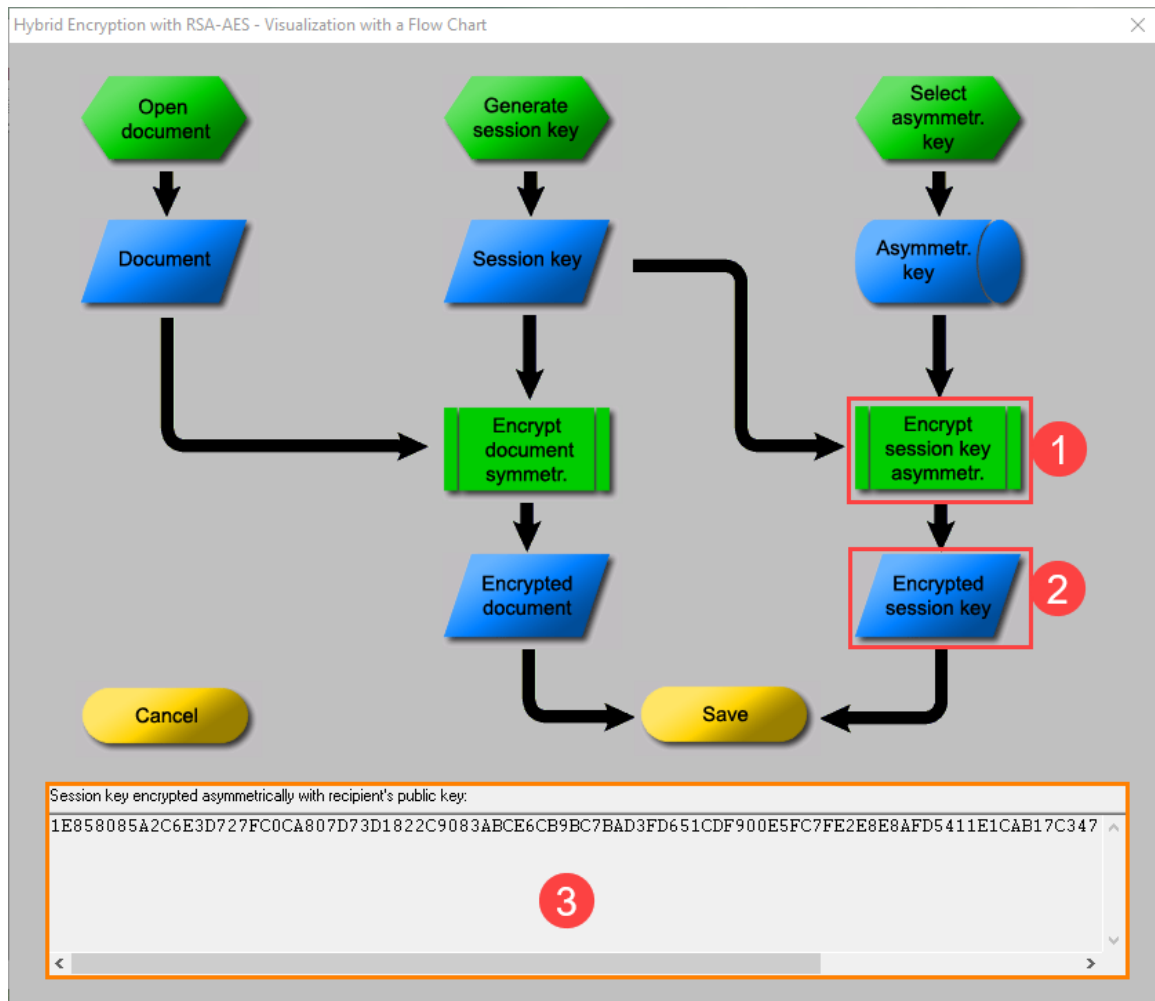
24. You will be taken back to the flow chart once again, but this time the *Asymmetr. key* step is highlighted. Let's click **Asymmetr. key** as seen in *item 1* to see what the public key looks like in *item 2*. Once you are done, close the window by clicking **Back** as seen in *item 3* below.



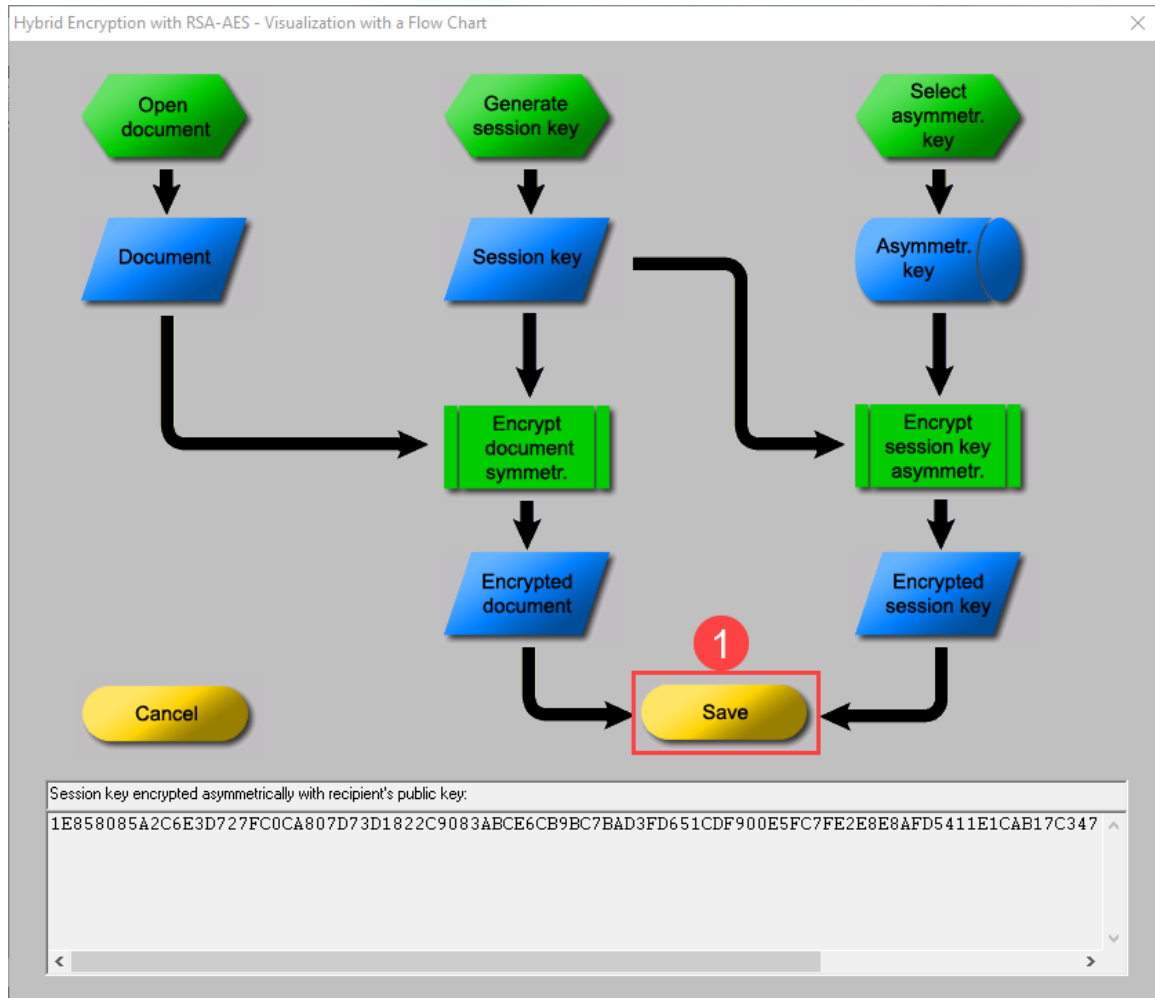
25. Now let's Encrypt the document using the symmetric session key by clicking the **Encrypt document symmetr.** button, seen in *item 1* below. Next, click the **Encrypted document** step to view the encrypted document's contents in the pane at the bottom of the window seen in *items 2* and *3* below.



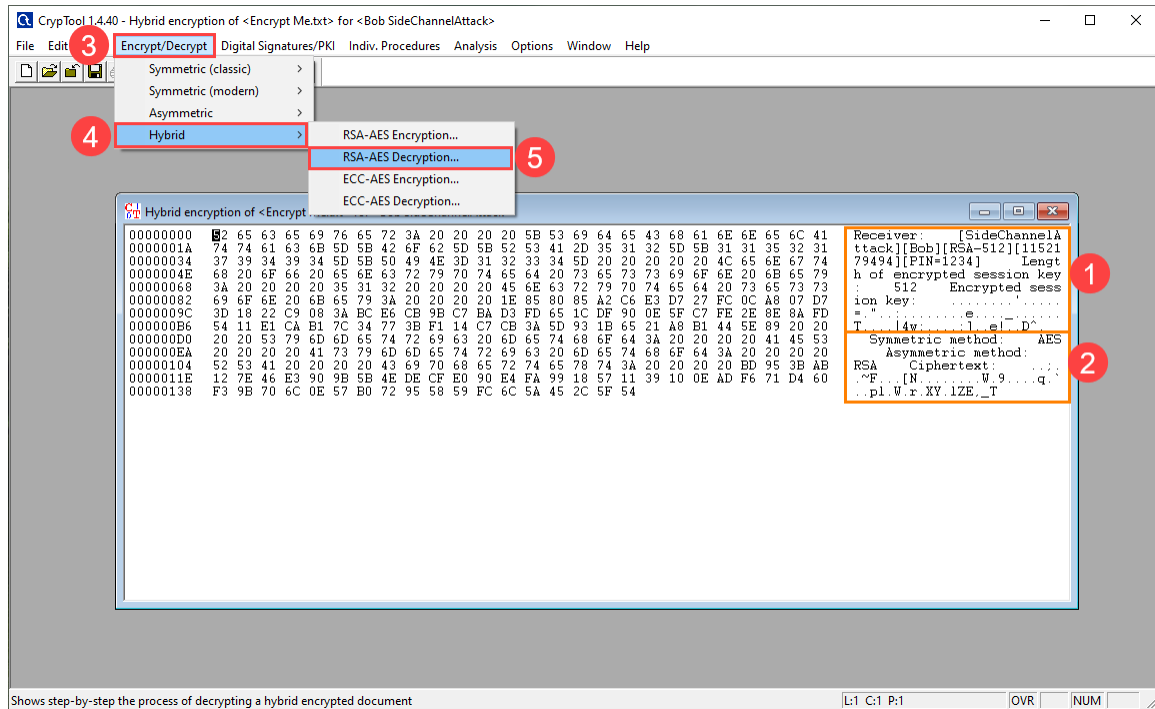
26. Since the document is encrypted using the session key, let's encrypt the session key now, using the public key. To do this, click **Encrypt session key asymmetry** as seen in *item 1*. Next, click the **Encrypted session key** step to reveal it in the pane at the bottom of the window seen in *items 2* and *3*.



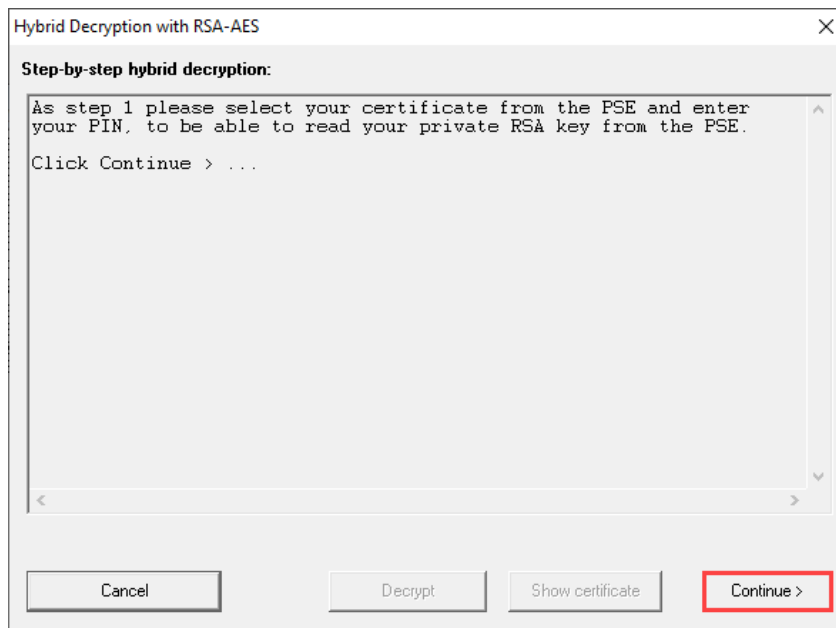
27. The final step is to save the encrypted data. This will create a file that has the encrypted data along with the encrypted session key. To do this, click **Save**, as seen in *item 1* below.



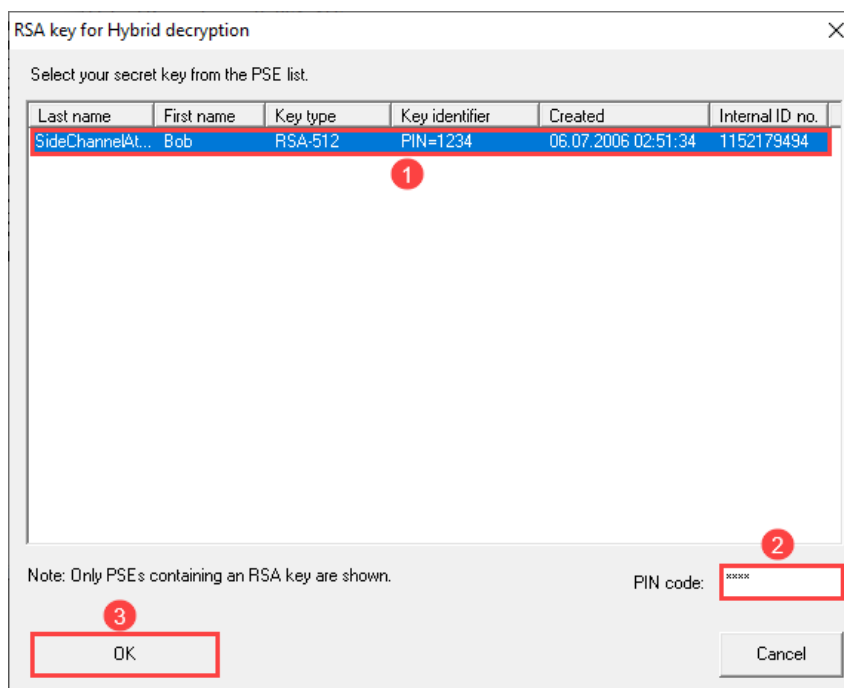
28. You will be taken back to *CrypTool's* main window, where you can see the encrypted data in hexadecimal and plain text. As you can see in *item 1*, the encrypted session key is at the top of the document. In *item 2*, you can see the encrypted data and the encryption methods. Let's decrypt this data now. To do this, navigate to **Encrypt/Decrypt > Hybrid > RSA-AES Decryption** as seen in *items 3, 4*, and *5* below.



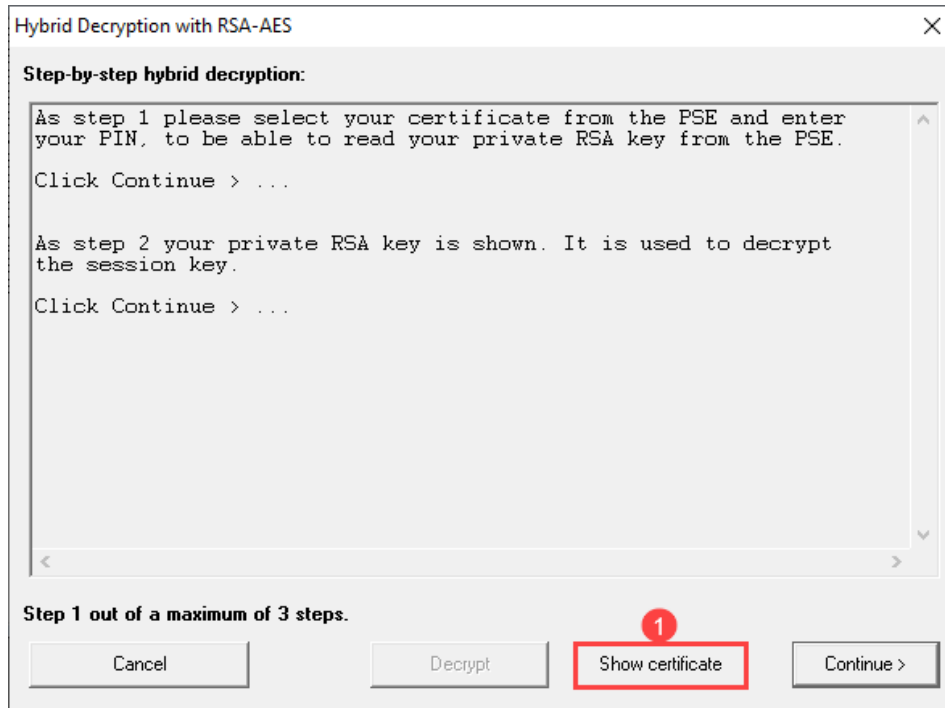
29. The *Hybrid Decryption with RSA-AES* window will appear. The first thing we'll do is select the certificate associated with the private key to start the decryption process.



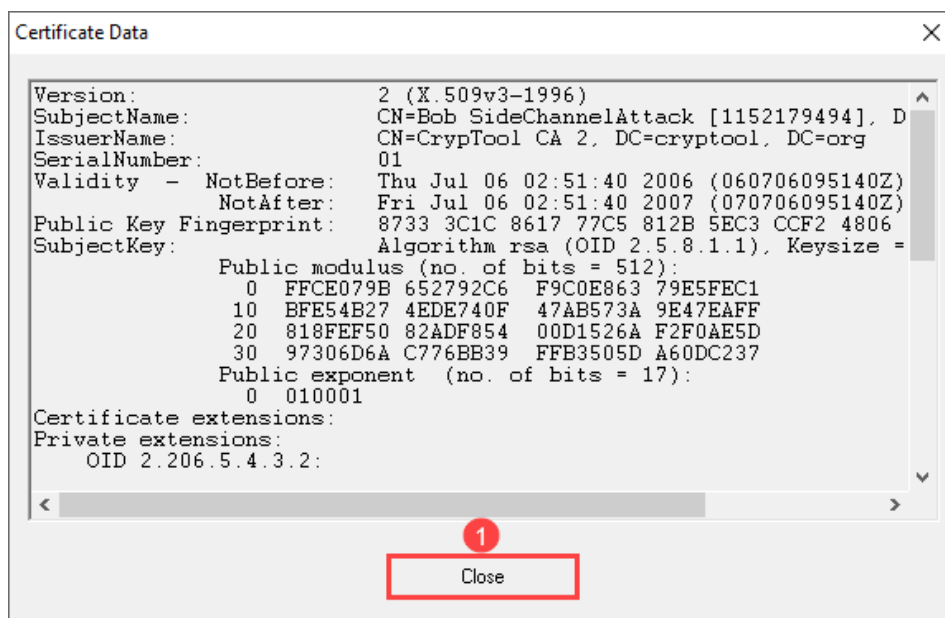
30. The *RSA key for Hybrid decryption* window will appear. Like before, click the one entry in the window as seen in *item 1*. Next, type the PIN from the *Key identifier* row in the *PIN code* field in *item 2*. Once you are done, click **OK** to go back to the *Hybrid Decryption with RSA-AES* window.



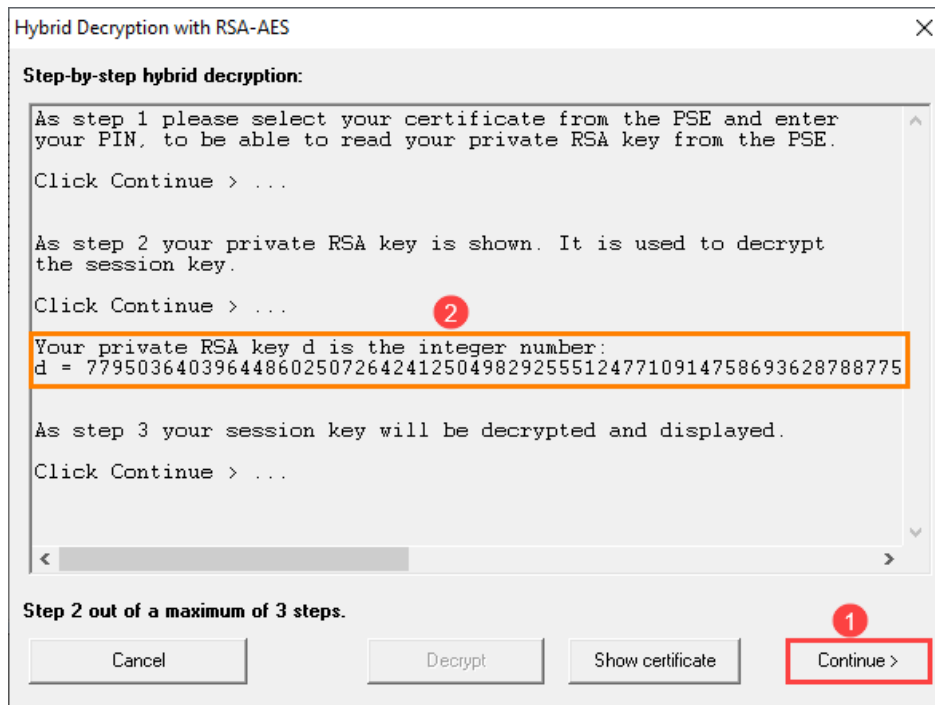
31. In the *Hybrid Decryption with RSA-AES* window, the *Show certificate* button will now be highlighted. Click **Show certificate**, seen in *item 1*, to view the contents of the certificate.



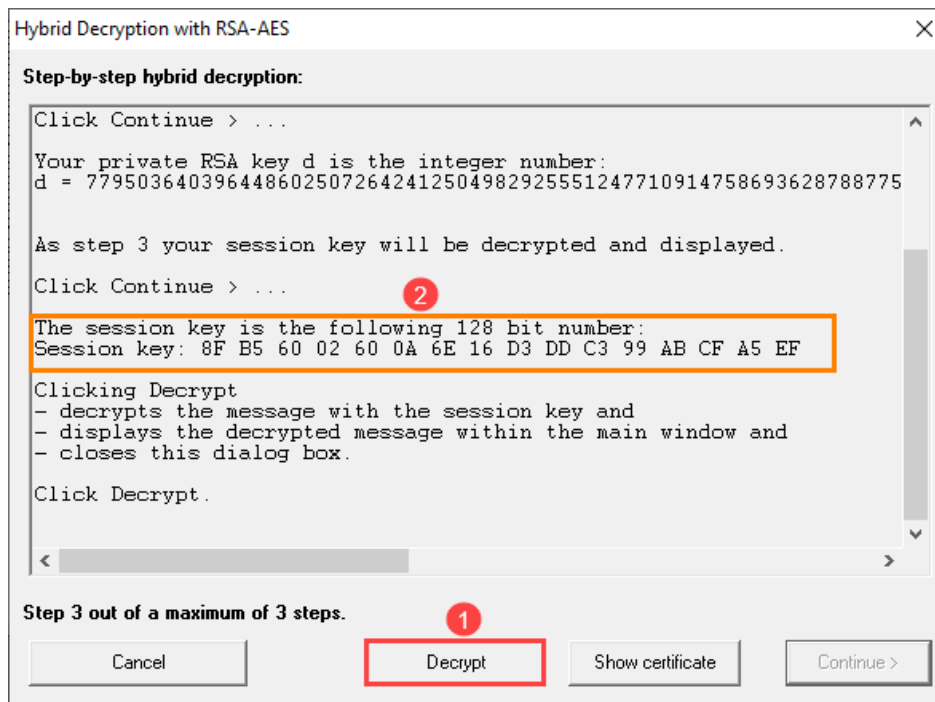
32. The *Certificate Data* window will appear. You can read through it to get an understanding of what the certificate contains. Once you are done, click **Close**, as seen in *item 1* below.



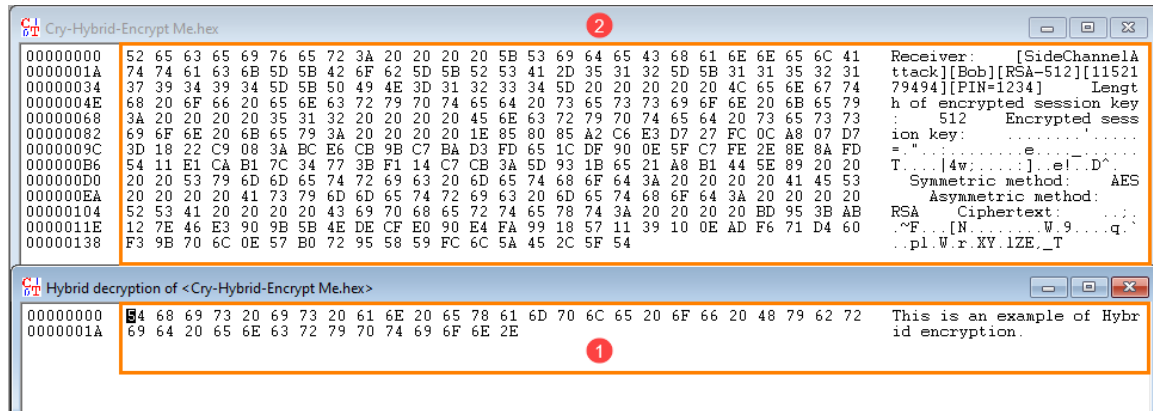
33. Back in the *Hybrid Decryption with RSA-AES* window, click the **Continue** button seen in *item 1* to reveal the private key, which will be revealed in the window as seen in *item 2*. Next, click the **Continue** button seen in *item 1* once again to decrypt the symmetric session key that we used to encrypt the document.



34. You will now see the decrypted session key appear, as seen in *item 1*. This is the key that is needed to decrypt the document. Click **Decrypt** as seen in *item 2* to begin the decryption process.



35. You will be taken back to the main window, where you will see the decrypted data in *item 1*. There will also be the encrypted data in the other window seen in *item 2*. You can use these windows to compare the encrypted data before and after.



Window 2: Cry-Hybrid-Encrypt Me.hex

```

00000000 52 65 63 65 69 76 65 72 3A 20 20 20 20 5B 53 69 64 65 43 68 61 6E 6E 65 6C 41 Receiver: [SideChannel
00000001 74 74 61 63 6B 5D 5B 42 6F 62 5D 5B 52 53 41 2D 35 31 32 5D 5B 31 31 35 32 31 ttrack][Bob][RSA-512][11521
00000003 37 39 34 39 34 5D 5B 50 49 4E 3D 31 32 33 34 5D 20 20 20 20 4C 65 6E 67 74 79494][PIN=1234] Leng
00000004 68 20 6F 66 20 65 6E 63 72 79 70 74 65 64 20 73 65 73 69 6F 6E 20 6B 65 79 h of encrypted session key
00000008 3A 20 20 20 20 35 31 32 20 20 20 20 45 6E 63 72 79 70 74 65 64 20 73 65 73 73 : 512 Encrypted sess
00000009 69 6F 6E 20 6B 65 79 3A 20 20 20 20 1E 85 80 85 A2 C6 E3 D7 27 FC 0C A8 07 D7 ion key: .....
0000000C 3D 18 22 C9 08 3A BC E6 CB 9B C7 BA D3 FD 65 1C DF 90 0E 5F C7 FE 2E 8E 8A FD = "...e....
0000000B 54 11 E1 CA B1 7C 34 77 3B F1 14 C7 CB 3A 5D 93 1B 65 21 A8 B1 44 5E 89 20 20 T...[4w.....]..e...D^
0000000D 20 20 53 79 6D 6D 65 74 72 69 63 20 6D 65 74 68 6F 64 3A 20 20 20 41 45 53 Symmetric method: AES
0000000E 20 20 20 20 41 73 79 6D 6D 65 74 72 69 63 20 6D 65 74 68 6F 64 3A 20 20 20 Asymmetric method:
00000104 52 53 41 20 20 20 43 69 70 68 65 72 74 65 78 74 3A 20 20 20 20 20 RSA Ciphertext:
0000011E 12 7E 46 E3 90 9B 5B 4E DE CF E0 90 E4 FA 99 18 57 11 39 10 0E AD F6 71 D4 60 ^F...[N.....W.9....q...
00000138 F3 9B 70 6C 0E 57 B0 72 95 58 59 FC 6C 5A 45 2C 5F 54 .pl.W.r.XV.IZE._T
  
```

Window 1: Hybrid decryption of <Cry-Hybrid-Encrypt Me.hex>

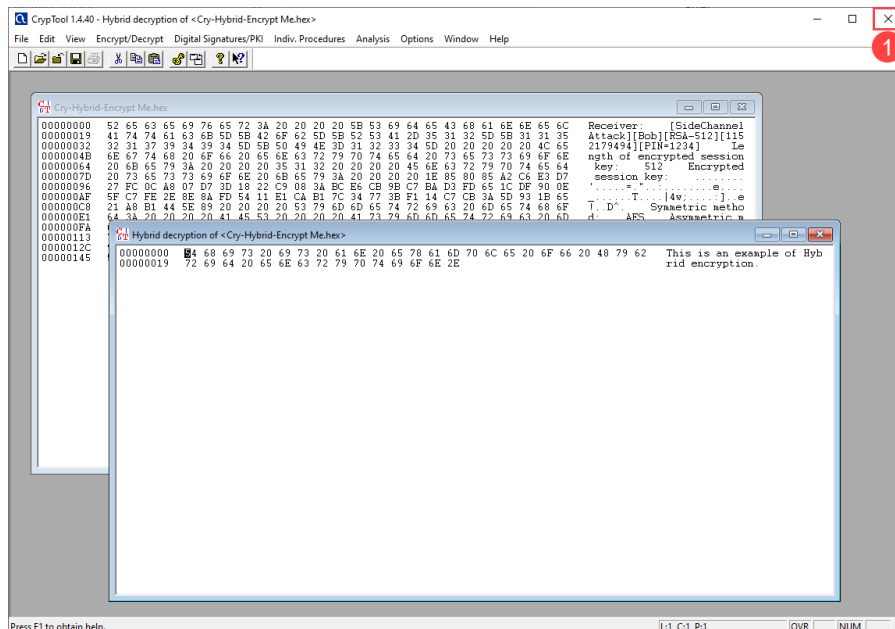
```

00000000 54 68 69 73 20 69 73 20 61 6E 20 65 78 61 6D 70 6C 65 20 6F 66 20 48 79 62 72 This is an example of Hybr
0000001A 69 64 20 65 6E 63 72 79 70 74 69 6F 6E 2E id encryption.
  
```



In this exercise, we covered encrypting data using the symmetric, asymmetric, and hybrid approaches. In the complex hybrid method, we generated a session key then encrypted a document with the session key. Next, we encrypted the session key using a public key. Then, we decrypted the session key using the pre-shared private key. Finally, we decrypted the encrypted document using the decrypted session key.

36. As you can see, the encryption process can be very technical, but it is ideal for protecting data. This lab is now over. To exit *CrypTool*, click the **X** at the top-right corner of the window to close it, as seen in *item 1*.



CrypTool 1.4.40 - Hybrid decryption of <Cry-Hybrid-Encrypt Me.hex>

File Edit View Encrypt/Decrypt Digital Signatures/PKI Indiv. Procedures Analysis Options Window Help

Hybrid decryption of <Cry-Hybrid-Encrypt Me.hex>

```

00000000 52 65 63 65 69 76 65 72 3A 20 20 20 20 5B 53 69 64 65 43 68 61 6E 6E 65 6C Receiver: [SideChannel
00000001 74 74 61 63 6B 5D 5B 42 6F 62 5D 5B 52 53 41 2D 35 31 32 5D 5B 31 31 35 32 31 ttrack][Bob][RSA-512][115
00000003 37 39 34 39 34 5D 5B 50 49 4E 3D 31 32 33 34 5D 20 20 20 20 4C 65 6E 67 74 79494][PIN=1234] Le
00000004 68 20 6F 66 20 65 6E 63 72 79 70 74 65 64 20 73 65 73 69 6F 6E 20 6B 65 79 h of encrypted session key
00000008 3A 20 20 20 20 35 31 32 20 20 20 20 45 6E 63 72 79 70 74 65 64 20 73 65 73 73 : 512 Encrypted sess
00000009 69 6F 6E 20 6B 65 79 3A 20 20 20 20 1E 85 80 85 A2 C6 E3 D7 27 FC 0C A8 07 D7 ion key: .....
0000000C 3D 18 22 C9 08 3A BC E6 CB 9B C7 BA D3 FD 65 1C DF 90 0E 5F C7 FE 2E 8E 8A FD = "...e....
0000000B 54 11 E1 CA B1 7C 34 77 3B F1 14 C7 CB 3A 5D 93 1B 65 21 A8 B1 44 5E 89 20 20 T...[4w.....]..e...D^
0000000D 20 20 53 79 6D 6D 65 74 72 69 63 20 6D 65 74 68 6F 64 3A 20 20 20 41 45 53 Symmetric method: AES
0000000E 20 20 20 20 41 73 79 6D 6D 65 74 72 69 63 20 6D 65 74 68 6F 64 3A 20 20 20 Asymmetric method:
00000104 52 53 41 20 20 20 43 69 70 68 65 72 74 65 78 74 3A 20 20 20 20 20 RSA Ciphertext:
0000011E 12 7E 46 E3 90 9B 5B 4E DE CF E0 90 E4 FA 99 18 57 11 39 10 0E AD F6 71 D4 60 ^F...[N.....W.9....q...
00000138 F3 9B 70 6C 0E 57 B0 72 95 58 59 FC 6C 5A 45 2C 5F 54 .pl.W.r.XV.IZE._T
  
```

Hybrid decryption of <Cry-Hybrid-Encrypt Me.hex>

```

00000000 54 68 69 73 20 69 73 20 61 6E 20 65 78 61 6D 70 6C 65 20 6F 66 20 48 79 62 72 This is an example of Hybr
0000001A 69 64 20 65 6E 63 72 79 70 74 69 6F 6E 2E id encryption.
  
```

Press F1 to obtain help. L:1 C:1 P:1 OVR NUM