ĐẠI HỌC THÁI NGUYÊN

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

BÀI THỰC HÀNH MÔN HỌC

**MẬT MÃ HỌC VÀ ỨNG DỤNG**

Tên môn học: **Mật mã học và ứng dụng**

Số tin chỉ/ĐVHT: **3TC (2LT: 1TH)**

Hệ đào tạo: **Đại học chính quy**

Số tín chỉ thực hành: **1 TC**

Giảng viên biên soạn: **Trịnh Minh Đức**

Bộ môn: **An toàn Hệ thống thông tin**

**Mục lục**

[BÀI THỰC HÀNH 1 5](#_Toc443469369)

[Các hệ mã hóa đối xứng cổ điển: dịch chuyển, thay thế (hoán vị toàn cục), Affine. 5](#_Toc443469370)

[1.1. Mục tiêu 5](#_Toc443469371)

[1.2. Yêu cầu 5](#_Toc443469372)

[1.3. Nội dung thực hành 5](#_Toc443469373)

[1.3.1. Cài đặt Python trên hệ điều hành Windows 5](#_Toc443469374)

[1.3.2. Cài đặt python trên Ubuntu 7](#_Toc443469375)

[1.3.3. Chương trình mẫu hệ mã hóa thay thế hay mã Caesar. 7](#_Toc443469376)

[1.3.4. Chương trình thám mã Caesar 10](#_Toc443469377)

[1.3.5. Bài tập 11](#_Toc443469378)

[1.3.6. Bài tập thực hành 11](#_Toc443469379)

[BÀI THỰC HÀNH 2 12](#_Toc443469380)

[Các hệ mã hóa đối xứng cổ điển: VIGENERE, HILL, hoán vị cục bộ 12](#_Toc443469381)

[2.1. Mục tiêu 12](#_Toc443469382)

[2.2. Yêu cầu 12](#_Toc443469383)

[2.3 Nội dung thực hành 12](#_Toc443469384)

[2.3.1. Chương trình mẫu hệ mã hóa VIGENERE. 12](#_Toc443469385)

[2.3.2. Chương trình thám mã VIGENERE 16](#_Toc443469386)

[2.3.3. Bài tập 18](#_Toc443469387)

[2.3.4. Bài tập thực hành 19](#_Toc443469388)

[BÀI THỰC HÀNH 3. Cài đặt chương trình mã hóa DES 19](#_Toc443469389)

[3.1. Mục tiêu 19](#_Toc443469390)

[3.2. Yêu cầu 19](#_Toc443469391)

[3.3. Nội dung thực hành 19](#_Toc443469392)

[3.3.1. Cài đặt Visual studio 2010. 19](#_Toc443469393)

[3.3.2. Demo xây dựng một ứng dụng  Windows Forms đầu tiên trong Visual Studio 19](#_Toc443469394)

[3.3.3. Bài tập 24](#_Toc443469395)

[3.3.4. Bài tập thực hành 24](#_Toc443469396)

[BÀI THỰC HÀNH 4. Cài đặt chương trình giải mã DES 25](#_Toc443469397)

[3.1. Mục tiêu 25](#_Toc443469398)

[3.2. Yêu cầu 25](#_Toc443469399)

[3.3. Nội dung thực hành 25](#_Toc443469400)

[3.3.1. Bài tập 25](#_Toc443469401)

[3.3.2. Bài tập thực hành 25](#_Toc443469402)

[BÀI THỰC HÀNH 5. Cài đặt hệ mã hóa RSA, Elgamal 26](#_Toc443469403)

[5.1. Mục tiêu 26](#_Toc443469404)

[5.2. Yêu cầu 26](#_Toc443469405)

[5.3. Nội dung thực hành 26](#_Toc443469406)

[5.3.1. Bài tập 26](#_Toc443469407)

[BÀI THỰC HÀNH 6. Cài đặt chữ ký số RSA 27](#_Toc443469408)

[6.1. Mục tiêu 27](#_Toc443469409)

[6.2. Yêu cầu 27](#_Toc443469410)

[6.3. Nội dung thực hành 27](#_Toc443469411)

[6.3.1. Bài tập thực hành 27](#_Toc443469412)

[BÀI THỰC HÀNH 7. Cài đặt chữ ký số Elgamal 28](#_Toc443469413)

[7.1. Mục tiêu 28](#_Toc443469414)

[7.2. Yêu cầu 28](#_Toc443469415)

[7.3. Nội dung thực hành 28](#_Toc443469416)

[7.3.1. Bài tập thực hành 28](#_Toc443469417)

[BÀI THỰC HÀNH 8. Cài đặt chữ ký số DSS 29](#_Toc443469418)

[8.1. Mục tiêu 29](#_Toc443469419)

[8.2. Yêu cầu 29](#_Toc443469420)

[8.3. Nội dung thực hành 29](#_Toc443469421)

[8.3.1. Bài tập thực hành 29](#_Toc443469422)

[BÀI THỰC HÀNH 9. Cài đặt chữ ký không thể phủ định 30](#_Toc443469423)

[9.1. Mục tiêu 30](#_Toc443469424)

[9.2. Yêu cầu 30](#_Toc443469425)

[9.3. Nội dung thực hành 30](#_Toc443469426)

[9.3.1. Bài tập thực hành 30](#_Toc443469427)

[10.1. Mục tiêu 31](#_Toc443469428)

[10.2. Yêu cầu 31](#_Toc443469429)

# **BÀI THỰC HÀNH 1**

# **Các hệ mã hóa đối xứng cổ điển: dịch chuyển, thay thế (hoán vị toàn cục), Affine.**

## **Mục tiêu**

* Củng cố lại kiến thức về các hệ mã hóa đối xứng cổ điển: dịch chuyển, thay thế, Affine.
* Làm quen với môi trường lập trình Python
* Biết cài đặt các hệ mã dịch chuyển, thay thế, Affine.

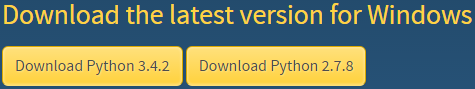
## **Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về các hệ mã.
* Xem trước nội dung bài thực hành ở nhà

## **Nội dung thực hành**

### **Cài đặt Python trên hệ điều hành Windows**

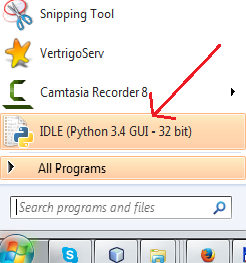
Trước tiên bạn cần lên trang chủ python.org để download phiên bản mới nhất.



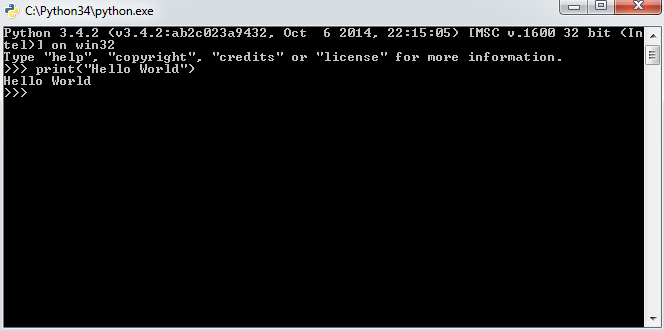
Bạn click vào chữ download python 3.4.2 dành cho windows nhé.

Việc cài đặt Python cũng giống như bạn cài đặt các phần mềm khác bạn cứ click next, next và next nhé.

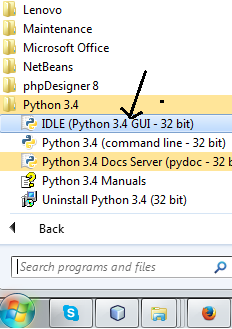
Sau khi cài đặt xong thì ta sẽ có một biểu tượng chạy chương trình như hình dưới đây:



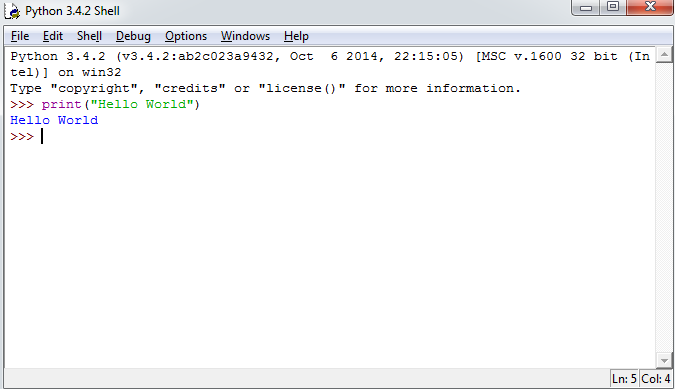
Bạn click vào biểu tượng IDLE (Python 3.4 GUI - 32bit) và sẽ xuất hiện một cửa sổ như hình dưới đây:



Python còn có một chương trình khác nhìn chuyên nghiệp hơn đó là Python GUI như hình dưới đây:



Bây giờ bạn click vào biểu tượng IDLE (Python 3.4 GUI - 32bit) (giống tên ở phía trên nhưng giao diện sẽ khác):



Tại đây chúng ta có thể viết các chương trình bằng ngôn ngữ Python

### **Cài đặt python trên Ubuntu**

Trên ubuntu thì việc cài đặt Python vô cùng dễ dàng với *apt-get*. Bạn vào Command Line gõ lệnh sau: sudo apt-get install python

Sau đó các bạn nhập pass root vào và chờ đợi để trình cài đặt tải python xuống và cài đặt.

### **Chương trình mẫu hệ mã hóa thay thế hay mã Caesar.**

Trước tiên chúng ta cần tải về module *pyperclip.py* từ [*http://invpy.com/pyperclip.py*](http://invpy.com/pyperclip.py), đây là một module chứa nhiều hàm đã xây dựng sẵn hỗ trỡ chúng ta viết các chương trình Python, chúng tã sẽ đặt file này vào cùng thư mục với chương trình nguồn của chúng ta.

Tiếp theo gõ vào những dòng code sau.

*# Caesar Cipher*

*# http://inventwithpython.com/hacking (BSD Licensed)*

import pyperclip

*# the string to be encrypted/decrypted*

message = 'This is my secret message.'

*# the encryption/decryption key*

key = 13

*# tells the program to encrypt or decrypt*

mode = 'encrypt' *# set to 'encrypt' or 'decrypt'*

*# every possible symbol that can be encrypted*

LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

*# stores the encrypted/decrypted form of the message*

translated = ''

*# capitalize the string in message*

message = message.upper()

*# run the encryption/decryption code on each symbol in the message string*

for symbol in message:

if symbol in LETTERS:

*# get the encrypted (or decrypted) number for this symbol*

num = LETTERS.find(symbol) *# get the number of the symbol*

if mode == 'encrypt':

num = num + key

elif mode == 'decrypt':

num = num - key

*# handle the wrap-around if num is larger than the length of*

*# LETTERS or less than 0*

if num >= len(LETTERS):

num = num - len(LETTERS)

elif num < 0:

num = num + len(LETTERS)

*# add encrypted/decrypted number's symbol at the end of translated*

translated = translated + LETTERS[num]

else:

*# just add the symbol without encrypting/decrypting*

translated = translated + symbol

*# print the encrypted/decrypted string to the screen*

print(translated)

*# copy the encrypted/decrypted string to the clipboard*

pyperclip.copy(translated)

Lưu chương trình này lại với tên *caesar.py*, sau đó chạy chương trình chúng ta sẽ nhân được kết quả sau: GUVF VF ZL FRPERG ZRFFNTR.

Đoạn văn bản trên là chuỗi *‘This is my secret message.’* Được mã hóa với mã Caesar sử dụng khóa 13. Để giải mã chỉ việc thay đổi lại giá trị biến *massage* bằng đoạn văn bản đã mã hóa ở trên, đồng thời thay đổi giá trị biến *mode* thành *decrypt*

6. # the string to be encrypted/decrypted

7. message = 'GUVF VF ZL FRPERG ZRFFNTR.'

8.

9. # the encryption/decryption key

10. key = 13

11.

12. # tells the program to encrypt or decrypt

13. mode = 'decrypt' # set to 'encrypt' or 'decrypt'

### **Chương trình thám mã Caesar**

1. *# Caesar Cipher Hacker*

2. *# http://inventwithpython.com/hacking (BSD Licensed)*

3.

4. message = 'GUVF VF ZL FRPERG ZRFFNTR.'

5. LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

6.

7. *# loop through every possible key*

8. for key in range(len(LETTERS)):

9.

10. *# It is important to set translated to the blank string so that the*

11. *# previous iteration's value for translated is cleared.*

12. translated = ''

13.

14. *# The rest of the program is the same as the original Caesar program:*

15.

16. *# run the encryption/decryption code on each symbol in the message*

17. for symbol in message:

18. if symbol in LETTERS:

19. num = LETTERS.find(symbol) *# get the number of the symbol*

20. num = num - key

21.

22. *# handle the wrap-around if num is 26 or larger or less than 0*

23. if num < 0:

24. num = num + len(LETTERS)

25.

26. *# add number's symbol at the end of translated*

27. translated = translated + LETTERS[num]

28.

29. else:

30. *# just add the symbol without encrypting/decrypting*

31. translated = translated + symbol

32.

33. *# display the current key being tested, along with its decryption*

34. print('Key #%s: %s' % (key, translated))

### **Bài tập**

1. Giải mã bản mã sau, giả sử mã hóa Ceasar được sử dụng để mã hóa với k=3: IRXUVFRUHDQGVHYHQBHDUVDJR
2. Mã hóa bản rõ sau: “enemy coming”, dùng phương pháp mã hóa thay thế với khóa hoán vị K là: IAUTMOCSNREBDLHVWYFPZJXKGQ

### **Bài tập thực hành**

1. Thực hiện lại chương trình trên máy, ghi lại các kết quả trung gian. Phân tích từng câu lệnh để xem chúng làm gì? Làm thế nào để mã hóa được các con số? viết lại chương trình để thực hiện việc mã hóa này.
2. Phá mã bản mã sau, giả sử mã hóa Ceasar được sử dụng: CSYEVIXIVQMREXIH ( Cần viết chương trình hỗ trợ phá mã)
3. Viết chương trình mã hóa và giải mã một file văn bản ASCII bằng phương pháp mã hóa Affine

* ***Yêu cầu sinh viên nộp mã nguồn chương trình vào trước buổi thực hành tiệp theo.***

# **BÀI THỰC HÀNH 2**

# **Các hệ mã hóa đối xứng cổ điển: VIGENERE, HILL, hoán vị cục bộ**

## **2.1. Mục tiêu**

* Củng cố lại kiến thức về các hệ mã hóa đối xứng cổ điển: VIGENERE, HILL, hoán vị cục bộ.
* Biết cài đặt các hệ mã dịch chuyển, thay thế, Affine.

## **2.2. Yêu cầu**

* Sinh viên đọc trước nội dung lý thuyết về các hệ mã cổ điển
* Xem trước nội dung bài thực hành ở nhà

## **2.3 Nội dung thực hành**

### **2.3.1. Chương trình mẫu hệ mã hóa VIGENERE.**

Trước tiên chúng ta cần tải về module *pyperclip.py* từ [*http://invpy.com/pyperclip.py*](http://invpy.com/pyperclip.py), đây là một module chứa nhiều hàm đã xây dựng sẵn hỗ trỡ chúng ta viết các chương trình Python, chúng tã sẽ đặt file này vào cùng thư mục với chương trình nguồn của chúng ta.

Tiếp theo gõ vào những dòng code sau.

*1. # Vigenere Cipher (Polyalphabetic Substitution Cipher)*

*2. # http://inventwithpython.com/hacking (BSD Licensed)*

*3.*

4. import pyperclip

*5.*

6. LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

*7.*

8. def main():

*9. # This text can be copy/pasted from http://invpy.com/vigenereCipher.py*

10. myMessage = """Alan Mathison Turing was a British mathematician,logician, cryptanalyst, and computer scientist. He was highly influential in the development of computer science, providing a formalisation of the concepts of "algorithm" and "computation" with the Turing machine. Turing is widely considered to be the father of computer science and artificial intelligence. During World War II, Turing worked for the Government Code and Cypher School (GCCS) at Bletchley Park, Britain's codebreaking centre. For a time he was head of Hut 8, the section responsible for German naval cryptanalysis. He devised a number of techniques for breaking German ciphers, including the method of the bombe, an electromechanical machine that could find settings for the Enigma machine. After the war he worked at the National Physical Laboratory, where he created one of the first designs for a stored-program computer, the ACE. In 1948 Turing joined Max Newman's Computing Laboratory at Manchester University, where he assisted in the development of the Manchester computers and became interested in mathematical biology. He wrote a paper on the chemical basis of morphogenesis, and predicted oscillating chemical reactions such as the Belousov-Zhabotinsky reaction, which were first observed in the 1960s. Turing's homosexuality resulted in a criminal prosecution in 1952, when homosexual acts were still illegal in the United Kingdom. He accepted treatment with female hormones (chemical castration) as an alternative to prison. Turing died in 1954, just over two weeks before his 42nd birthday, from cyanide poisoning. An inquest determined that his death was suicide; his mother and some others believed his death was accidental. On 10 September 2009, following an Internet campaign, British Prime Minister Gordon Brown made an official public apology on behalf of the British government for "the appalling way he was treated." As of May 2012 a private member's bill was before the House of Lords which would grant Turing a statutory pardon if enacted."""

11. myKey = 'ASIMOV'

12. myMode = 'encrypt' *# set to 'encrypt' or 'decrypt'*

13.

14. if myMode == 'encrypt':

15. translated = encryptMessage(myKey, myMessage)

16. elif myMode == 'decrypt':

17. translated = decryptMessage(myKey, myMessage)

18.

19. print('%sed message:' % (myMode.title()))

20. print(translated)

21. pyperclip.copy(translated)

22. print()

23. print('The message has been copied to the clipboard.')

24.

25.

26. def encryptMessage(key, message):

27. return translateMessage(key, message, 'encrypt')

28.

29.

30. def decryptMessage(key, message):

31. return translateMessage(key, message, 'decrypt')

32.

33.

34. def translateMessage(key, message, mode):

35. translated = [] *# stores the encrypted/decrypted message string*

36.

37. keyIndex = 0

38. key = key.upper()

39.

40. for symbol in message: *# loop through each character in message*

41. num = LETTERS.find(symbol.upper())

42. if num != -1: *# -1 means symbol.upper() was not found in LETTERS*

43. if mode == 'encrypt':

44. num += LETTERS.find(key[keyIndex]) *# add if encrypting*

45. elif mode == 'decrypt':

46. num -= LETTERS.find(key[keyIndex]) *# subtract if decrypting*

47.

48. num %= len(LETTERS) *# handle the potential wrap-around*

49.

50. *# add the encrypted/decrypted symbol to the end of translated.*

51. if symbol.isupper():

52. translated.append(LETTERS[num])

53. elif symbol.islower():

54. translated.append(LETTERS[num].lower())

55.

56. keyIndex += 1 *# move to the next letter in the key*

57. if keyIndex == len(key):

58. keyIndex = 0

59. else:

60. *# The symbol was not in LETTERS, so add it to translated as is.*

61. translated.append(symbol)

62.

63. return ''.join(translated)

64.

65.

66. *# If vigenereCipher.py is run (instead of imported as a module) call*

67. *# the main() function.*

68. if \_\_name\_\_ == '\_\_main\_\_':

69. main()

Lưu lại chương trình trên với tên *vigenereCipher.py*

### **2.3.2. Chương trình thám mã VIGENERE**

Để thám mã chúng ta sẽ sử dụng phương pháp tấn công dựa trên từ điển, đây là một kỹ thuật tấn công vét cạn, để giải mã chúng ta sử dụng các từ từ một từ điển như là các khóa, trong đoạn chương trình dưới đây giả sử chỉ xét cho các từ trong tiếng anh. Truy cập địa chỉ <http://invpy.com/dictionary.txt> để tải về file *dictionary.txt,* file này chứa khoảng 45000 từ tiếng anh. Để chương trình có thể hoạt động cần import 2 file *vigenereCipher.py* *detectEnglish.py* (tải xuống từ website http://invpy.com/detectEnglish.py)

1. *# Vigenere Cipher Dictionary Hacker*

2. *# http://inventwithpython.com/hacking (BSD Licensed)*

3.

4. import detectEnglish, vigenereCipher, pyperclip

5.

6. def main():

7. ciphertext = """Tzx isnz eccjxkg nfq lol mys bbqq I lxcz."""

8. hackedMessage = hackVigenere(ciphertext)

9.

10. if hackedMessage != None:

11. print('Copying hacked message to clipboard:')

12. print(hackedMessage)

13. pyperclip.copy(hackedMessage)

14. else:

15. print('Failed to hack encryption.')

16.

17.

18. def hackVigenere(ciphertext):

19. fo = open('dictionary.txt')

20. words = fo.readlines()

21. fo.close()

22.

23. for word in words:

24. word = word.strip() *# remove the newline at the end*

25. decryptedText = vigenereCipher.decryptMessage(word, ciphertext)

26. if detectEnglish.isEnglish(decryptedText, wordPercentage=40):

27. *# Check with user to see if the decrypted key has been found.*

28. print()

29. print('Possible encryption break:')

30. print('Key ' + str(word) + ': ' + decryptedText[:100])

31. print()

32. print('Enter D for done, or just press Enter to continue

breaking:')

33. response = input('> ')

34.

35. if response.upper().startswith('D'):

36. return decryptedText

37.

38. if \_\_name\_\_ == '\_\_main\_\_':

39. main()

### **2.3.3. Bài tập**

1. Mã hóa từ “explanation” bằng phương pháp Vigenere, từ khóa là LEG.

2. Mã hóa thông điệp sau bằng phương pháp hoán vị cục bộ “we are all together” biết khóa k = 24153

3. Xét phương pháp Vigenere. Giả sử biết bản mã “PVRLHFMJCRNFKKW” có bản rõ tương ứng là “networksecurity”. Hãy tìm khóa K.

### **2.3.4. Bài tập thực hành**

1. Thực hiện lại chương trình trên máy, ghi lại các kết quả trung gian. Phân tích từng câu lệnh để xem chúng làm gì?

2. Viết chương trình mã hóa và giải mã một file văn bản ASCII bằng phương pháp mã hóa HILL, hoán vị cục bộ.

* ***Yêu cầu sinh viên nộp mã nguồn chương trình vào trước buổi thực hành tiếp theo.***

# **BÀI THỰC HÀNH 3. Cài đặt chương trình mã hóa DES**

### **3.1. Mục tiêu**

Củng cố lại lý thuyết về hệ mã hóa DES:

* Sơ đồ mã hóa, giải mã.
* Tính khóa K­i.
* Tính Hàm f(Ri , Ki ).

### **3.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về hệ mã DES.
* Xem trước nội dung bài thực hành ở nhà

### **3.3. Nội dung thực hành**

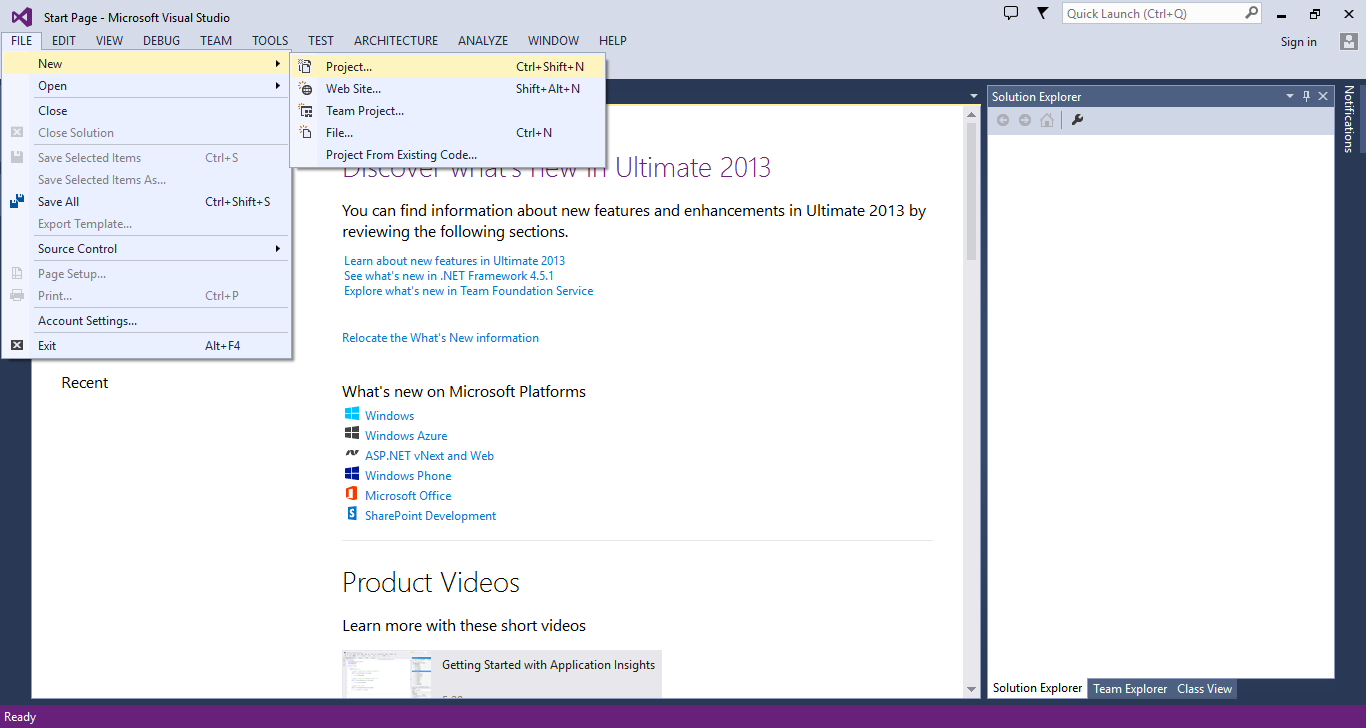
### **3.3.1. Cài đặt Visual studio 2010.**

Các bước cài đặt sẽ được hướng dẫn cụ thể trên phòng máy.

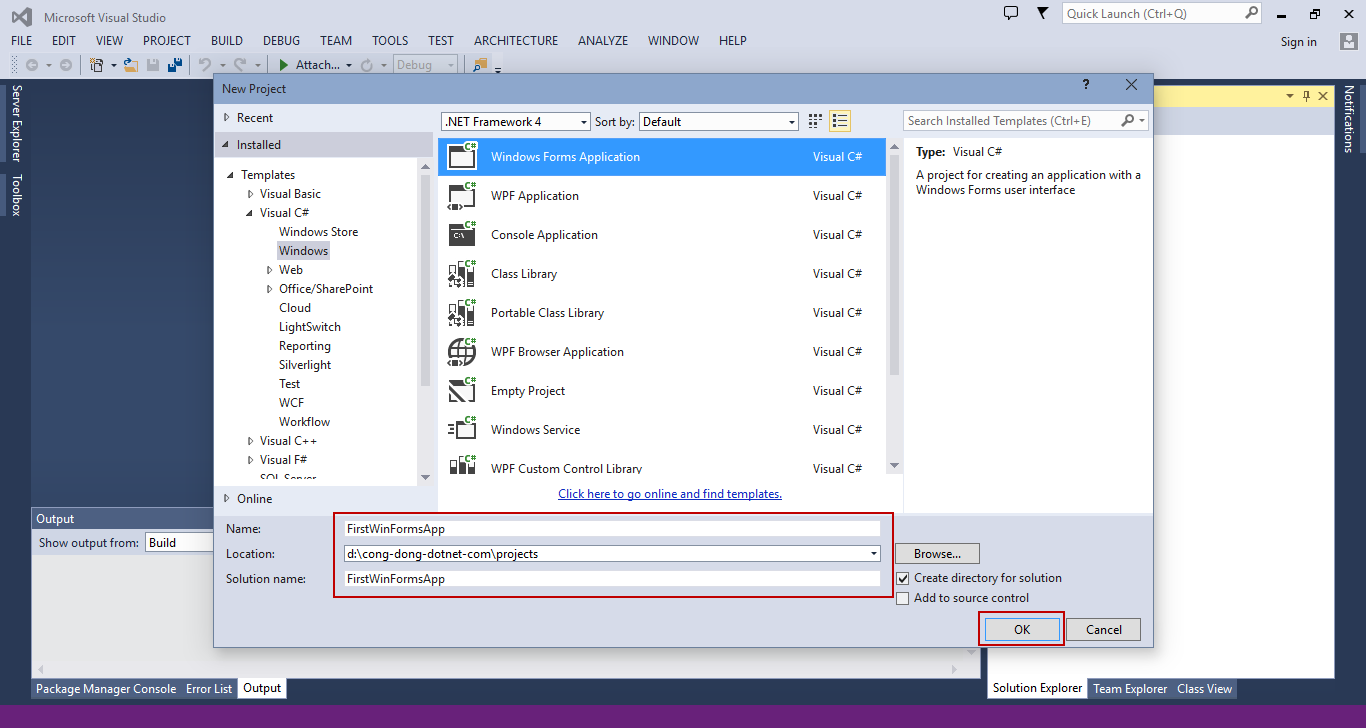
### **3.3.2. Demo xây dựng một ứng dụng**  **Windows Forms đầu tiên trong Visual Studio**

Các bước thực hiện:

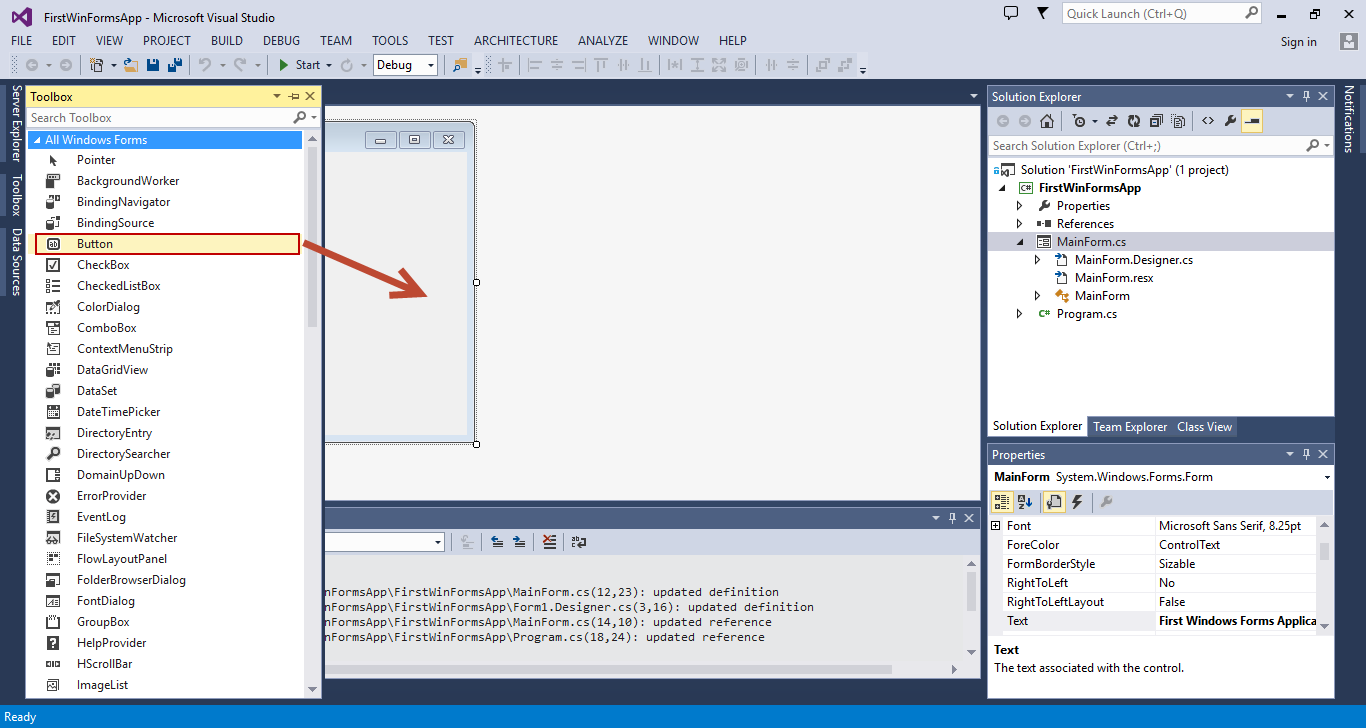
* Bước 1: tạo dự án mới trong VS 2013 bằng cách chọn menu File > New > Project... như hình bên dưới:

[](http://i.imgur.com/ZJbfpet.png)

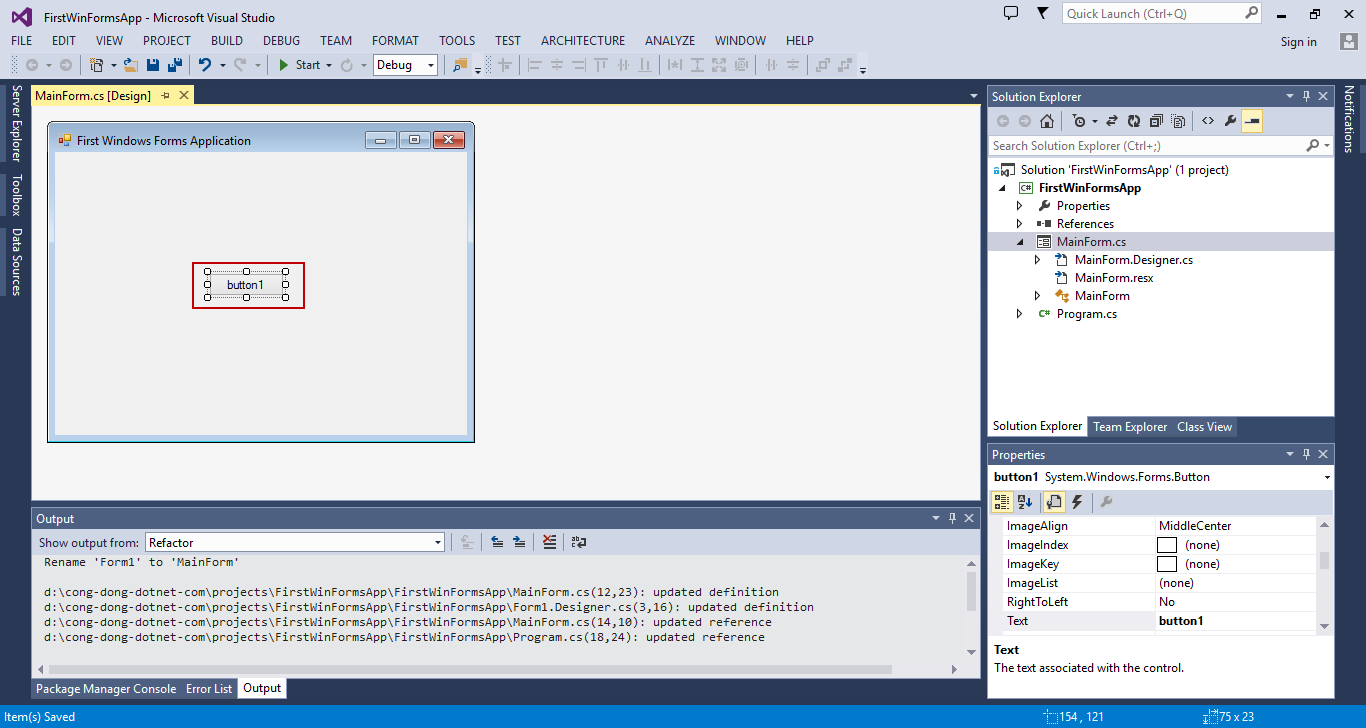
* Bước 2: một hộp thoại **New Project** hiện ra và sau đó bạn chọn như hình bên dưới:

[](http://i.imgur.com/7PJdbdQ.png)

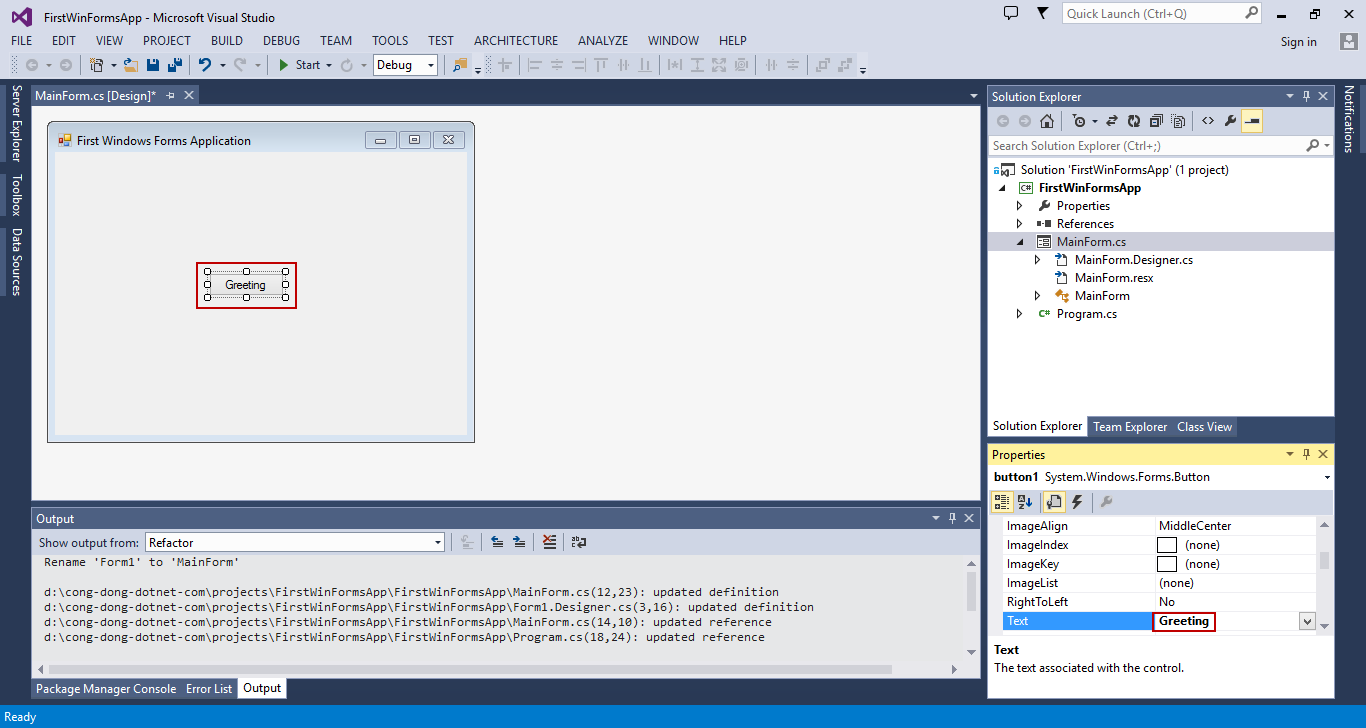
* Bước 3: kéo control Button từ toolbox đến Form trong phần designer như hình dưới đây:

[](http://i.imgur.com/VyXmo3p.png)

* Bước 4: sau khi kéo control Button từ toolbox xong bạn sẽ thấy button xuất hiện trong Form

[](http://i.imgur.com/3ZKVotQ.png)

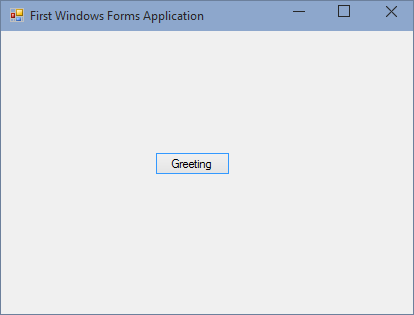
* Bước 5: thay đổi lại text của control Button đến "Greeting" như hình dưới đây:

[](http://i.imgur.com/fDkYJoQ.png)

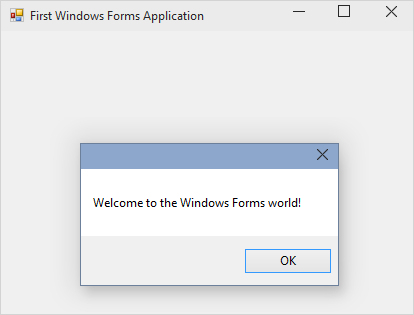
* Bước 6: Double click vào button vừa tạo trong Form để tạo sự kiện Click cho control Button đó
* Bước 7: Thêm đoạn code này vào sự kiện Click của control Button

MessageBox.Show("Welcome to the Windows Forms world!");

* Bước 8: Bấm Ctrl+F5 để chạy ứng dụng Windows Forms.
* Bước 9: Chương trình Windows Forms đầu tiên sẽ hiển thị giống như hình dưới đây:

[](http://i.imgur.com/jvUKrVe.png)

* Bước 10: Bạn click vào button Greeting. Một hộp thoại sẽ xuất hiện như hình dưới đây:

[](http://i.imgur.com/PVVXnFE.png)

* Bước 11: Hoàn tất.

### **3.3.3. Bài tập**

1. Xét mã TinyDES trong đó khóa K là 10100100. Hãy tính bản mã trong trường hợp bản rõ là P = 01001011

### **3.3.4. Bài tập thực hành**

1. Tìm hiểu về thư viện mã hóa của môi trường lập trình .NET (namespace

System.Security.Cryptography) Xây dựng một chương trình Windows Form có các chức năng theo mẫu sau:

**Menu chính.**

L. Lập mã.

K. Kết thúc.

L. **Menu Lập mã.**

1. Nhập bản tin (Xâu ký tự): RÕ\_CHỮ.

2. Chuyển RÕ\_CHỮ =====> RÕ\_SỐ.

3. Chuyển RÕ\_SỐ =====> MÃ\_SỐ.

4. Chuyển MÃ\_SỐ =====> MÃ\_CHỮ.

# **BÀI THỰC HÀNH 4. Cài đặt chương trình giải mã DES**

### **3.1. Mục tiêu**

Củng cố lại lý thuyết về hệ mã hóa DES:

* Sơ đồ mã hóa, giải mã.
* Tính khóa K­i.
* Tính Hàm f(Ri , Ki ).

### **3.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về hệ mã DES.
* Xem trước nội dung bài thực hành ở nhà

### **3.3. Nội dung thực hành**

### **3.3.1. Bài tập**

Giải mã bản mã sau **C=** **85E813540F0AB405** biết khóa **K** = **133457799BBCDFF1**

### **3.3.2. Bài tập thực hành**

Sử dụng kết quả của bài thực hành 3 và thư viện mã hóa của môi trường lập trình .NET (namespace System.Security.Cryptography), xây dựng một chương trình Windows Form có các chức năng theo mẫu sau:

**Menu chính.**

L. Lập mã.

G. Giải mã.

K. Kết thúc.

L. **Menu Lập mã.**

1. Nhập bản tin (Xâu ký tự): RÕ\_CHỮ.

2. Chuyển RÕ\_CHỮ =====> RÕ\_SỐ.

3. Chuyển RÕ\_SỐ =====> MÃ\_SỐ.

4. Chuyển MÃ\_SỐ =====> MÃ\_CHỮ.

G. **Menu Giải mã.**

1. Nhập bản tin (Xâu ký tự): MÃ\_CHỮ.

2. Chuyển MÃ\_CHỮ =====> MÃ\_SỐ.

3. Chuyển MÃ\_SỐ =====> RÕ\_SỐ.

4. Chuyển RÕ\_SỐ =====> RÕ\_CHỮ.

* ***Yêu cầu sinh viên gửi mã nguồn chương trình vào trước buổi thực hành tiếp theo.***

# **BÀI THỰC HÀNH 5. Cài đặt hệ mã hóa RSA, Elgamal**

## **5.1. Mục tiêu**

* Củng cố lại lý thuyết về hệ mã hóa RSA, Elgamal: Sơ đồ mã hóa, giải mã.

## **5.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về hệ mã RSA, Elgamal.
* Xem trước nội dung bài thực hành ở nhà

## **5.3. Nội dung thực hành**

### **5.3.1. Bài tập**

1. Cho a = 13, p = 20. Tìm giá trị nghịch đảo của a trong phép modulo p dùng thuật toán Euclid mở rộng .

2. Áp dụng thuật toán bình phương liên tiếp tính *721 mod 13*

3. Cho p = 5, q= 11, e = 7. Tính khóa riêng (d, N) trong phương pháp RSA.

4. Alice chọn p = 7, q = 11, e = 17, Bob chọn p = 11, q = 13, e = 11:

a. Tính khóa riêng KRA của Alice và KRB của Bob

b. Alice muốn gởi cho Bob bản tin x = 9. Hãy thực hiện quá trình mã hóa và giải mã

**5.3.2. Bài tập thực hành**

1. Xây dựng một chương trình mã hóa theo mẫu sau (có thể sử dụng thư viện mã hóa của môi trường lập trình .NET (namespace System.Security.Cryptography)

**Menu chính.**

L. Lập mã RSA.

G. Lập mã Elgamal.

K. Kết thúc.

L. **Menu Lập mã RSA.**

1. Nhập bản tin (Xâu ký tự): RÕ\_CHỮ.

2. Chuyển RÕ\_CHỮ =====> RÕ\_SỐ.

3. Chuyển RÕ\_SỐ =====> MÃ\_SỐ.

4. Chuyển MÃ\_SỐ =====> MÃ\_CHỮ.

G. **Menu Lập mã Elgamal.**

1. Nhập bản tin (Xâu ký tự): RÕ\_CHỮ.

2. Chuyển RÕ\_CHỮ =====> RÕ\_SỐ.

3. Chuyển RÕ\_SỐ =====> MÃ\_SỐ.

4. Chuyển MÃ\_ =====> MÃ\_CHỮ. SỐ

* ***Yêu cầu sinh viên gửi mã nguồn chương trình vào trước buổi thực hành tiếp theo.***

# **BÀI THỰC HÀNH 6. Cài đặt chữ ký số RSA**

## **6.1. Mục tiêu**

* Củng cố lại lý thuyết về hệ mã hóa RSA,sơ đồ chữ ký RSA.

## **6.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về hệ mã RSA, sơ đồ chữ ký RSA.
* Xem trước nội dung bài thực hành ở nhà

## **6.3. Nội dung thực hành**

### **6.3.1. Bài tập thực hành**

Xây dựng chương trình ký số RSA thực hiện các chức năng sau:

**Menu chính.**

S. Ký số.

V. Kiểm thử chữ ký.

K. Kết thúc.

* ***Yêu cầu sinh viên gửi mã nguồn chương trình vào trước buổi thực hành tiếp theo.***

# **BÀI THỰC HÀNH 7. Cài đặt chữ ký số Elgamal**

## **7.1. Mục tiêu**

Củng cố lại lý thuyết về hệ mã hóa Elgamal,sơ đồ chữ ký Elgamal.

## **7.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về hệ mã Elgamal, sơ đồ chữ ký Elgamal.
* Xem trước nội dung bài thực hành ở nhà

## **7.3. Nội dung thực hành**

### **7.3.1. Bài tập thực hành**

Xây dựng chương trình ký số Elgamal thực hiện các chức năng sau:

**Menu chính.**

S. Ký số.

V. Kiểm thử chữ ký.

K. Kết thúc.

* ***Yêu cầu sinh viên gửi mã nguồn chương trình vào trước buổi thực hành tiếp theo.***

# **BÀI THỰC HÀNH 8. Cài đặt chữ ký số DSS**

## **8.1. Mục tiêu**

* Củng cố lại lý thuyết về chữ ký DSS.

## **8.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về sơ đồ chữ ký DSS.
* Xem trước nội dung bài thực hành ở nhà

## **8.3. Nội dung thực hành**

### **8.3.1. Bài tập thực hành**

Xây dựng chương trình ký số DSS thực hiện các chức năng sau:

**Menu chính.**

S. Ký số.

V. Kiểm thử chữ ký.

K. Kết thúc.

* ***Yêu cầu sinh viên gửi mã nguồn chương trình vào trước buổi thực hành tiếp theo.***

# **BÀI THỰC HÀNH 9. Cài đặt chữ ký không thể phủ định**

## **9.1. Mục tiêu**

* Củng cố lại lý thuyết về chữ ký không thể phủ định.

## **9.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về sơ đồ chữ ký không thể phủ định.
* Xem trước nội dung bài thực hành ở nhà

## **9.3. Nội dung thực hành**

### **9.3.1. Bài tập thực hành**

Xây dựng chương trình ký số không thể phủ định thực hiện các chức năng sau:

**Menu chính.**

S. Ký số.

V. Kiểm thử chữ ký.

K. Kết thúc.

* ***Yêu cầu sinh viên gửi mã nguồn chương trình vào trước buổi thực hành tiếp theo.***

**BÀI THỰC HÀNH 10. Hàm băm**

## **10.1. Mục tiêu**

* Củng cố lại lý thuyết về hàm băm.

## **10.2. Yêu cầu**

* Sinh viên đọc trươc nội dung lý thuyết về hàm băm.
* Xem trước nội dung bài thực hành ở nhà

**10.3. Nội dung thực hành**

**10.3.1. Bài tập thực hành**

1. Tìm hiểu về phương pháp sử dụng hàm hash MD5 và SHA trong thư viện mã hóa của .NET. Áp dụng viết chương trình mã hóa password lưu trữ và kiểm tra password.

2. Gần đây, người ta đã phát hiện điểm yếu của hàm hash MD5, tức tìm ra hai thông điệp có cùng giá trị hash MD5. Bạn hãy tìm những bít khác nhau của 2 thông điệp bên dưới và dùng thư viện của .NET để tính giá trị hash MD5 của chúng.

*Thông điệp 1 (dạng số thập lục phân):*  
d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c  
2f ca b5 87 12 46 7e ab 40 04 58 3e b8 fb 7f 89  
55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 71 41 5a  
08 51 25 e8 f7 cd c9 9f d9 1d bd f2 80 37 3c 5b  
96 0b 1d d1 dc 41 7b 9c e4 d8 97 f4 5a 65 55 d5  
35 73 9a c7 f0 eb fd 0c 30 29 f1 66 d1 09 b1 8f  
75 27 7f 79 30 d5 5c eb 22 e8 ad ba 79 cc 15 5c  
ed 74 cb dd 5f c5 d3 6d b1 9b 0a d8 35 cc a7 e3

*Thông điệp 2 (dạng số thập lục phân):*  
d1 31 dd 02 c5 e6 ee c4 69 3d 9a 06 98 af f9 5c  
2f ca b5 07 12 46 7e ab 40 04 58 3e b8 fb 7f 89  
55 ad 34 06 09 f4 b3 02 83 e4 88 83 25 f1 41 5a  
08 51 25 e8 f7 cd c9 9f d9 1d bd 72 80 37 3c 5b  
96 0b 1d d1 dc 41 7b 9c e4 d8 97 f4 5a 65 55 d5   
35 73 9a 47 f0 eb fd 0c 30 29 f1 66 d1 09 b1 8f  
75 27 7f 79 30 d5 5c eb 22 e8 ad ba 79 4c 15 5c  
ed 74 cb dd 5f c5 d3 6d b1 9b 0a 58 35 cc a7 e3

3. Viết chương trình tính giá trị MD5 cho một file trên máy tính tương tự như hình dưới đây.

