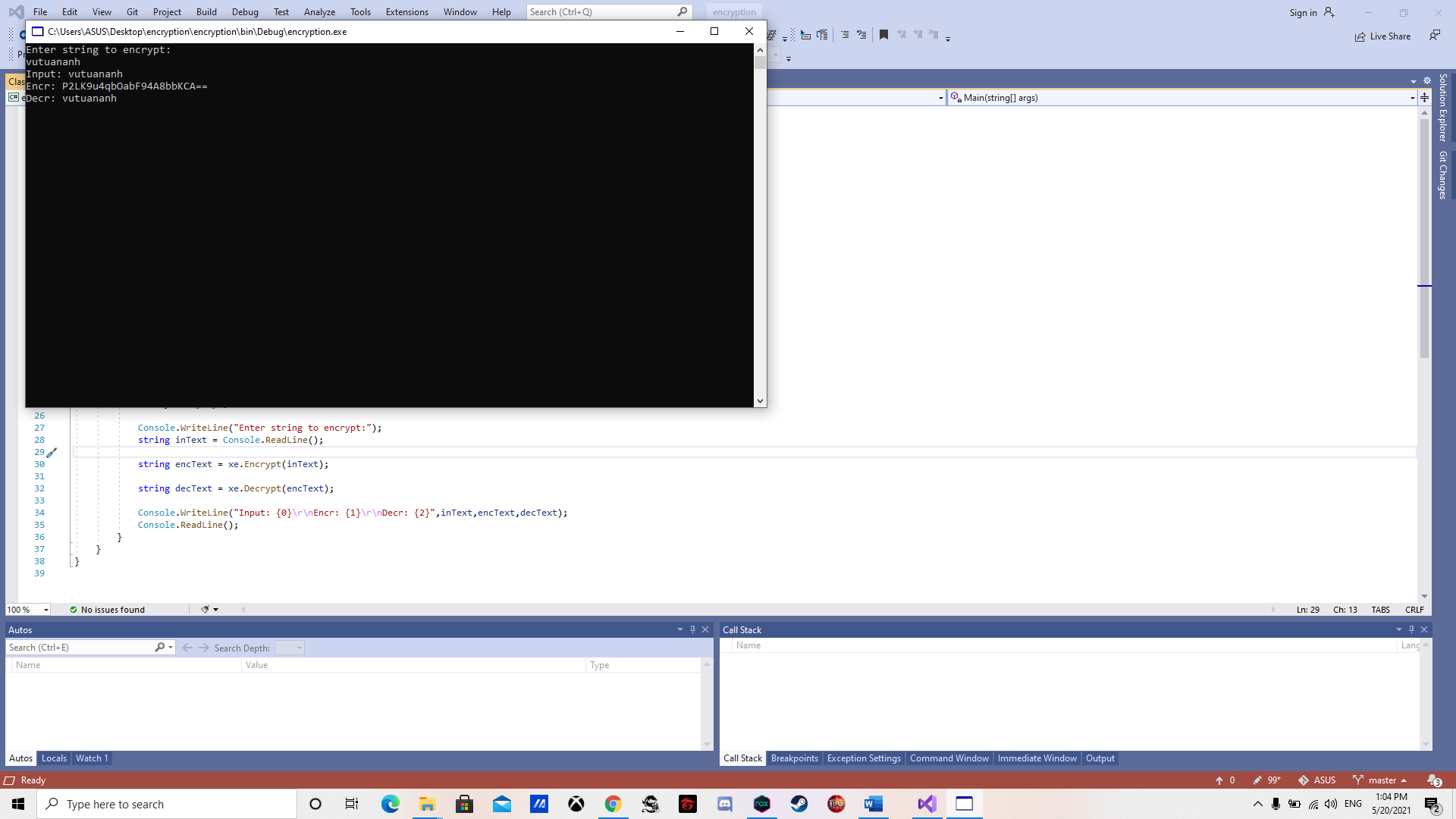
**Lab 3a: Private Key Encryption**

**Student Name: Vu Tuan Anh**

**Student ID: SE130255**

**1.**



**2. Implement a program for the MD5, SHA, SHA (256-bit), SHA (384-bit), SHA**

**(512-bit) and complete the following table (for the first few characters of the**

**signature):**

**apple**

**MD5:** HzhwvidPbEmz4xoMZyiVfw==

**SHA:** 0L4txCG+T80BcuWvzuo5cOLz2UA=

**SHA(256):** OnvT4jYKPSnupDb8+35ExzXRF8QtHBg1QgtrmULdTxs=

**SHA(384):** PYeG/LWIyTNIdWxkKXF9xsN0oU9wKTYigaOyHcECUN3w0FeAUnSYIusIvA3B5osP

**SHA(512):** hE2HeRA7lMGPSqTMDDtEdAWFgKmR+6hdPKaYoLyeUsWUD+t6ZaOikOF+ayPulD7MT3PnSQMnJFtP5dXvtZD+sg==

**Apple**

**MD5:** n2KQ9ENuWiNR8S4DtkM8PA==

**SHA:** n2KQ9ENuWiNR8S4DtkM8PA==

**SHA(256):** 8iP6qW8ikWKUkisXGiaW2Gj9H5EpMC60GkWyououu/0=

**SHA(384):** rnVnEyjjnkVyJVYmDPghZlR+iEH/PCB5wWL6T/xcgM1Z+4D6faQPoSAS9jpI07os

**SHA(512):** OTN0yklIul0w0kxFYRoXp/Y6yYkE3RfaSozasoVKYR2rokeGolLS1HakRstHalMyuvtBsUCVkWg7alznaAup4g==

**Apples**

**MD5:** 2uzPCtPB/IyAFSBcMy9bQg==

**SHA:** dsJDa1k/J6oHPwskBFMbjeBKauc=

**SHA(256):** 9ZA/UeNBp4Pmn/wtmzNQSHFvXwQKeConZM1OcosPdNk=

**SHA(384):** QH9IIJ4As9ssQ6wDUYVvuODPOsKEjpCl/aQs8sfRES+B15ZUtBsZp+UO3h5i/lGq

**SHA(512):** D++qVHQNC5rI+SXX9o5joC5D4odBkXhVPBa40dKrDKhotnFNg92M8oHRxNcVQVCie4d/YPkGPp51SskyH5OgKA==

**This is it.**

**MD5:** VERTSYh3ORXDSpdjl9l+2g==

**SHA:** SuLibn+V8522/NoAizwq2MwDI6U=

**SHA(256):** fIRfsSunVDjQkB/YV1Xb8MkzHDfmDGubltlv6kZaqDM=

**SHA(384):** dQHMeWHqxbtLPeoUkZlHvPBAFJui8C43BNPTkA4Aoajsglz3OYDh1Sxu2JJ9rbNJ

**SHA(512):** 7B1cGHRex3ReJWLLGVdJ8vHayW4UHDWW2cGOAv3H5sIJr+uESwIpUPnH8YMYxoyv5ocCoIdW7v/NoQUGgfezKQ==

**This is it**

**MD5:** rqBtbiZmnyC3aj73bL65jA==

**SHA:** ZrYtLU01BN0L01DQ29jvRWwj9Cg=

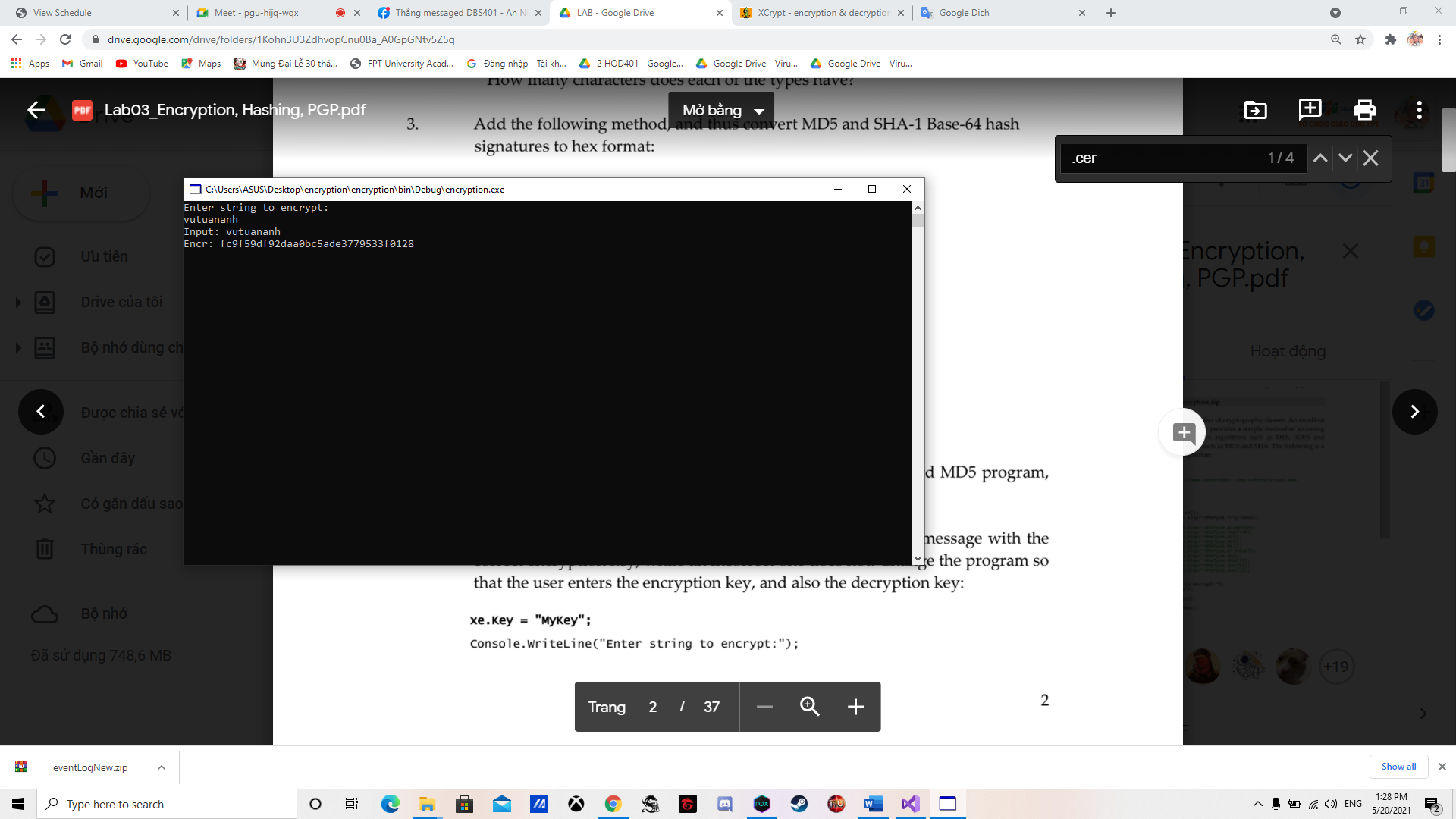
**SHA(256):** nCv62pZ8/C9GUcNNSF2bY+iUy4vdCFrWzf+b7S9H8Z8=

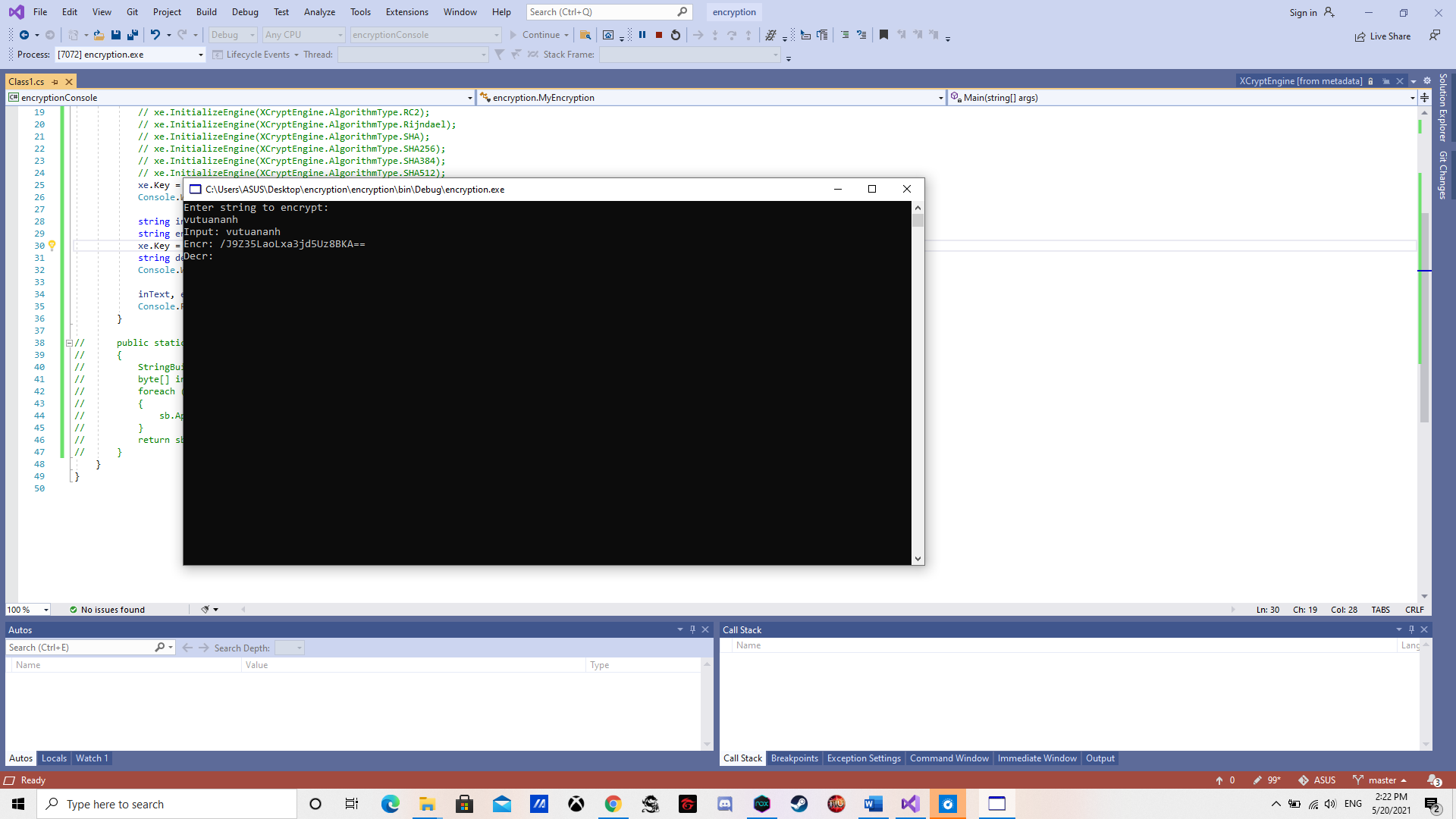
**SHA(384):** 1ArDENbXfJXmYocWqItiTIn9cUgHnD7lzVGh7+yGFGSFoBYPrsCSZ4xjvxmOBOrK

**SHA(512):** FUYHlWl6oTswUrH8liMJKuqYKGI/VHroKAHkhCw3492KpHYIzu9yqf2yEOlI8vNVIA1FpgAPtl1iNyOetn7+uA==

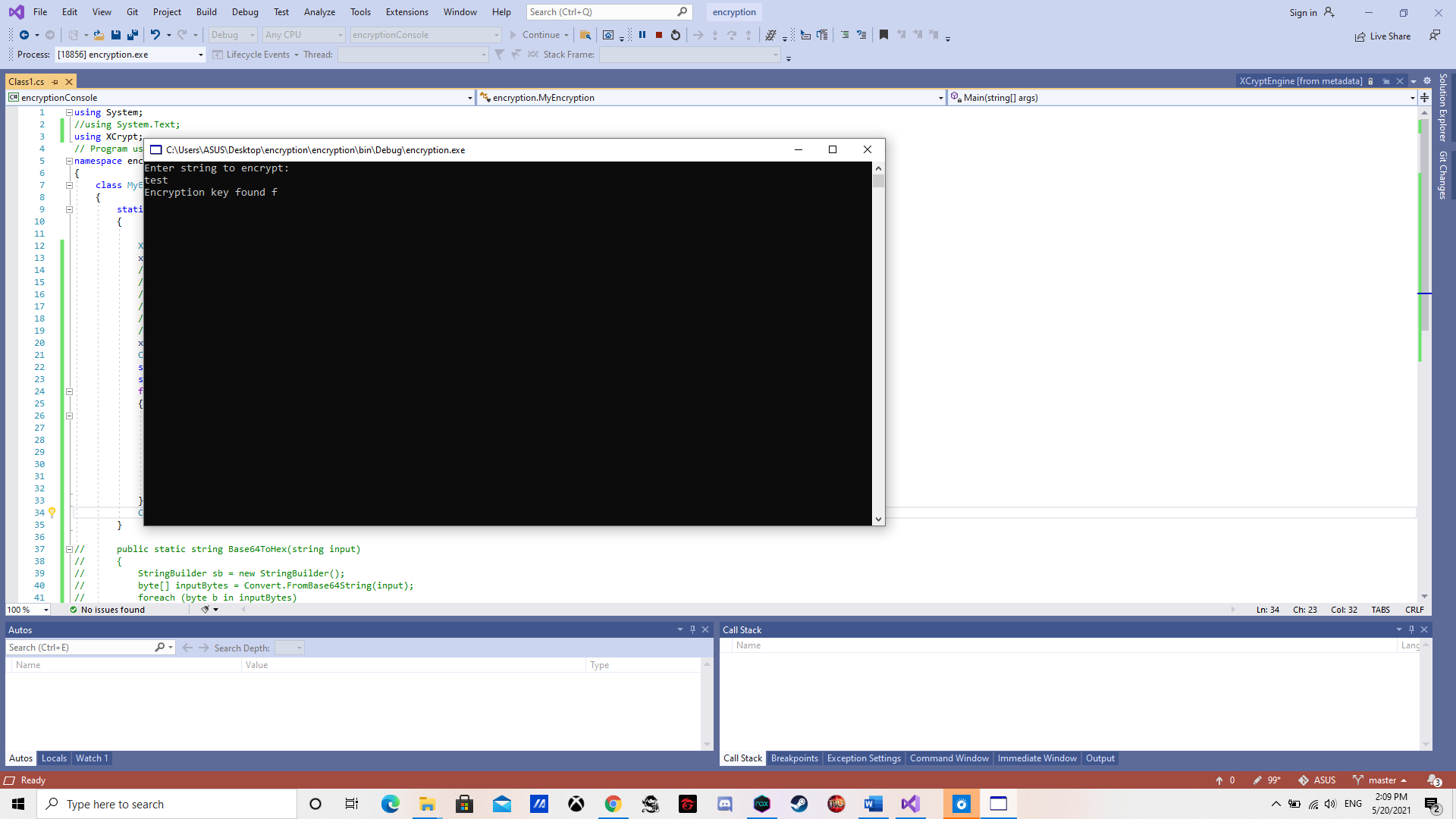
**3. Add the following method, and thus convert MD5 and SHA-1 Base-64 hash**

**signatures to hex format:**

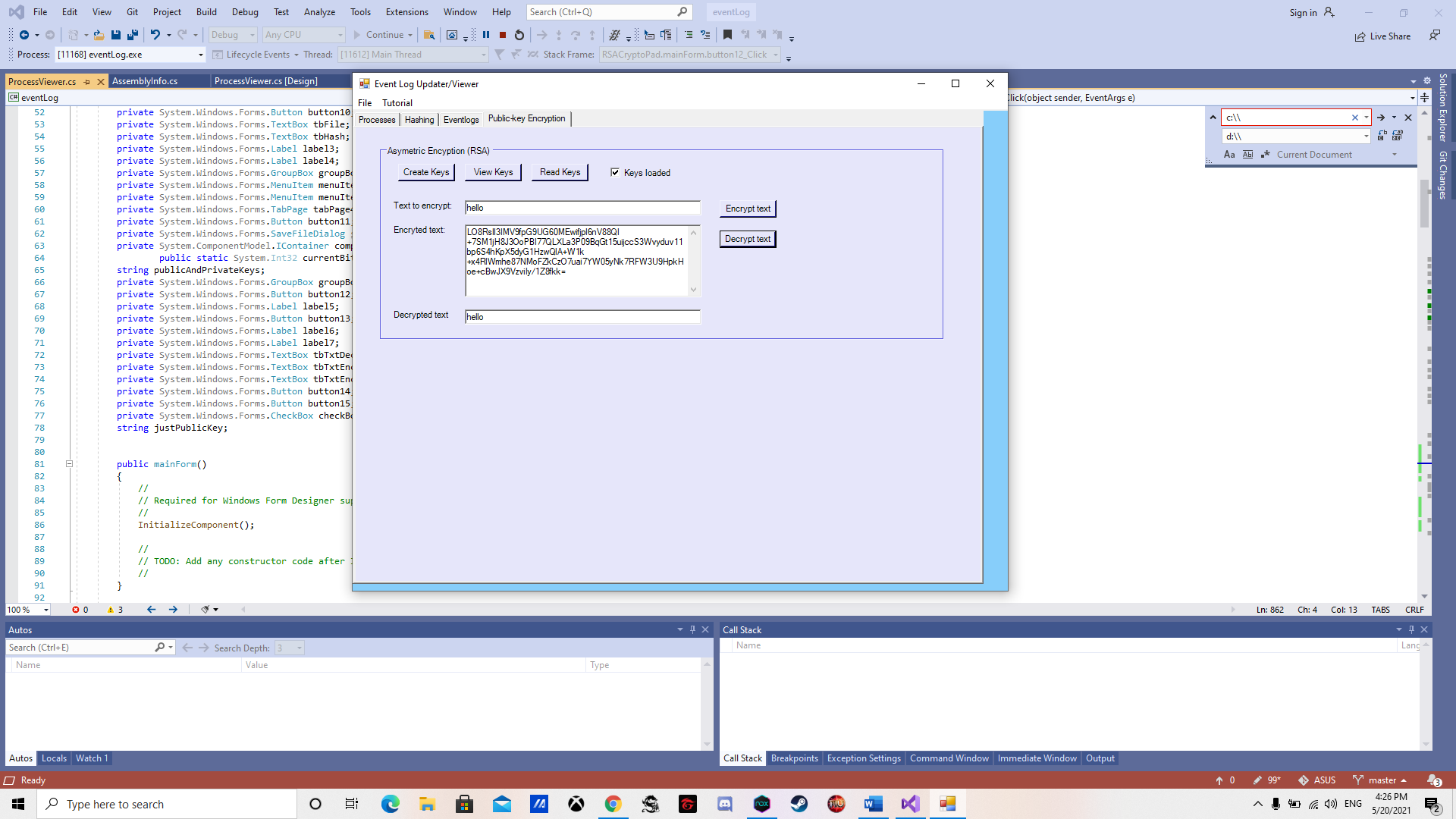


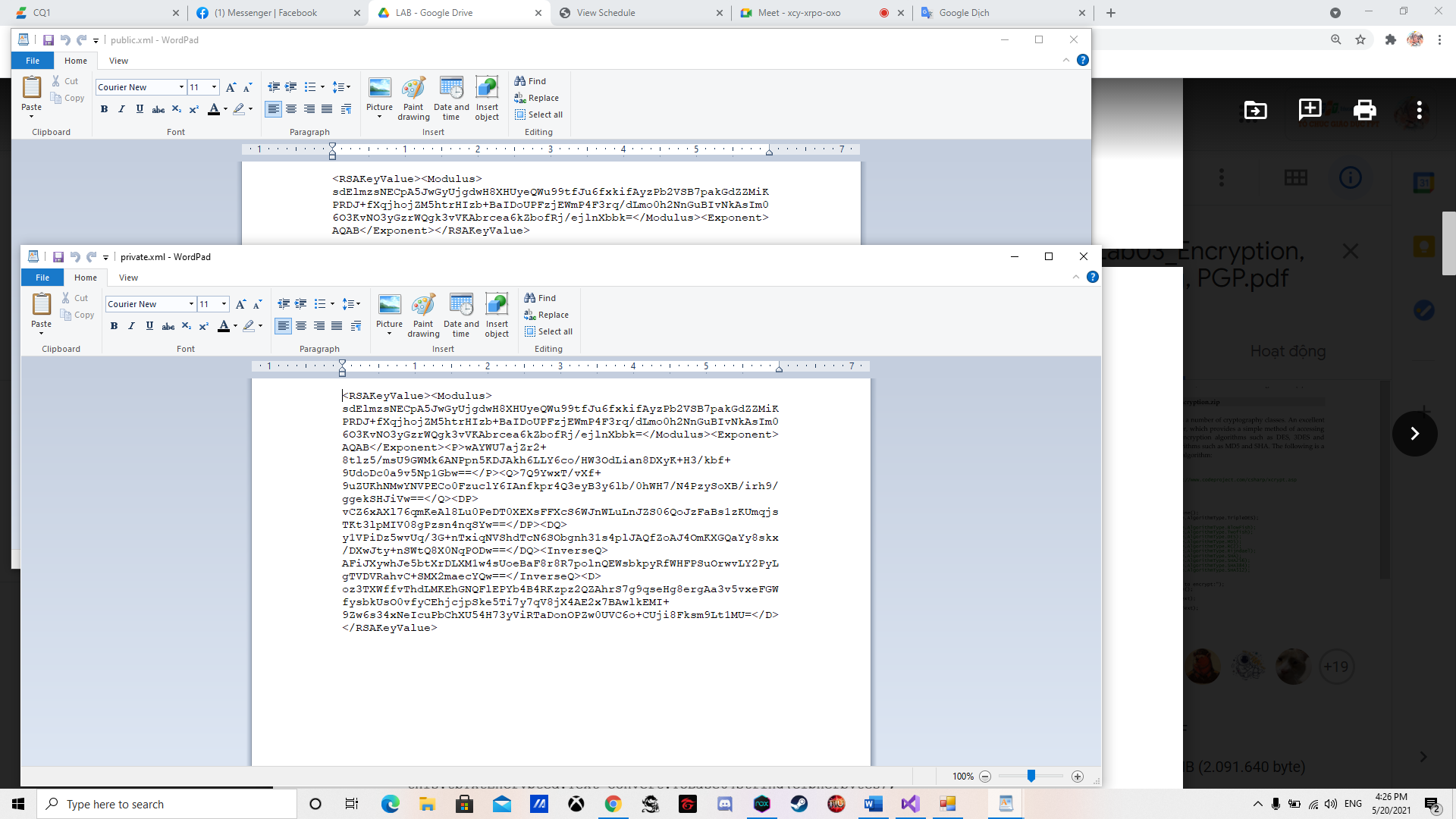
4. 

5.



**Lab 3b: Public-Key Encryption**

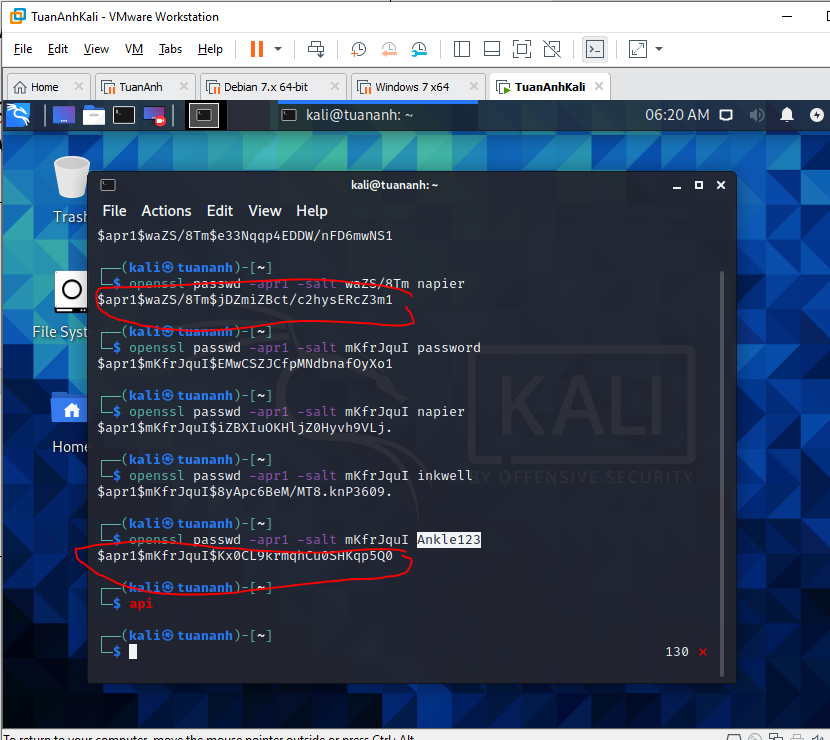


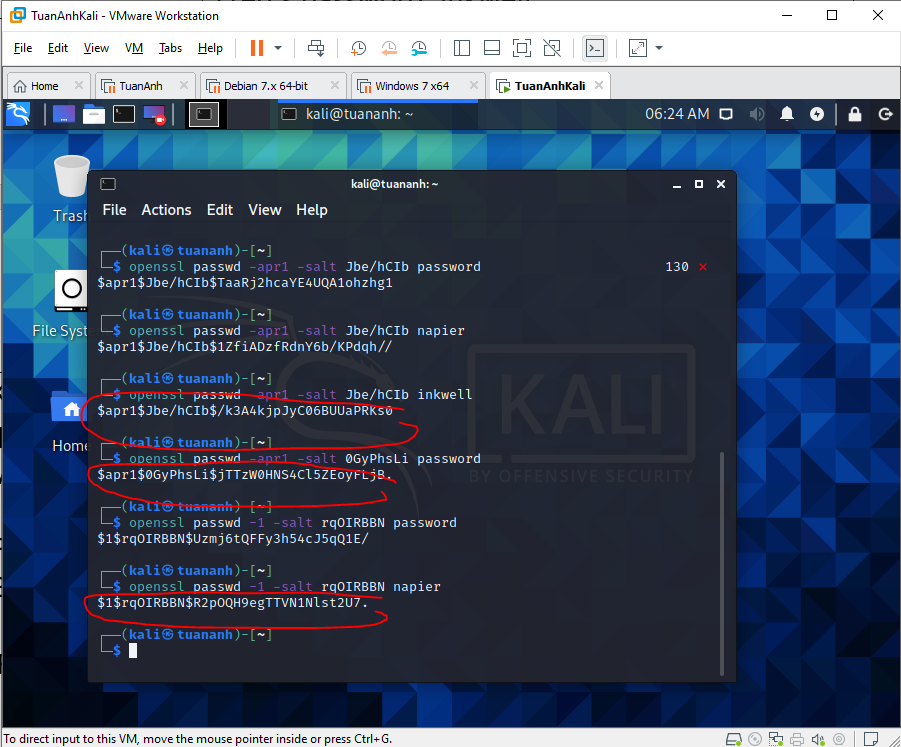


**Lab 3c: Hash Methods**

**1 Hashing**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Using (either on your Windows desktop or on Kali):  http://asecuritysite.com/encryption/md5  Match the hash signatures with their words (“Falkirk”,  “Edinburgh”, “Glasgow” and “Stirling”).  03CF54D8CE19777B12732B8C50B3B66F  D586293D554981ED611AB7B01316D2D5  48E935332AADEC763F2C82CDB4601A25  EE19033300A54DF2FA41DB9881B4B723 | 03CF5: Is it **Edinburgh**  D5862: Is it **Glasgow** 48E93: Is it **Falkirk**  EE190: Is it **Stirling** |
| 2 | Repeat Part 1, but now use openssl, such as:  echo -n 'Falkirk' | openssl md5 |  |
| 3 | Using (either on your Windows desktop or on Kali):  http://asecuritysite.com/encryption/md5  Determine the number of hex characters in the following hash signatures. | MD5 hex chars:  SHA-1 hex chars:  SHA-256 hex chars:  SHA-384 hex chars:  SHA-512 hex chars:  How does the number of hex characters relate to the length of the hash signature: |
| 4 | From your Windows desktop or Kali, for the following  /etc/shadow file, determine the matching password:  bill:$apr1$waZS/8Tm$jDZmiZBct/c2hysERcZ3m1  mike:$apr1$mKfrJquI$Kx0CL9krmqhCu0SHKqp5Q0  fred:$apr1$Jbe/hCIb$/k3A4kjpJyC06BUUaPRKs0  ian:$apr1$0GyPhsLi$jTTzW0HNS4Cl5ZEoyFLjB.  jane: $1$rqOIRBBN$R2pOQH9egTTVN1Nlst2U7. | The passwords are password, napier, inkwell and Ankle123.  [Hint: openssl passwd -apr1 -salt ZaZS/8TF napier]  Bill’s password: napier  Mike’s password: Ankle123  Fred’s password: inkwell  Ian’s password: password  Jane’s password: napier |
| 5 | From your Windows desktop or Kali, download the following:  <http://asecuritysite.com/files02.zip>  and the files should have the following MD5 signatures:  MD5(1.txt)= 5d41402abc4b2a76b9719d911017c592  MD5(2.txt)= 69faab6268350295550de7d587bc323d  MD5(3.txt)= fea0f1f6fede90bd0a925b4194deac11  MD5(4.txt)= d89b56f81cd7b82856231e662429bcf2 | Which file(s) have been modified0:  File 2.txt has been modified  MD5(2.txt) = e3fc91b12a36c2334ebb5b66caa2d75b |
| 6 | From your Windows desktop or Kali, download the following ZIP  file:  http://asecuritysite.com/letters.zip  View the Postscript files using:  <http://view.samurajdata.se/> | Outline what the letters contain:  Now determine the MD5 signature for them. What can you observe from the result?  MD5(letter\_of\_rec.ps) = a25f7f0b29ee0b3968c860738533a4b9  MD5(order.ps) = a25f7f0b29ee0b3968c860738533a4b9 |
| 7 | Select either Windows or Kali for this part:  On Kali, download the following ZIP file and run the two  programs, and run them in a command console:  http://asecuritysite.com/files01u.zip  Or on Windows, download the following ZIP file and run the two  programs, and run them in a command console:  <http://asecuritysite.com/files01.zip> | What do the programs do?  Now determine the MD5 signature for them. What can you  observe from the result? |





|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | On Kali, next create a words file (words) with the words of “napier”, “password” “Ankle123” and “inkwell”  Using hashcat crack the following MD5 signatures (hash1):  232DD5D7274E0D662F36C575A3BD634C  5F4DCC3B5AA765D61D8327DEB882CF99  6D5875265D1979BDAD1C8A8F383C5FF5  04013F78ACCFEC9B673005FC6F20698D  Command used: hashcat –m 0 hash1 words | 232DD...634C Is it napier  5F4DC...CF99 Is it password  6D587...5FF5 Is it Ankle123  04013...698D Is it nkwell |
| 2 | Using the method used in the first part of this tutorial, find crack  the following for names of fruits (the fruits are all in lowercase):  FE01D67A002DFA0F3AC084298142ECCD  1F3870BE274F6C49B3E31A0C6728957F  72B302BF297A228A75730123EFEF7C41  8893DC16B1B2534BAB7B03727145A2BB  889560D93572D538078CE1578567B91A | FE01D: orange  1F387: apple  72B30:banana  8893D: pear  88956: peach |

**3 Hashing Cracking (LM Hash/Windows)**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | On Kali, and using John the Ripper, and using a word list with the names of fruits, crack the following pwdump  passwords:  fred:500:E79E56A8E5C6F8FEAAD3B435B51404EE:  5EBE7DFA074DA8EE8AEF1FAA2BBDE876:::  bert:501:10EAF413723CBB15AAD3B435B51404EE:  CA8E025E9893E8CE3D2CBF847FC56814::: | Fred: APPLE  Bert: ORANGE |
| 2 | On Kali, and using John the Ripper, the following pwdump passwords (they are names of major Scottish cities/towns):  Admin:500:629E2BA1C0338CE0AAD3B435B51404EE:  9408CB400B20ABA3DFEC054D2B6EE5A1:::  fred:501:33E58ABB4D723E5EE72C57EF50F76A05:  4DFC4E7AA65D71FD4E06D061871C05F2:::  bert:502:BC2B6A869601E4D9AAD3B435B51404EE:  2D8947D98F0B09A88DC9FCD6E546A711::: | Admin: DUNDEE  Fred: ABERDEE  Bert: PERTH |
| 3 | On Kali, and using John the Ripper, crack the following pwdump passwords (they are the names of animals):  fred:500:5A8BB08EFF0D416AAAD3B435B51404EE:  85A2ED1CA59D0479B1E3406972AB1928:::  bert:501:C6E4266FEBEBD6A8AAD3B435B51404EE:  0B9957E8BED733E0350C703AC1CDA822:::  admin:502::333CB006680FAF0A417EAF50CFAC29C3:  D2EDBC29463C40E76297119421D2A707::: | Fred: SNAKE  Bert: TIGER  Admin: (empty) |

Lab 3d: Digital Certificates

1 Introduction

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | From:  http://asecuritysite.com/encryption/digitalcert  Open up certificate 1 and identify the following. | Serial number: 702958  Effective date: 4/24/2008 8:18:42 PM  Name: CN=Fred Smith, OU=None, E=fred@home, O=Nowhere, L=Edinburgh, S=Lothian, C=GB  Issuer: CN=Fred Smith, OU=None, E=fred@home, O=Nowhere, L=Edinburgh, S=Lothian, C=GB  What is CN used for: Fred Smith  What is ON used for: None  What is O used for: Nowhere  What is L used for: Edinburgh |
| 2 | Now open-up the ZIP file for the certificate, and view the CER file. | What other information can you gain from the certificate:  What is the size of the public key:  280  Which hashing method has been used:  hashcalc  Is the certificate trusted on your system: No] |
| 3 | For Example 2 to Example 6. Complete the following table: |  |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Cert | Organisation  (Issued to) | Date range when  valid | Size of public  key | Issuer | Root CA | Hash method | Is it trusted? |
| 1 | Fred Smith | 4/25/2008 – 4/25/2010 | R | CN = Fred Smith  OU = None  E = fred@home  O = Nowhere  L = Edinburgh  S = Lothian  C = GB | not in the Trusted Root CA store. | RSA | Not trusted |
| 2 | No One | 10/30/2011 – 10/29/2013 | 1024 bits | CN = No One  OU = None  E = noone@home  O = Nowhere Ltd  L = Flat Land  S = None  C = GB | not in the Trusted Root CA store. | RSA | Not trusted |
| 3 | www.google.co.uk | 1/3/2013 - 6/8/2013 | 1024 bits | CN = Google Internet Authority  O = Google Inc  C = US | none | RSA | This certificate has expired or is not yet valid. |
| 4 | www.cisco.com | 7/10/2012 – 7/12/2013 | 1024 bits | CN = VeriSign Class 3 Secure Server CA - G3  OU = Terms of use at https://www.verisign.com/rpa (c)10  OU = VeriSign Trust Network  O = VeriSign, Inc.  C = US | none | RSA | This certificate has expired or is not yet valid |
| 5 | login.live.com | 9/8/2012 – 9/20/2014 | 2048 bits | CN = VeriSign Class 3 Extended Validation SSL SGC CA  OU = Terms of use at https://www.verisign.com/rpa (c)06  OU = VeriSign Trust Network  O = VeriSign, Inc.  C = US | none | RSA | This certificate has expired or is not yet valid |
| 6 | login.oracle.com | 5/16/2012 – 7/16/2013 | 2048 bits | CN = VeriSign Class 3 International Server CA - G3  OU = Terms of use at https://www.verisign.com/rpa (c)10  OU = VeriSign Trust Network  O = VeriSign, Inc.  C = US | none | RSA | This certificate has expired or is not yet valid |

2 PFX files

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | We will now view some PFX certificate files, and which are protected  with a password:  http://asecuritysite.com/encryption/digitalcert2 | For Certificate 1, can you open it in the Web browser with an incorrect password:  Now enter “apples” as a password, and record some of the key details of the certificate:  Serial Number: 00FCA689  Now repeat for Certificate 2:  Serial Number: 00FCA689 |
| 2 | Now with the PFX files (contained in the ZIP files from the Web site),  try and import them onto your computer. Try to enter an incorrect  password first, and observe the message. | Was the import successful?  **yes**  If successful, outline some of the details of the certificates: |

3 Creating certificates

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Create your own certificate from:  http://asecuritysite.com/encryption/createcert  Add in your own details. | View the certificate, and verify some of the details on the  certificate.  Can you view the DER file? Yes |

4 Creating a self signed certificate

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | On Kali, login and get an IP address using:  sudo dhclient eth0 |  |
| 2 | Create your RSA key pair with:  openssl genrsa -out ca.key 2048  Next create a self-signed root CA certificate ca.crt for My Little Corp:  openssl req -new -x509 -days 1826 -key ca.key -out ca.crt |  |
| 3 | Next go to Places, and from your Home folder, open up ca.crt and view the details of the certificate. |  |
| 4 | Next we will create a subordinate CA (My Little Corp), and which will be used for the signing of the certificate. First, generate the  key:  openssl genrsa -out ia.key 2048  Next we will request a certificate for our newly created subordinate CA:  openssl req -new -key ia.key -out ia.csr  We can then create a certificate from the subordinate CA certificate and signed by the root CA.  openssl x509 -req -days 730 -in ia.csr -CA ca.crt -CAkey ca.key -set\_serial 01 -out ia.crt |  |
| 5 | If we want to use this certificate to digitally sign files and verify the signatures, we need to convert it to a PKCS12 file:  openssl pkcs12 -export -out ia.p12 -inkey ia.key -in ia.crt -chain -CAfile ca.crt |  |
| 6 | The crt format is in encoded in binary. If we want to export to a Base64 format, we can use DER:  openssl x509 -inform pem -outform pem -in  ca.crt -out ca.cer  and for My Little Corp:  openssl x509 -inform pem -outform pem -in ia.crt -out ia.cer |  |

**Lab 3e Hashing Implementation**

**A LM Hash**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Create a Python script to determine the LM  hash and NTLM hash of the following  words: | “Napier”  LM Hash: 12B9C54F6FE0EC80AAD3B435B51404EE  NT Hash: 3CA6CEF4B84985B6E3CD7B24843EA7D1  “Foxtrot”  LM Hash: 82121098B60F69F5AAD3B435B51404EE  NT Hash: 828F0524D3FFFD8632EE97253183FEF3 |

**B APR1**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Create a Python script to create the APR1  hash for the following:  Prove them against on-line APR1 generator  (or from the page given above). | changeme : $apr1$xR7wiK2w$d1kU3SYDjRNkJr.1tz1TH.  123456 : $apr1$t7snwQFh$zx7jgSKRWENpKoZ.AgYRe1  password : $apr1$ycJw8BCz$LQQ02Ctlgix6FDv2bSD/F/ |

**C SHA**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Create a Python script to create the SHA  hash for the following:  Prove them against on-line SHA generator  (or from the page given above). | **changeme :**  **SHA1:** fa9beb99e4029ad5a6615399e7bbae21356086b3  **SHA256:** 057ba03d6c44104863dc7361fe4578965d1887360f90a0895882e58a6248fc86  **SHA512:** f1891cea80fc05e433c943254c6bdabc159577a02a7395dfebbfbc4f7661d4af56f2d372131a45936de40160007368a56ef216a30cb202c66d3145fd24380906  **123456 :**  **SHA1:** 7c4a8d09ca3762af61e59520943dc26494f8941b  **SHA256:** 8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc6c92  **SHA512:** ba3253876aed6bc22d4a6ff53d8406c6ad864195ed144ab5c87621b6c233b548baeae6956df346ec8c17f5ea10f35ee3cbc514797ed7ddd3145464e2a0bab413  **password :**  **SHA1:** 5baa61e4c9b93f3f0682250b6cf8331b7ee68fd8  **SHA256:** 5e884898da28047151d0e56f8dc6292773603d0d6aabbdd62a11ef721d1542d8  **SHA512:** b109f3bbbc244eb82441917ed06d618b9008dd09b3befd1b5e07394c706a8bb980b1d7785e5976ec049b46df5f1326af5a2ea6d103fd07c95385ffab0cacbc86 |

**D PHPass**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Create a Python script to create the PHPass  hash for the following:  Prove them against on-line PHPass generator  (or from the page given above).  Just note the first five characters of the  hashed value. | **changeme:**  **Salt:** vutuananh  **Hash:** $P$5vutuanans3OZtLBUwvyPOFfhA7RpA.  **123456:**  **Salt:** vutuananh  **Hash:** $P$5vutuananOsGqK8gy4WfXxb5tobCeM1  **Password:**  **Salt:** vutuananh  **Hash:** $P$5vutuanan5ojaXpJUGKMpS68qn1vID0 |

**E PBKDF2**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Create a Python script to create the PBKDF2  hash for the following (uses a salt value of  “ZDzPE45C”). You just need to list the first six  hex characters of the hashed value. | **“changeme”:**  **SHA256:** 659cd4d35d10e6fe2d20716cc1f89be9  **“123456”:**  **SHA256:** 99420ff9d3ee31d6c04cfd570d747abe  **“password”:**  **SHA256**: eeb5055b7135504568326bb26500652a |

**F Bcrypt**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | Create the hash for the word “hello” for the  different methods (you only have to give the  first six hex characters for the hash):  Also note the number hex characters that the  hashed value uses: | **MD5:** 5d41402abc4b2a76b9719d911017c592  **SHA1:** aaf4c61ddcc5e8a2dabede0f3b482cd9aea9434d  **SHA256:** 2cf24dba5fb0a30e26e83b2ac5b9e29e1b161e5c1fa7425e73043362938b9824  **SHA512:** 9b71d224bd62f3785d96d46ad3ea3d73319bfbc2890caadae2dff72519673ca72323c3d99ba5c11d7c7acc6e14b8c5da0c4663475c2e5c3adef46f73bcdec043  **DES:** ZDVX7N5Bz.8wk  **MD5:** $1$ZDzPE45C$dOTT0LUnoqs6J7mNLdyse0  **Sun MD5:** $md5,rounds=34000$ZDzPE45C$$fdZ8uoSiWj6RcJOoMiaKX1  **SHA-1:** $sha1$480000$ZDzPE45C$LnzxSENDwEXBWKTQ1fc9/6BervKU  **SHA-256:** $5$rounds=535000$ZDzPE45C$TTN/Qd.elve1rHLazTSL0KCFxi7z5X9B/5l3xwFniaD  **SHA-512:** $6$rounds=656000$ZDzPE45C$6VWOiufRnOnxxetIEuLTZiM709Z3SBuNxhCf0Y0N4MUOgSTE85Nf9lI7FJJO4Autc2WEahI4URTibVYNy9V8w. |

**G A more complete set of hashes**

|  |  |  |
| --- | --- | --- |
| No | Description | Result |
| 1 | In the code, what does the modifier of  “[:22]” do?  In running the methods, which of them take  the longest time to compute?  Of the methods used, outline how you would  identify some of the methods. For APR1 has  an identifier of $apr1$. | * Modifier 22 is used for increased procedural services and demonstrates when a physician has gone above and beyond the typical framework of a particular procedure. * SHA512, Bcrypt, PBKDF2 (SHA1), PBKDF2 (SHA256) |

**H Reflective statements**

**1. Why might increasing the number of iterations be a better method of protecting a hashed password than using a salted version?**

**2. Why might the methods BCrypt, Phpass and PBFDK2 be preferred for storing**

**passwords than MD5, SHA?**

**Lab 3f: DLP - Disk/Email Encryption**

|  |  |  |
| --- | --- | --- |
| **No** | **Description** | **Result** |
| **1** | From the console on your Windows desktop (C:\Program Files  (x86)\GNU\GnuPG), create a key pair with (RSA and 2,048 bit keys):  **gpg --gen-key**  Now export your public key using the form of:  **gpg --export -a "Your name" > mypub.key**  Now export your private key using the form of:  **gpg --export-secret-key -a "Your name" > mypriv.key** | How is the randomness generated?    Outline the contents of your key file: |
| **2** | Now send your lab partner your public key in the contents of an email, and ask  them to import it onto their key ring:  **gpg --import theirpubickey.key**  Now list your keys with:  **gpg --list-keys** | Which keys are stored on your key ring and what  details do they have: |
| **3** | Create a text file, and save it. Next encrypt the file with their public key:  **gpg -e -a -u "Your Name" -r "Your Lab Partner Name"**  **hello.txt** |  |
| **4** | Send your encrypted file in an email to your lab partner, and get one back from  them.  Now create a file (such as myfile.asc) and decrypt the email using the public  key received from them with:  **gpg –d myfile.asc > myfile.txt** |  |
| **5** | Next using this public key file, send Bill (w.buchanan@napier.ac.uk) a question  (http://asecuritysite.com/public.txt):  -----BEGIN PGP PUBLIC KEY BLOCK-----  Version: BCPG C# v1.6.1.0  mQENBFTvF+gBCACkpcMPybSe1NTE1hDg86gPcQqoT8kD9oS/ankGwbB4R5zT+3Ny  MZWZwT43lL99R7sfkluglwVkqko74Lemy9pBF/rbWeWev6mCR3z1V3yTTv3zP1V5  tLcz3K65flRHPQU/FzxqH1T4kaH6dDiL/UuKKcyYMxXNnqERitJPU7ZJVhqeM3gi  4cG4znKY5fw8bdSpNC//pgkDzEaWYJFdyq/KqCwRK5r/Egj7FVHaLGC37lDgZKR5  dBoIVaOTfXykJLe3Vc3dIv9LU58U3YHqsc/w6X4E5R/eEnp0IwKyb7oXdrFOM5ud  DSoJ7aT24IqZW678vNtufGdr4OD+BF5r2UZpABEBAAG0GHcuYnVjaGFuYW5hQG5h  cGllci5hYy51a4kBHAQQAQIABgUCVO8X6AAKCRBOV4Uk9xMsXJgNB/4jfAnXLHjZ  +I4z3Hhqn9UMokU6Q4cQtrGX0OhelymKZTMXNoSKhT5fB9GBlIIbwMkZHxcUNmUB  PuAwq+RAhFqtRkcH3x1a5eNBhEvcfi9hS2ls43gfsrXjMzekY6dyzD/ePM7HvihJ  vrsQNZNI7ZIaP5viCZFgQqmwYQA1LCrEy/xpSXBNrqrOwuti+2+xeZsswitYLAzA  ryDMgCG9GPuSfkmvatYJJrl5QAhj1p0FKERhL1/h3bh18i8Llh1K9tEBxIJf4ZIy  ivV1bX5G36jciOrKCLi7/m6xhHh86brRQA++qwUdXU/3MMqvRwuinSO9NYeVCf6Y  V66cJqTgdR1F  =uiw7  -----END PGP PUBLIC KEY BLOCK----- |  |
| **6** | Next send your public key to Bill (w.buchanan@napier.ac.uk). |  |

**3 TrueCrypt Volumes**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **File** | **Size** | **Encryption type** | **Key size** | **Files/folders on disk** | **Hidden partition (y/n)** | **Hash method** |
| http://asecuritysite.com/tctest01.zip | 4.8 MB | AES | 256 bits | Hello.txt  Hello folder  Google.txt | No | HMAC-RIPEMD-160 |
| http://asecuritysite.com/tctest02.zip | 4.8 MB | Serpent | 256 bits | Left.txt  Right.txt | No | HMAC-SHA-512 |
| http://asecuritysite.com/tctest03.zip | 4.8 MB | AES-Twofish-Serpent | 768 bits | Hello123.txt  Test.txt | No | HMAC-Whirlpool |

**4 EFS**

|  |  |  |
| --- | --- | --- |
| **No** | **Description** | **Result** |
| **1** | Go to your Windows 7 instance. Now create a folder named:  **My\_Enc\_yourname**  Add some files to the folder, and then right click on the folder and encrypt it. | How does the name of the folder change when it is encrypted?  **My\_Enc\_TuanAnh.tar.gpg** |
| **2** | Now use:  Cipher /u | Which files are encrypted on your drive:  **None** |
| **3** | Using:  Cipher /c filename | Which encryption type and key size has been used for the file  encryption? |
| **4** | Make sure you can view your files.  Now export your certificates with:  Cipher /r:filename | Which are the names of the files created?   * My\_Enc\_TuanAnh.tar.gpg.CER * My\_Enc\_TuanAnh.tar.gpg.PFX   View the CER and PFX file. What is the difference between the two  files? **Same** |
| **5** | Go to Control Panel -> Internet Options  Then click on the Content tab and select the Certificates button.  Now view your EFS certificate. | Outline some of the details of the EFS certificate.  When does it expire? 5/21/2021 – 4/27/2121  What type of encryption does it use? SHA1RSA  What is the length of the encryption key? 2048 bits  Who has signed it? ASUS |
| **6** | Now delete the EFS certificate from the store and reboot your instance. | After reboot, can you access your files? **[Yes]** |
| **7** | Now import the PFX certificate that you created. | Can you access your files? **Yes** |

**6 Cracking digital certificates and file types**

|  |  |  |
| --- | --- | --- |
| **No** | **Description** | **Result** |
| **1** | Run Networksims.  Now run Toolkit client (Figure 2).  Goto the Encryption tab and select Digital Certificate from the left-hand menu. Next click on the Dictionary Search button, and load each of the following files (remember to extract to PFX):  <http://asecuritysite.com/log/fred.zip>  <http://asecuritysite.com/log/sample01.zip> | **What are the passwords for the PFX files?** |