ICSI MODULE 5:

-

The History of Encryption

5.1 The History of Encryption

Encrypting communications is a very old idea. People have found the need to send private communications for most of the history of civilization. The need for privacy originally started from military and political needs, but has expanded beyond that. Businesses need to keep data private to maintain a competitive edge. People want to keep certain information, such as their medical records and financial records, private.

For much of human history, private communications meant encrypting written communications. Over the past century, that has expanded to radio transmission, telephone communications, and computer/Internet communications. In the past several decades, the encryption of computerized transmissions has actually become ordinary. In fact you can find computer/Internet communications encrypted more often than phone or radio. The digital environment makes implementing a particular type of encryption much easier.

Whatever the nature of the data you are encrypting, or the mode of transmitting data, the basic concept is actually quite simple. Messages must be changed in such a way that they cannot be read easily by any party that intercepts them but can be decoded easily by the intended recipient. In this section, a few historical methods of encryption will be examined. Note that these are very old methods, and they cannot be used for secure communication today. An amateur could easily crack the methods discussed in this section. However, they are wonderful examples for conveying the concept of encryption without having to incorporate a great deal of math, which is required of the more complex encryption methods.

5.1.1 The Caesar Cipher

One of the oldest recorded encryption methods is the Caesar cipher. This name is based on a claim that ancient Roman emperors used this method. This method is simple to implement, requiring no technological assistance.

You choose some number by which to shift each letter of a text. For example, if the text is "**A cat"** and you choose to shift by two letters, then the message becomes **"C ecv"**. Or, if you choose to shift by three letters, it becomes **"D fdw"**.

In this example, you can choose any shifting pattern you want. Either you can shift to the right or to left by any number of spaces you like. Because this is a simple method to understand, it makes a good place to start your study of encryption. It is, however, extremely easy to crack. You see, any language has a certain letter and word frequency, meaning that some letters are used more frequently than others. In the English language, the most common single-letter word is "**a"**. The most common three-letter word is **"the"**.

Knowing these two characteristics alone could help you decrypt a Caesar cipher. For example, if you saw a string of seemingly nonsense letters and noticed that a three-letter word was frequently repeated in the message, you might easily guess that this word was "**the**"—and the odds are highly in favour of this being correct.

Furthermore, if you frequently noticed a single-letter word in the text, it is most likely the letter **"a"**. You now have found the substitution scheme for **a, t, h,** and **e**. You can now either translate all of those letters in the message and attempt to surmise the rest or simply analyse the substitute letters used for **a, t, h,** and **e** and derive the substitution cipher that was used for this message. Decrypting a message of this type does not even require a computer. Someone with no background in cryptography could do it in less than ten minutes using pen and paper.

Caesar ciphers belong to a class of encryption algorithms known as substitution ciphers. The name derives from the fact that each character in the unencrypted message is substituted by one character in the encrypted text.

The particular substitution scheme used (for example, 12 or 11) in a Caesar cipher is called a substitution alphabet (that is, b substitutes for a, u substitutes for t, etc.). Because one letter always substitutes for one other letter, the Caesar cipher is sometimes called a mono-alphabet substitution method, meaning that it uses a single substitution for the encryption.

The Caesar cipher, like all historical ciphers, is simply too weak for modern use. It is presented here just to help you understand the concepts of cryptography.

5.1.2 ROT 13

ROT 13 is another single alphabet substitution cipher. All characters are rotated 13 characters through the alphabet. For example the phrase **"A CAT"** will become **"N PNG"**.

5.1.3 Multi-Alphabet Substitution

Eventually, a slight improvement on the Caesar cipher was developed, called multi-alphabet substitution (also called polyalphabetic substitution). In this scheme, you select multiple numbers by which to shift letters (that is, multiple substitution alphabets). For example, if you select three substitution alphabets (12, 22, 13), then **"A CAT"** becomes **"C ADV".**

Notice that the fourth letter starts over with another 12, and you can see that the first A was transformed to C and the second A was transformed to D. This makes deciphering the underlying text more difficult. Although this is harder to decrypt than a Caesar cipher, it is not overly difficult to decode. It can be done with simple pen and paper and a bit of effort. It can be cracked quickly with a computer. In fact, no one would use such a method today to send any truly secure message, for this type of encryption is considered very weak.

Multi-alphabet ciphers are more secure than single-substitution ciphers. However, they are still not acceptable for modern

cryptographic usage. Computer-based cryptanalysis systems can crack historical cryptographic methods (both single alphabet and multi-alphabet) easily. The single-substitution and multi-substitution alphabet ciphers are discussed just to show you the history of cryptography, and to help you get an understanding of how cryptography works.

5.1.4 Rail Fence

All the preceding ciphers are substitution ciphers. Another approach to classic cryptography is the transposition cipher. The rail fence cipher may be the most widely known transposition cipher. You simply take the message you wish to encrypt and alter each letter on a different row. So **"attack at dawn"** is written as

A tc a dw

ta k ta n

Next, you write down the text reading from left to right as one normally would, thus producing

atcadwtaktan

In order to decrypt the message, the recipient must write it out on rows:

A tc a dw

ta k ta n

Then the recipient reconstructs the original message. Most texts use two rows as examples; however, this can be done with any number of rows you wish to use.

5.1.5 Vigenère

Vigenère is a polyalphabetic cipher and uses multiple substitutions in order to disrupt letter and word frequency. Let us consider a simple example. Remember a Caesar cipher has a shift, for example a shift of +2 (two to the right). A polyalphabetic substitution cipher would use multiple shifts. Perhaps a +2, –1, +1, +3. When you get to the fifth letter, you simply start over again. So, consider the word **"Attack"**, being encrypted

A (1) + 2 = 3 or C

T (20) –1 = 19 or S

T (20) +1 = 21 or U

A (1) +3 = 4 or D

C (3) +2 = 5 or E

K (11) –1 = 10 or J

Therefore, the ciphertext is **"CSUDEJ"**. Given that each letter has four possible substitutions, the letter and word frequency is significantly disrupted.

Perhaps the most widely known polyalphabetic cipher is the Vigenère cipher. This cipher was actually invented in 1553 by Giovan Battista Bellaso, though it is named after Blaise de Vigenère. It is a method of encrypting alphabetic text by using a series of different mono-alphabet ciphers selected, based on the letters of a keyword. Bellaso added the concept of using any keyword one might wish, thereby making the choice of substitution alphabets difficult to calculate.

5.1.6 Enigma

It is really impossible to have a discussion about cryptography and not talk about Enigma. Contrary to popular misconceptions, the Enigma is not a single machine but rather a family of machines. The first version was invented by German engineer Arthur Scherbius near the end of World War I. It was used by several different militaries, not just the Germans.

Some military texts encrypted using a version of Enigma were broken by Polish cryptanalysts Marian Rejewski, Jerzy Rozycki, and Henryk Zygalski. The three basically reverse engineered a working Enigma machine and used that information to develop tools for breaking Enigma ciphers, including one tool named the cryptologic bomb.

The core of the Enigma machine was the rotors, or disks, that were arranged in a circle with 26 letters on them. The rotors were lined up. Essentially, each rotor represented a different single substitution cipher. You can think of the Enigma as a sort of mechanical polyalphabetic cipher. The operator of the Enigma machine would be given a message in plaintext and then type that message into Enigma. For each letter that was typed in, Enigma would provide a different ciphertext based on a different substitution alphabet. The recipient would type in the ciphertext, getting out the plaintext, provided both Enigma machines had the same rotor settings.

There were actually several variations of the Enigma machine. The Naval Enigma machine was eventually cracked by British cryptographers working at the now famous Bletchley Park. Alan Turing and a team of analysts were able to eventually break the Naval Enigma machine. Many historians claim this shortened World War II by as much as two years.

Modern Encryption Methods

5.2 Modern Encryption Methods

Modern methods of encryption are more secure than the historical methods discussed in the previous section. All the methods discussed in this section are in use today and are considered reasonably secure. In some cases, the algorithm behind these methods requires a sophisticated understanding of mathematics.

Number theory often forms the basis for encryption algorithms. Fortunately, for our purposes, having the exact details of these encryption algorithms is not important; this means that you do not require a strong mathematics background to follow this material. More important is a general understanding of how a particular encryption method works and how secure it is.

5.2.1 Symmetric Encryption

Symmetric encryption refers to the methods where the same key is used to encrypt and decrypt the plaintext.

5.2.1.1 Binary Operations

Part of modern symmetric cryptography ciphers involves using binary operations. Various operations on binary numbers (numbers made of only zeroes and ones) are well known to programmers and programming students. However, for those readers not familiar with them, a brief explanation follows. When working with binary numbers, three operations are not found in normal math: AND, OR, and XOR operations. Each is illustrated next.

AND

To perform the AND operation, you take two binary numbers and compare them one place at a time. If both numbers have a **"one"** in both places, then the resultant number is a **"one"**. If not, then the resultant number is a **"zero"**, as you see below:

1 1 0 1
1 0 0 1
---
1 0 0 1

OR

The OR operation checks to see whether there is a **"one"** in either or both numbers in a given place. If so, then the resultant number is **"one"**. If not, the resultant number is **"zero"**, as you see here:

1 1 0 1
1 0 0 1
---
1 1 0 1

XOR

The XOR operation impacts your study of encryption the most. It checks to see whether there is a **"one"** in a number in a given place, but not in both numbers at that place. If it is in one number but not the other, then the resultant number is **"one"**. If not, the resultant number is **"zero"**, as you see here:

1 1 0 1
1 0 0 1
---
0 1 0 0

XORing has an interesting property in that it is reversible. If you XOR the resultant number with the second number, you get back the first number. In addition, if you XOR the resultant number with the first number, you get the second number.

0 1 0 0
1 0 0 1
---
1 1 0 1

Binary encryption using the XOR operation opens the door for some rather simple encryption. Take any message and convert it to binary numbers and then XOR that with some key. Converting a message to a binary number is a simple two-step process. First, convert a message to its ASCII code, and then convert those codes to binary numbers.

Each letter/number will generate an eight-bit binary number. You can then use a random string of binary numbers of any given length as the key. Simply XOR your message with the key to get the encrypted text, and then XOR it with the key again to retrieve the original message.

This method is easy to use and great for computer science students; however, it does not work well for truly secure communications because the underlying letter and word frequency remains. This exposes valuable clues that even an amateur cryptographer can use to decrypt the message. Yet, it does provide a valuable introduction to the concept of single-key encryption.

Although simply XORing the text is not the method typically employed, single-key encryption methods are widely used today. For example, you could simply include a multi-alphabet substitution that was then XORed with some random bit stream—variations of which do exist in a few actual encryption methods currently used.

Modern cryptography methods, as well as computers, make decryption a rather advanced science. Therefore, encryption must be equally sophisticated in order to have a chance of success.

5.2.1.2 Data Encryption Standard

Data Encryption Standard, or DES as it is often called, was developed by IBM in the early 1970s and made public in 1976. DES uses a symmetric key system, which means the same key is used to encrypt and to decrypt the message. DES uses short keys and relies on complex procedures to protect its information. The actual DES algorithm is quite complex. The basic concept, however, is as follows:

1. The data is divided into 64-bit blocks, and those blocks are then reordered.
2. Reordered data are then manipulated by 16 separate rounds of encryption, involving substitutions, bit-shifting, and logical operations using a 56-bit key.
3. Finally, the data are reordered one last time.

DES uses a 56-bit cipher key applied to a 64-bit block. There is actually a 64-bit key, but one bit of every byte is actually used for error detection, leaving just 56 bits for actual key operations. The problem with DES is the same problem that all symmetric key algorithms have: How do you transmit the key without it becoming compromised? This issue led to the development of public key encryption.

5.2.1.3 Blowfish

Blowfish is a symmetric block cipher. This means that it uses a single key to both encrypt and decrypt the message and works on "blocks" of the message at a time. It uses a variable-length key ranging from 32 to 448 bits. This flexibility in key size allows you to

use it in various situations. Blowfish was designed in 1993 by Bruce Schneier. It has been analysed extensively by the cryptography community and has gained wide acceptance. It is also a non-commercial (that is, free of charge) product, thus making it attractive to budget-conscious organisations.

5.2.1.4 AES (Advanced Encryption Standard)

Advanced Encryption Standard (AES) uses the Rijndael algorithm. The developers of this algorithm have suggested multiple alternative pronunciations for the name, including "reign dahl," "rain doll," and "rhine dahl." This algorithm was developed by two Belgian researchers, Joan Daemen of Proton World International and Vincent Rijmen, a postdoctoral researcher in the Electrical Engineering Department of Katholieke Universiteit Leuven.

AES specifies three key sizes: 128, 192, and 256 bits. By comparison, DES keys are 56 bits long, and Blowfish allows varying lengths up to 448 bits. AES uses a block cipher. This algorithm is widely used, considered very secure, and therefore a good choice for many encryption scenarios.

5.2.2 Public Key Encryption

Public key encryption is essentially the opposite of single-key encryption. With any public key encryption algorithm, one key is used to encrypt a message (called the public key) and another is used to decrypt the message (the private key). You can freely distribute your public key so that anyone can encrypt a message to send to you, but only you have the private key and only you can decrypt the message. The actual mathematics behind the creation and applications of the keys is a bit complex and beyond the scope of this book. Many public key algorithms are dependent, to some extent, on large prime numbers, factoring, and number theory.

5.2.2.1 RSA

The RSA method is a widely used encryption algorithm. You cannot discuss cryptography without at least some discussion of RSA. This public key method was developed in 1977 by three mathematicians: Ron Rivest, Adi Shamir, and Len Adleman. The name RSA is derived from the first letter of each mathematician's last name.

One significant advantage of RSA is that it is a public key encryption method. That means there are no concerns with distributing the keys for the encryption. However, RSA is much slower than symmetric ciphers. In fact, in general, asymmetric ciphers are slower than symmetric ciphers.

The steps to create the key are as follow:

1. Generate two large random primes, **p** and **q**, of approximately equal size.

2. Pick two numbers so that when they are multiplied together the product will be the size you want (that is, 2048 bits, 4096 bits, etc.).

3. Now multiply **p** and **q** to get **n**.

4. Let **n = pq**.

5. Multiply Euler's totient for each of these primes. If you are not familiar with this concept, the Euler's Totient is the total number of co-prime numbers. Two numbers are considered co-prime if they have no common factors.

For example, if the original number is 7, then 5 and 7 would be co-prime. It just so happens that for prime numbers, this is always the number minus 1. For example, 7 has 6 numbers that are co-prime to it (if you think about this a bit you will see that 1, 2, 3, 4, 5, 6 are all co-prime with 7).

6. Let **m = (p – 1)(q – 1).**

7. Select another number; call this number **e**. You want to pick **e** so that it is co-prime to **m**.

8. Find a number **d** that when multiplied by **e** and modulo **m** would yield 1. (Note: Modulo means to divide two numbers and return the remainder. For example, 8 modulo 3 would be 2.)

9. Find **d**, such that **de mod m ≡ 1**.

Now you publish **e** and **n** as the public key and keep **d** and **n** as the secret key. To encrypt you simply take your message raised to the **e** power and modulo **n = Me % n**

To decrypt you take the ciphertext, and raise it to the **d** power modulo **n**:

P = Cd % n

RSA has become a popular encryption method. It is considered quite secure and is often used in situations where a high level of security is needed.

5.2.2.2 Elliptic Curve

The Elliptic Curve algorithm was first described in 1985 by Victor Miller (IBM) and Neil Koblitz (University of Washington).

The security of Elliptic Curve cryptography is based on the fact that finding the discrete logarithm of a random elliptic curve element with respect to a publicly known base point is difficult to the point of being impractical to do.

The size of the elliptic curve determines the difficulty of finding the algorithm, and thus the security of the implementation. The level of security afforded by an RSA-based system with a large modulus can be achieved with a much smaller elliptic curve group. There are actually several ECC algorithms. There is an ECC version of Diffie-Hellman, an ECC version of DSA, and many others.

The U.S. National Security Agency has endorsed ECC (Elliptic Curve Cryptography) by including schemes based on it in its Suite B set of recommended algorithms and allows their use for protecting information classified up to top secret with 384-bit keys.

5.2.3 Digital Signatures and Certificates

A digital signature is not used to ensure the confidentiality of a message, but rather to guarantee who sent the message. This is referred to as non-repudiation. Essentially, a digital signature proves who the sender is. Digital signatures are actually rather simple, but clever. They simply reverse the asymmetric encryption process.

Recall that in asymmetric encryption, the public key (which anyone can have access to) is used to encrypt a message to the recipient, and the private key (which is kept secure, and private) can decrypt it. With a digital signature, the sender encrypts something with his or her private key. If the recipient is able to decrypt that with the sender's public key, then it must have been sent by the person purported to have sent the message.

5.2.3.1 Digital Certificates

Public keys are widely distributed and that getting someone's public key is fairly easy to do and are also needed to verify a digital signature. As to how public keys are distributed, probably the most common way is through digital certificates. The digital certificate

contains a public key and some means to verify whose public key it is.
X.509 is an international standard for the format and information contained in a digital certificate. X.509 is the most used type of digital certificate in the world. It is a digital document that contains a public key signed by a trusted third party, which is known as a certificate authority (CA). The contents of an X.509 certificate are:

- Version
- Certificate holder's public key
- Serial number
- Certificate holder's distinguished name
- Certificate's validity period
- Unique name of certificate issuer
- Digital signature of issuer
- Signature algorithm identifier

A certificate authority issues digital certificates. The primary role of the CA is to digitally sign and publish the public key bound to a given user. It is an entity trusted by one or more users to manage certificates.
A registration authority (RA) is used to take the burden off, of a CA by handling verification prior to certificates being issued. RAs act as a proxy between users and CAs. RAs receive a request, authenticate it, and forward it to the CA.
A public key infrastructure (PKI) distributes digital certificates. This network of trusted CA servers serves as the infrastructure for distributing digital certificates that contain public keys. A PKI is an arrangement that binds public keys with respective user identities by means of a CA.
What if a certificate is expired, or revoked? A certificate revocation list (CRL) is a list of certificates that have been revoked for one reason or another. Certificate authorities publish their own certificate revocation lists. A newer method for verifying certificates is Online Certificate Status Protocol (OCSP), a real-time protocol for verifying certificates.
There are several different types of X.509 certificates. They each have at least the elements listed at the beginning of this section, but are for different purposes. The most common certificate types are listed below:

- Domain validation certificates are among the most common. These are used to secure communication with a specific domain. This is a low-cost certificate that website administrators use to provide TLS for a given domain.
- Wildcard certificates, as the name suggests, can be used more widely, usually with multiple sub-domains of a given domain. So rather than have a different X.509 certificate for each sub-domain, you would use a wildcard certificate for all sub-domains.
- Code-signing certificates are X.509 certificates used to digitally sign some type of computer code. These usually require more validation of the person requesting the certificate, before they can be issued.
- Machine/computer certificates are X.509 certificates assigned to a specific machine. These are often used in authentication protocols. For example, in order for the machine to sign into the network, it must authenticate using its machine certificate.
- User certificates are used for individual users. Like machine/computer certificates, these are often used for authentication. The user must present his or her certificate to authenticate prior to accessing some resource.
- E-mail certificates are used for securing e-mail. Secure Multipurpose Internet Mail Extensions (S/MIME) uses X.509 certificates to secure e-mail communications.
- A Subject Alternative Name (SAN) is not so much a type of certificate as a special field in X.509. It allows you to specify additional items to be protected by this single certificate. These could be additional domains or IP addresses.
- Root certificates are used for root authorities. These are usually self-signed by that authority.

5.2.3.2 PGP Certificates

Pretty Good Privacy (PGP) is not a specific encryption algorithm, but rather a system. It offers digital signatures, asymmetric encryption, and symmetric encryption. It is often found in e-mail clients. PGP was introduced in the early 1990s, and it's considered to be a very good system.
PGP uses its own certificate format. The main difference, however, is that PGP certificates are self-generated. They are not generated by any certificate authority.

Windows and Linux Encryption

5.3 Windows and Linux Encryption

Microsoft Windows provides encryption tools to prevent loss of confidential data.

- Encrypting File System (EFS) encodes files in order anyone who is able to get the files not to be able to read them. The files are only readable when you sign in to the computer using your user account. You can use EFS to encrypt individual files and entire drives. It is recommended to encrypt folders or drives instead of individua files. When you encrypt a folder or a drive the files contained are also encrypyed. Even new files created in the encrypted folder are automatically encrypted.

- BitLocker Drive Encryption provides another layer of protection by encrypting the entire hard drive. By linking this encryption to a key stored in a Trusted Platform Module (TPM), bitLocker reduces the risk of data being lost when a computer is stolen or when a hard disk is stolen and placed in another computer. In such scenario the thief will boot into an alternate operating system and try to retrieve data from the stolen drive or computer. With BitLocker that type of offline attacke in neutered.
- BitLocker To Go extends BitLocker encryption to removable media such as USB flash drives.

Linux provides a number of cryptographic techniques to protect data on physical devices such as hard disks or removable media. Such technique is Linux Unified Key Setup (LUKS). This technique allows the encryption of Linux partitions.

Using LUKS you can encrypt the entire blcok device which is well suited to protect data on removable storage or the laptops disk drive. LUKS uses the existing device mapper kernel subsystem and also provides passprhase strengthening for protection against dictionary attacks.

Guided Exercise: Enabling BitLocker

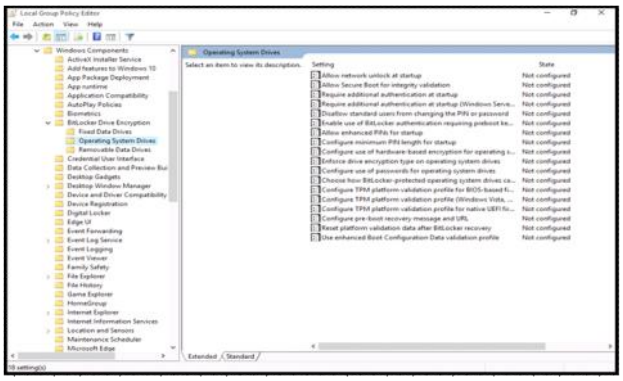
5.4 Guided Exercise: Enabling BitLocker

Resources	
Files	None
Machines	Windows 10

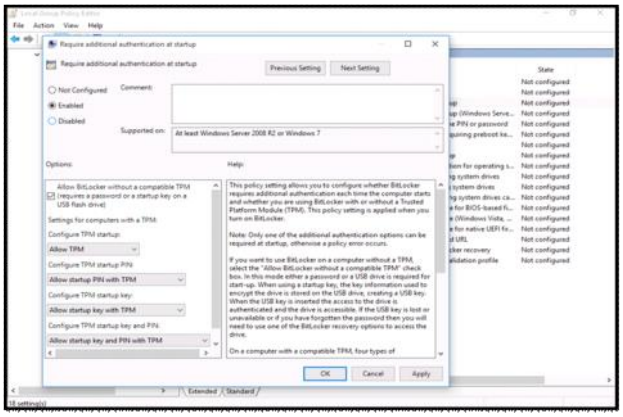
In this exercise you will enable Bitlocker to encrypt the hard drive.

Login to Windows 10 and once logged in click on the Start button and in the search box write gpedit.msc and press enter.

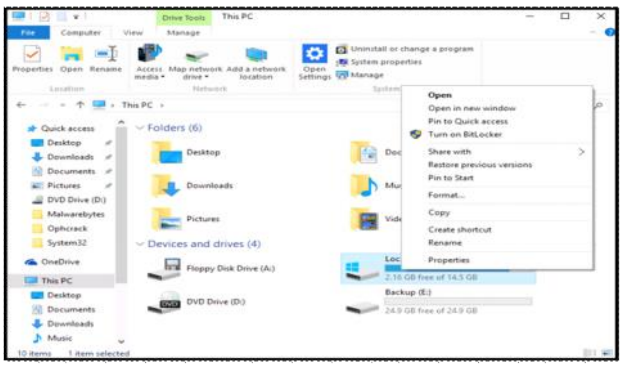
On the Local Group Policy Editor window expand from Computer Configuration -> Administrative Templates -> Windows Components -> BitLocker Drive Encryption -> Operating System Drives.



Double click the option Require additional authentcation at startup and on the new window that will open click Enabled (Make sure the box Allow BitLocker without a compatible TPM is ticket)and then Apply -> OK. After that close the Local Group Policy Editor window.

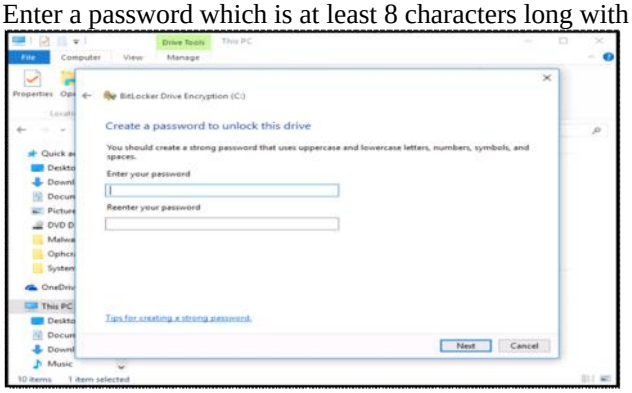


Open file explorer and click on This PC. Right click on the Local Disk (C:) and click on the option Turn on BitLocker.

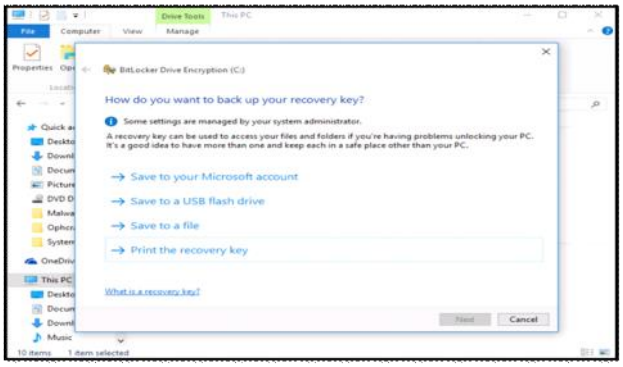


BitLocker will verify that the computer meets the requirements and will ask how to unlock the drive at startup. Choose the option Enter a password.

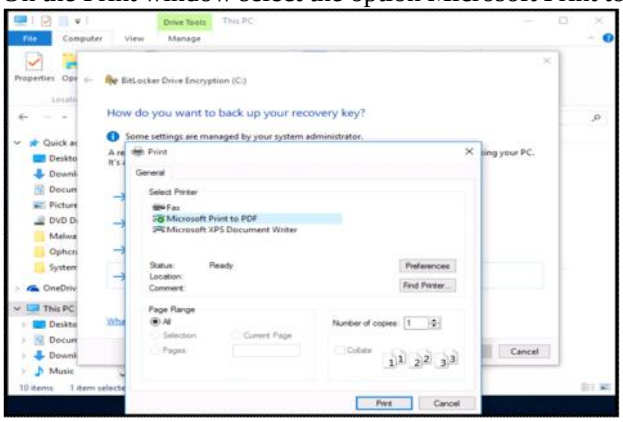
Enter a password which is at least 8 characters long with uppercase and lowercase letters, symbols and spaces.



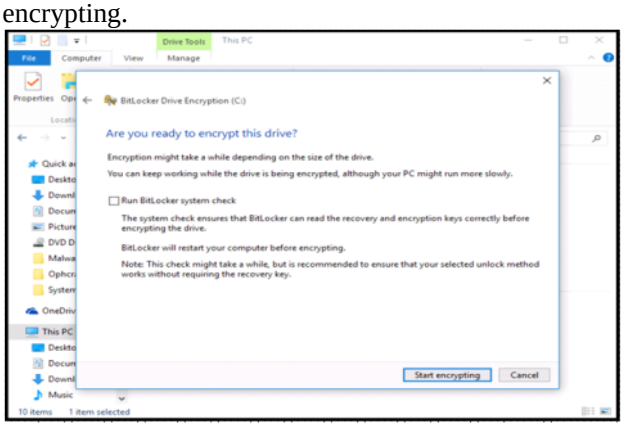
Then you should save (backup) the recovery key. You can save the recovery key to a USB flash drive, print it, save it to a file or to your Microsoft account. For this exercise choose the option to print the recovery key.



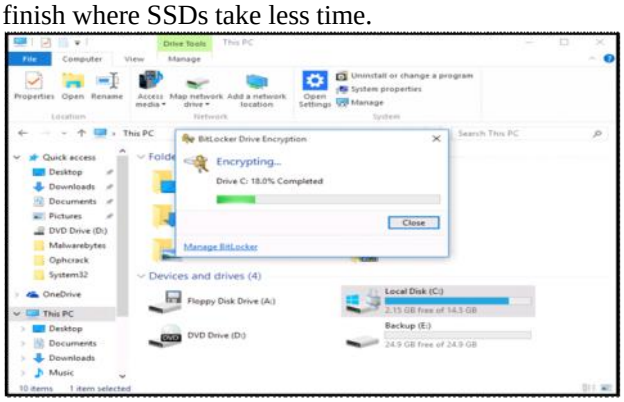
On the Print window select the option Microsoft Print to PDF and click Print.



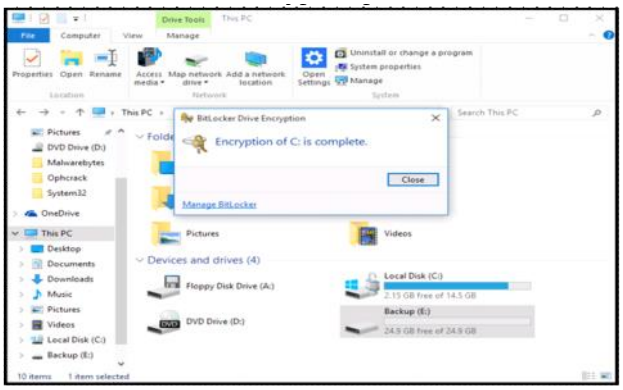
Provide a file name for the recovery key (BitLocker Key for example) and save it to the Backup (E:) drive. Click Save and then OK. Then it will ask you to Run BitLocker system check. Uncheck the box (although you can leave it to run a check) and click Start encrypting.



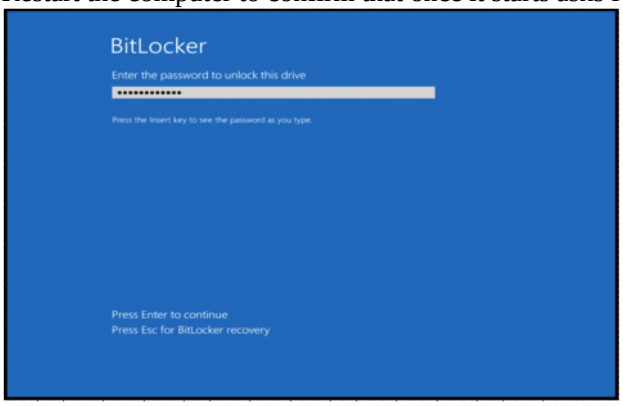
The encryption process will take some time to finish although if you have a large SATA disk such as 1TB will take a long time to finish where SSDs take less time.



Click Close once the encryption process has finished.



Restart the computer to confirm that once it starts asks for the BitLocker password. Enter the password and then press enter.



Then the computer will ask for the password to login.

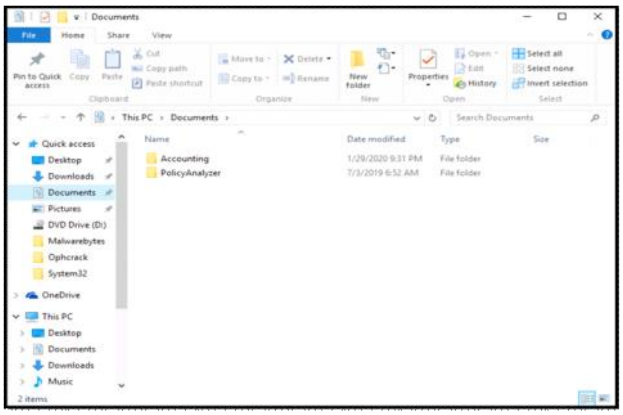
Guided Exercise: Encrypting a Folder Using EFS

5.5 Guided Exercise: Encrypting a Folder Using EFS

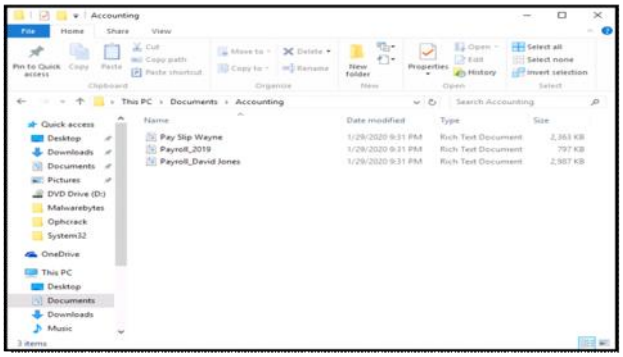
Resources	
Files	None
Machines	Windows 10

In this exercise you will encrypt a folder along with its contents.

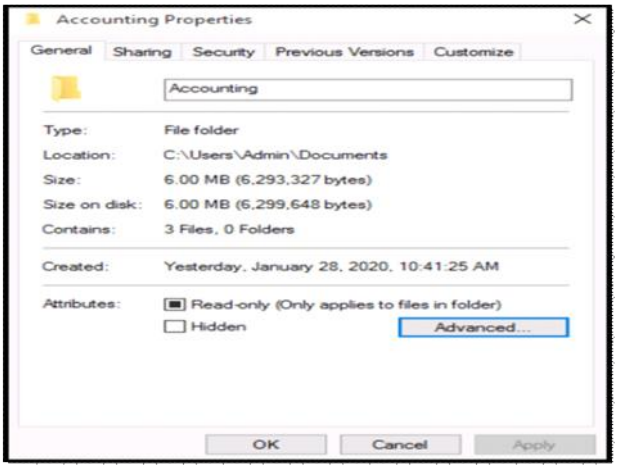
Login to Windows 10 and open File Explorer. Once File Explorer opens click on the Documents folder from the Quick Access menu on the left.



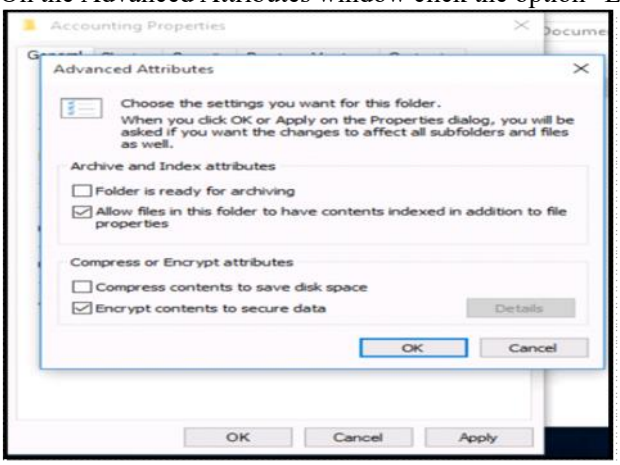
Within the Documents folder a subfolder exist with the name Accounting. Within the Accounting folder there are 3 files which are considered sensitive.



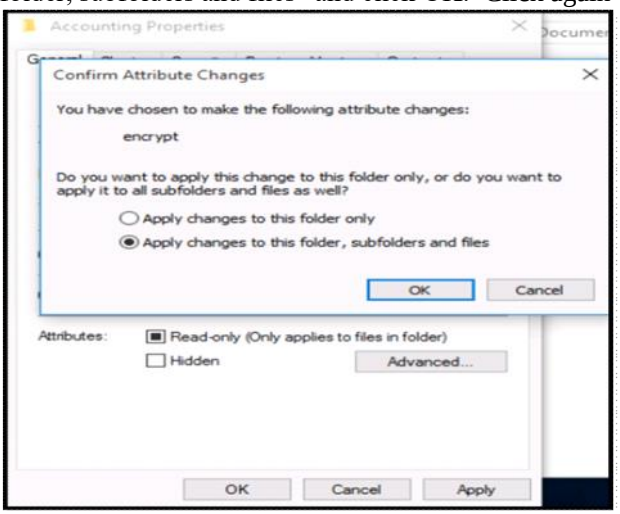
Right click on the Accounting folder and select Properties. On the Accounting Properties window, in the General tab click on Advanced.



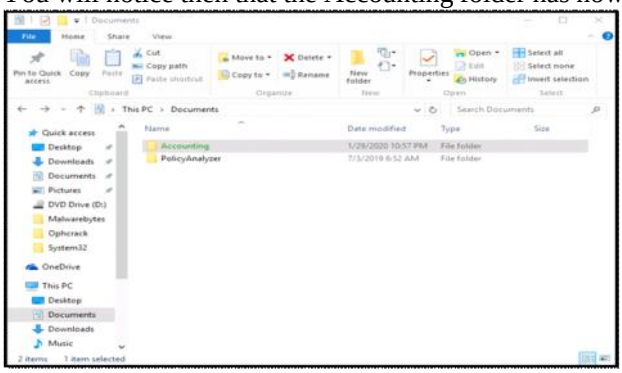
On the Advanced Attributes window click the option “Encrypt contents to secure data”. Then click OK and then Apply.



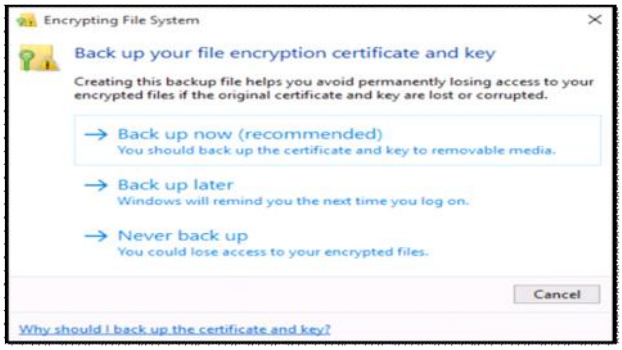
Once you click Apply a new window will open asking you to confirm the attribute changes. Select the option “Apply changes to this folder, subfolders and files” and click OK. Click again OK to close the folder properties.



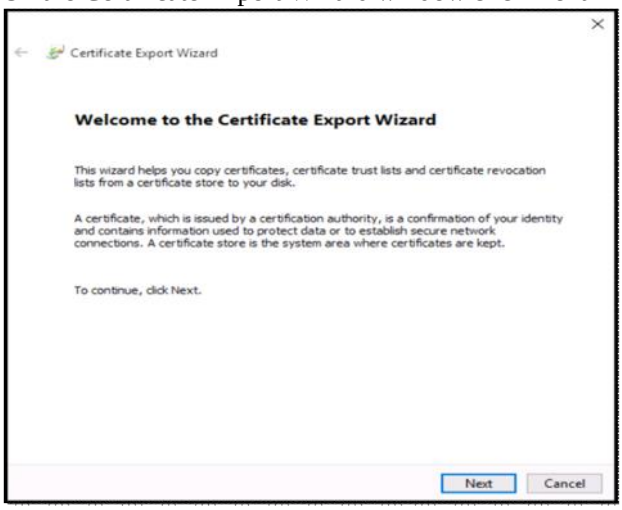
You will notice then that the Accounting folder has now the green color.



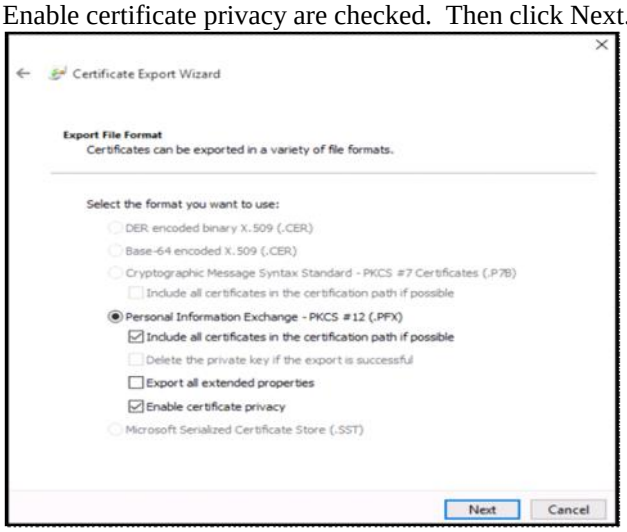
Now that EFS is enabled a small icon exists in the taskbar which is the EFS key backup notice. Always a good idea to backup the file encryption certificate and key. Click on the EFS icon in the taskbar and on the Window Encrypting File System select Back up now.



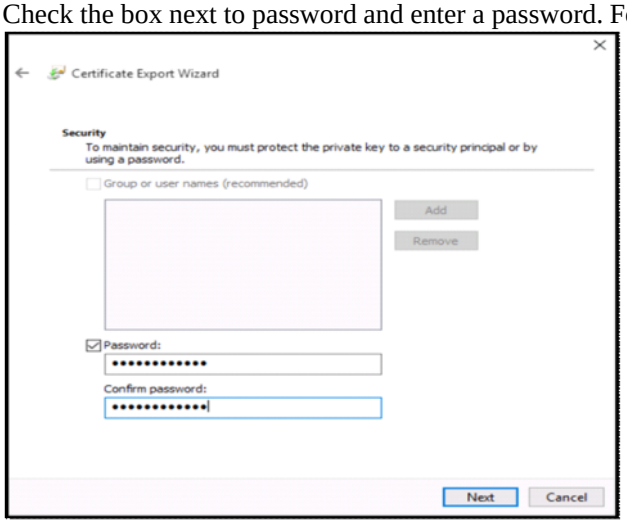
On the Certificate Export Wizard window click Next



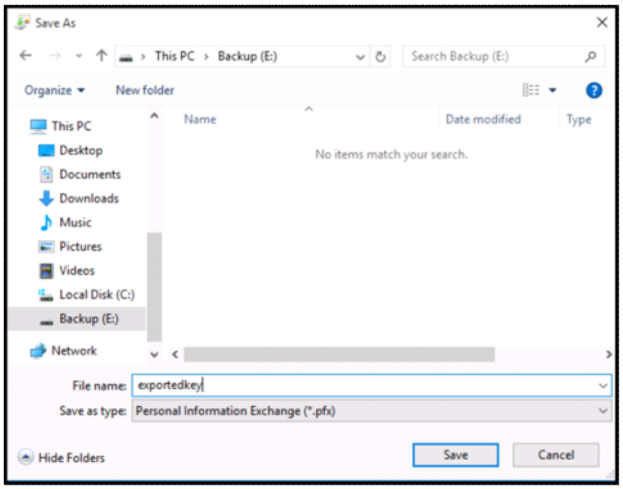
Ensure that the options Personal Information Exchange – PKCS, Include all certificates in the certification path if possible and Enable certificate privacy are checked. Then click Next.



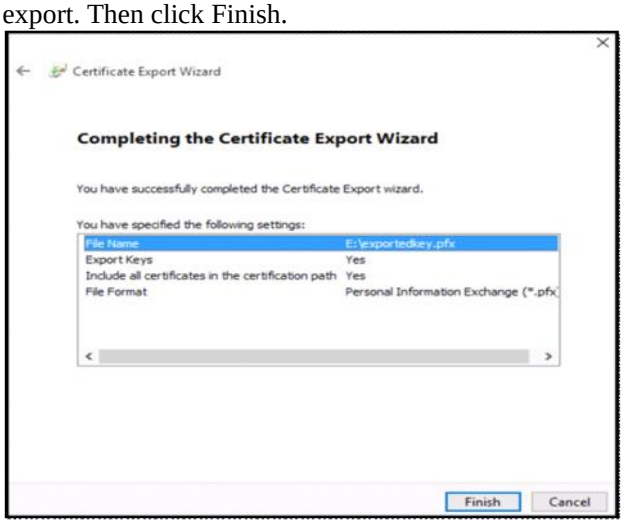
Check the box next to password and enter a password. For password use P@ssw0rd123! and click Next.



Then click browse to select a location and a filename where the key should be saved. Select the Backup drive and for filename use exportedkey.



Click Save and then Next. On the next window simply review the File Name, File Format and other information of the certificate export. Then click Finish.



Hashing

5.5 Hashing

A hash function, **"H"** for example, is a function that takes a variable-size input **"m"** and returns a fixed-size string. The value that is returned is called the hash value **"h"** or the digest. This can be expressed mathematically as **"h = H(m)"**. There are three properties a hash function should have:

- Variable length input with fixed length output. In other words, no matter what you put into the hashing algorithm, the same sized output is produced.
- H(x) is one-way; you cannot "un-hash" something.
- H(x) is collision-free. Two different input values do not produce the same output. A collision refers to a situation where two different inputs yield the same output. A hash function should not have collisions.

Hashing is how Windows stores passwords. For example, if your password is **"password"**, then Windows will first hash it, producing something like:

"0BD181063899C9239016320B50D3E896693A96DF".

It then stores that hash in the SAM (Security Accounts Manager) file in the Windows System directory. When you log on, Windows cannot "un-hash" your password, so what Windows does is take whatever password you type in, hash it, and then compare the result with what is in the SAM file. If they match (exactly) then you can log in.

Storing Windows passwords is just one application of hashing. There are others. For example, in computer forensics, hashing a drive before starting a forensic examination is common practice. Then later you can always hash it again to see whether anything was changed (accidently or intentionally). If the second hash matches the first, then nothing has been changed.

In relationship to hashing, the term **"salt"** refers to random bits that are used as one of the inputs to the hash. Essentially, the salt is intermixed with the message that will be hashed. Salt data complicates dictionary attacks that use pre-encryption of dictionary entries. Is also effective against rainbow table attacks. For best security, the salt value is kept secret, separate from the password database/file.

5.5.1 MD5

MD5 is a 128-bit hash that is specified by RFC 1321. It was designed by Ron Rivest in 1991 to replace an earlier hash function, MD4. In 1996, a flaw was found with the design of MD5. Although it was not a clearly fatal weakness, cryptographers began

recommending the use of other algorithms, such as SHA-1. The biggest problem with MD5 is that it is not collision resistant.

5.5.2 SHA

The Secure Hash Algorithm is perhaps the most widely used hash algorithm today. Several versions of SHA now exist. SHA (all versions) is considered secure and collision free. The versions include:

- SHA-1: This 160-bit hash function resembles the MD5 algorithm. This was designed by the National Security Agency (NSA) to be part of the Digital Signature Algorithm.
- SHA-2: This is actually two similar hash functions, with different block sizes, known as SHA-256 and SHA-512. They differ in the word size; SHA-256 uses 32-byte (256 bits) words whereas SHA-512 uses 64-byte (512 bits) words. There are also truncated versions of each standard, known as SHA-224 and SHA-384. These were also designed by the NSA.
- SHA-3: This is the latest version of SHA. It was adopted in October of 2012.

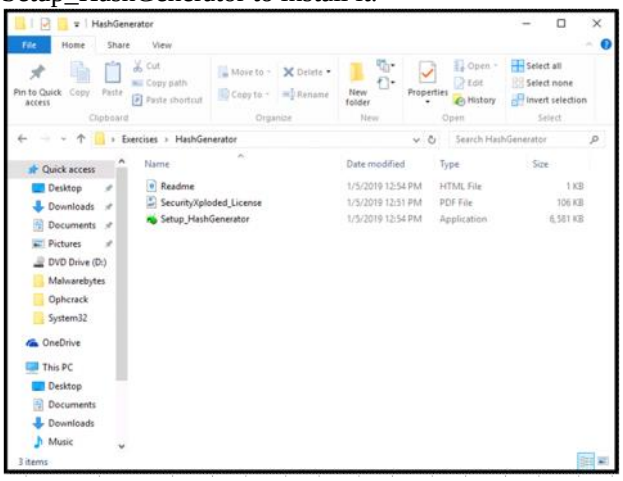
Guided Exercise: Hashing

5.6 Guided Exercise: Hashing

Resources	
Files	None
Machines	Windows 10

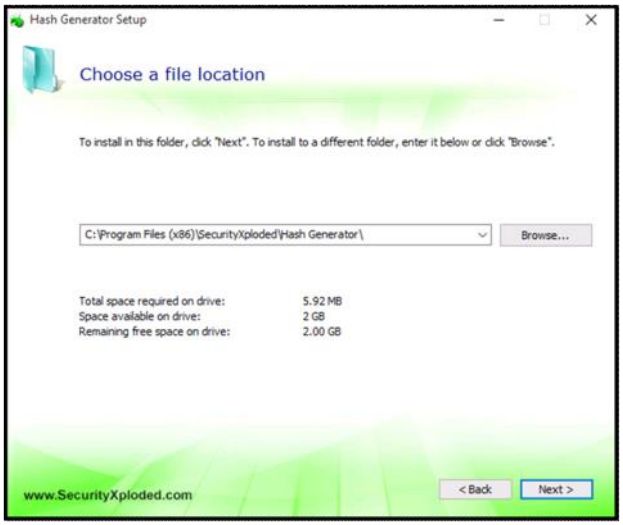
In this exercise we will use a tool called HashGenerator to generate hashes.

In the Windows 10 machine open the desktop folder called Exercises and then the folder HashGenerator. Double click the file Setup_HashGenerator to install it.



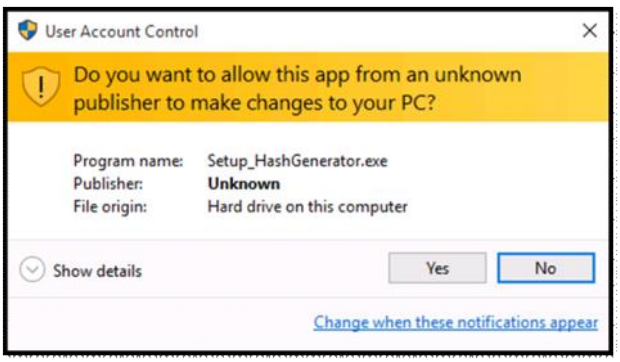
On the Hash Generator Setup window click on Next.

Click again Next on the window Choose a file location.



Click Install on the Begin Installation of Hash generator window.

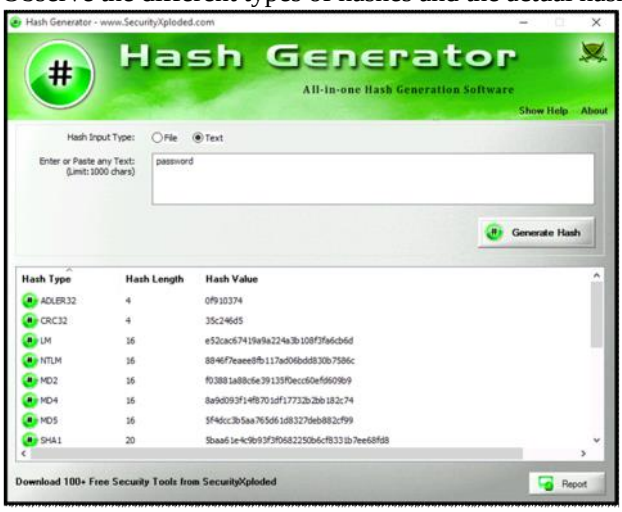
Click Yes on the User Account Control window.



Click Close on the Hash Generator Setup window.

On the Hash Generator tool Select Text as the Hash Input Type and the enter the word password on the “Enter or Paste any Text” text box and click on Generate Hash.

Observe the different types of hashes and the actual hash values.



Cracking Passwords

5.7 Cracking Passwords

Cracking passwords is not the same as breaking encrypted transmissions. If anyone has successfully cracked a password and particularly the administrator/root password then other security measures are rendered irrelevant.

5.7.1 John the Ripper

John the Ripper is a password cracker popular with both network administrators and hackers.

This product is completely command line–based and has no Windows interface. It enables the user to select text files for word lists to attempt cracking a password. Although John the Ripper is less convenient to use because of its command-line interface, it has been around for a long time and is well regarded by both the security and hacking communities.

John the Ripper works with password files rather than attempting to crack live passwords on a given system. Passwords are usually encrypted and in a file on the operating system. Hackers frequently try to get that file off the machine and download it to their own system so they can crack it at will. They might also look for discarded media in your dumpster in order to find old backup tapes that might contain password files. Each operating system stores that file in a different place:

- In Linux, it is /etc/passwd and /etc/shadow.
- In Windows 2000 and beyond, it is in a hidden .sam file.

After you have downloaded John the Ripper, you can run it by typing in (at a command line) the word john followed by the file you want it to try to crack:

- john passwd
- john –wordfile:/usr/share/wordlists/rockyou.txt –rules passwd
  Cracked passwords will be printed to the terminal and saved in a file called john.pot, found in the directory into which you installed John the Ripper.

5.7.2 Rainbow Tables

In 1980 Martin Hellman described a cryptanalytic time-memory trade-off that reduces the time of cryptanalysis by using pre-calculated data stored in memory. Essentially, these types of password crackers are working with pre-calculated hashes of all passwords available within a certain character space, be that **“a-z”** or **“a-zA-z”** or **“a-zA-Z0-9”** etc. These files are called rainbow

tables. They are particularly useful when trying to crack hashes. Because a hash is a one-way function, the way to break it is to attempt to find a match. The attacker takes the hashed value and searches the rainbow tables seeking a match to the hash. If one is found, then the original text for the hash is found. Popular hacking tools such as Ophcrack depend on rainbow tables.

5.7.3 Brute Force

This method simply involves trying every possible key. It is guaranteed to work, but is likely to take so long that it is simply not useable. For example, to break a Caesar cipher there are only 26 possible keys, which you can try in a very short time. But consider AES, with the smallest key size of 128 bits. If you tried 1 trillion keys a second, it could take 112,527,237,738,405,576,542 years to try them all.

NOTE : **RAINBOW TABLES: refer internet**

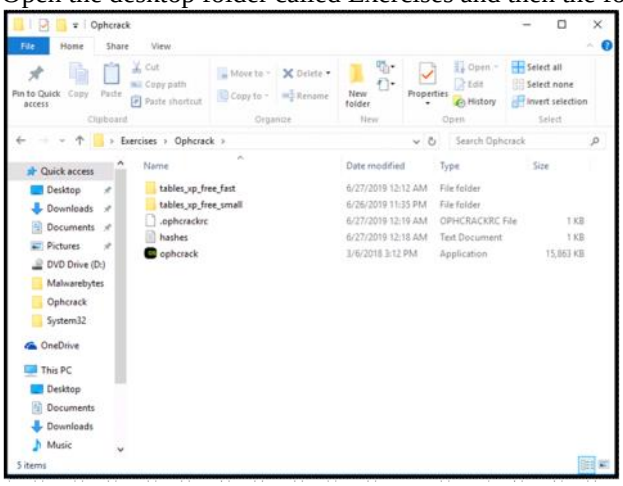
Guided Exercise: Cracking Passwords

5.8 Guided Exercise: Cracking Passwords

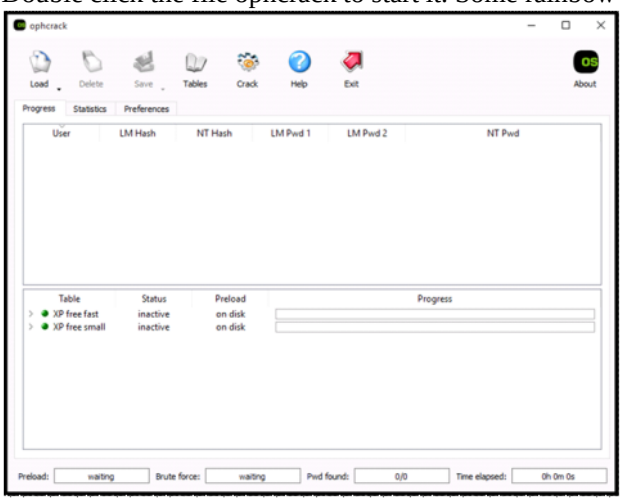
Resources	
Files	None
Machines	Windows 10

In this exercise you will use the tool called Ophcrack to crack password hashes.

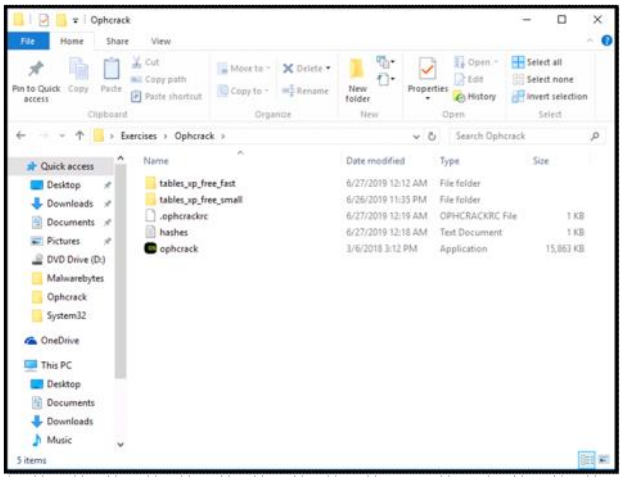
Open the desktop folder called Exercises and then the folder Ophcarck.



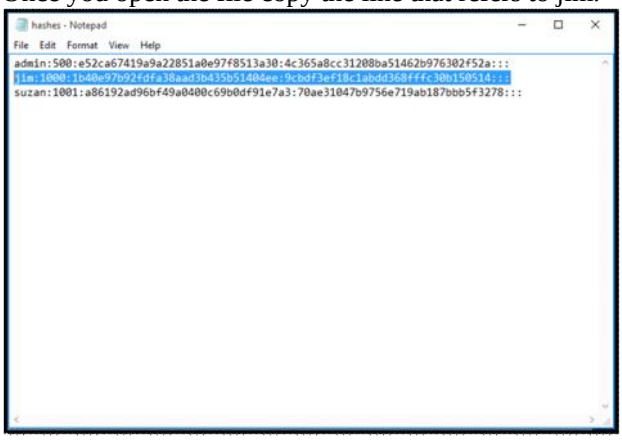
Double click the file ophcrack to start it. Some rainbow tables are already installed within Ophcrack.



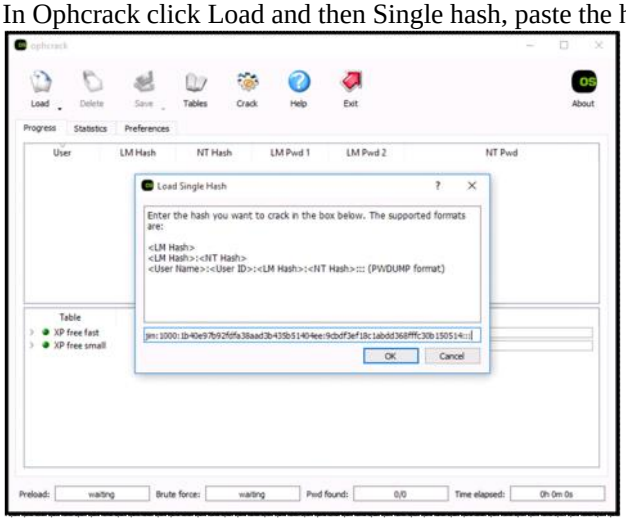
From the Ophcrack folder double click the file hashes to open it.



Once you open the file copy the line that refers to jim.



In Ophcrack click Load and then Single hash, paste the hash on Load Single Hash window and then click OK.



Click Crack to start cracking the password hash. Then you will see the actual password of the user jim.

QUIZ:

- 1.Which of the following encryption algorithms is a block cipher and uses the Rijndael algorithm?
- 2.Which of the following uses key sizes equal to 128, 192 and 256 bits?
- 3.Which hashing algorithm do modern Windows systems use?
- 4.What is a digital signature?
- 5.Secure Multipurpose Internet Mail Extensions (S/MIME) use X.509 certificates to secure e-mail communication
- 6.Which of the following is a symmetric key system using blocks?
- 7.What is the purpose of a certificate?
- 8.Which encryption algorithm uses a variable length symmetric key?
- 9.Which of the following is an encryption method developed by three mathematicians?
- 10.Blowfish is an asymmetric stream cipher