Chapter 6 – Cryptography

What this section is about – what you should learn

This chapter covers Cryptography, which is another pretty huge topic. In one sense cryptography is a pretty simple concept because generally speaking all it's doing is scrambling text or data to protect it. But when you get down to the specifics of how the encryption and hashing algorithms work can get complicated pretty quickly. This is especially true of modern cryptography, which is almost totally based on mathematical theory.

The material in the book it's probably not that helpful if you really want to understand how the different cryptographic methods function as the book material was designed to help people pass a certification exam. As we've discussed previously, the book assumes that you already have a good working knowledge of cryptology, and some hands-on experience. I have a good understanding of cryptography, as I have a degree in math and I teach the Cryptology class, but I find reading through the material in the book is a bit like drinking through a fire hose. What I'm trying to say is don't feel daunted or intimidated by the material in the book. They try to tell you everything that you would normally learn in a college cryptology class or two in 30 pages. As you read the book try and get the general concepts, but don't worry if you aren't able to do the calculations for RSA or ECC. I teach the Cryptology class, and know quite a bit about the subject, and I can barely spell ECC. :} For this class you'll need to have basic understanding of what cryptology is, and know some terminology to pass the test. And since the tests are open book you can always look terms up if you need to.

That being said, there are some key concepts about cryptology that I think you should know, so I'll give you a brief explanation of them here.

The first concept is that cryptology is one of the key components to cyber security. If we could not protect our data by using cryptology the only other way to protect it would be by physical means. That is, if we couldn't scramble our data to keep it private, the only way to protect it would be to somehow prevent other people seeing it. And that would mean physically protecting our data, which would mean not connecting to any network including the Internet.

I'm going to use the (in)famous Alice, Bob and Eve to explain further. The names Alice and Bob are used in cryptography because someone used them in a presentation to replace computer A and computer B. This person then also used the name Eve to represent an attacker, because she would be an eavesdropper. Eve … eavesdropper. Do you get the pun?

Assume Alice and Bob live in different towns. Alice and Bob write letters to each other and sometimes the letters contain information they would like to keep private, especially from that snoopy Eve. If Alice and Bob can encrypt their correspondence the messages they exchange will remain private. It won't matter if Eve sees the letters laying around the house, or intercepts them in the mail if the letters they write are encrypted. But, if they can't encrypt their

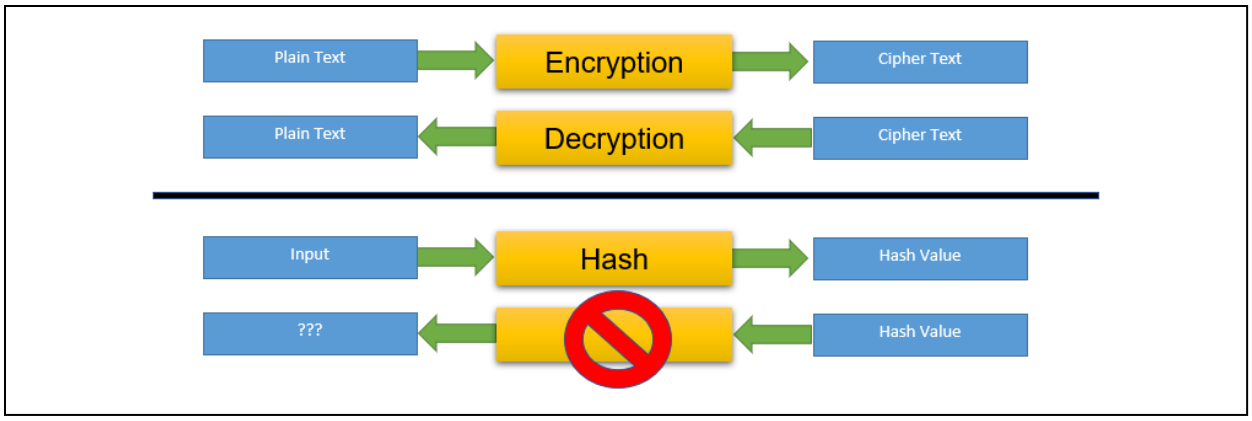
correspondence then it will be easily read by anyone that has access to it. This means the only way they will be able to protect it is to physically protect it. They will need to personally hand deliver each letter, and when the letters aren't in their physical possession, the only way to securely store them will be to lock them away in a place that only they can access.

(This cartoon is from https://imgs.xkcd.com/comics/protocol.png. You may be scratching your head in confusion now. But trust me, once you learn more about cryptography this cartoon will leave you ROFL. If you like geeky humor you should check out XKCD!)

Ok … back to the subject at hand, encryption. The same concept regarding encryption applies to digital data. If the data can be encrypted it can be stored or transmitted with the expectation that it will remain private. But if it can't be encrypted then the only way to protect it would be to hand deliver it, and physically lock it away when it's not in use. So, without cryptology and encryption there would literally be no cybersecurity. There would be no passwords to prevent unauthorized users from gaining access to systems or data, there would be no encryption to protect data, there would only be physical protection.

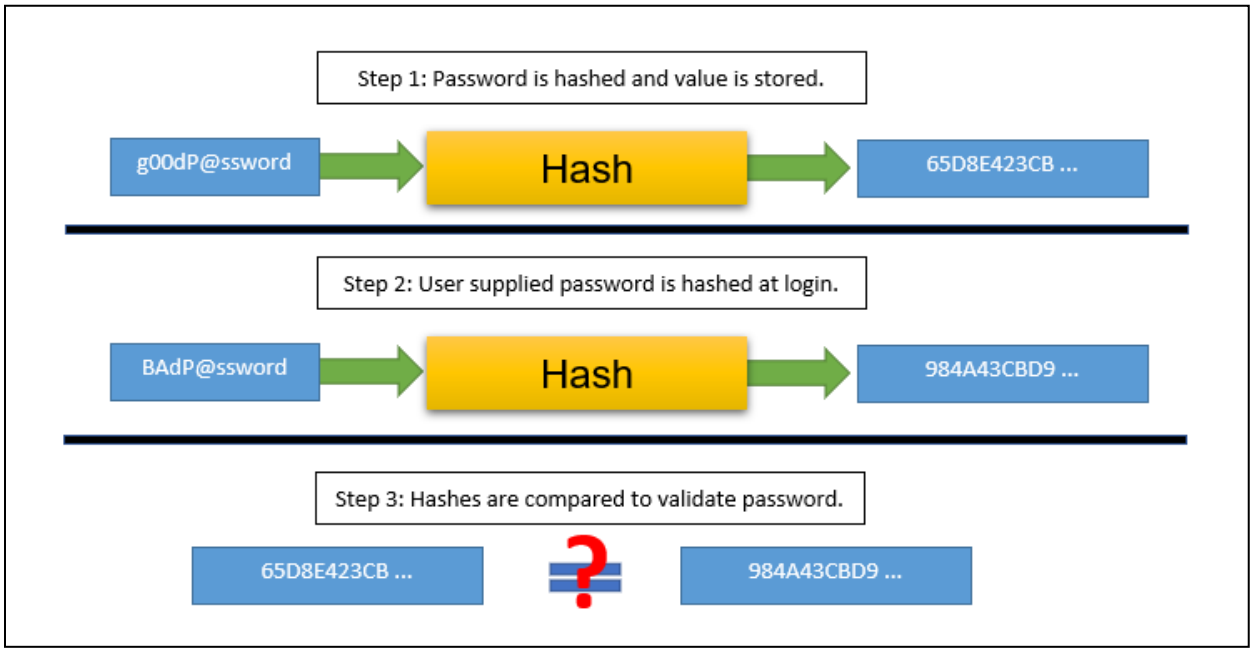
I want to make this point very obvious, so here it is again. If there were no cryptology there would be no cybersecurity.

The second key concept is the difference between hashing and encryption. These are both used to scramble data, but there is one huge difference between them. Encryption is reversible, while hashing is not. That is, any information that is encrypted can be decrypted, but any information that is hashed cannot be directly unhashed or dehashed.



This means that if you want to share information with anyone else it must be encrypted. When the recipient gets the encrypted information they can decrypt it, and then read the original information. If you hash a message and send it to someone, there is no way for them to get the original message back.

So when is hashing used? One of hashing's main purposes is to store things like passwords. We don't want to encrypt passwords because if someone were able to gain access to the file storing the passwords, or intercept a password while it's being transmitted, they would be able to decrypt it. But if we store hashed passwords then we don't have to worry about anyone decrypting them, as this is impossible.



I'm sure you're wondering how a password system works, so here's another brief explanation. When you set a password, it's hashed and the hash value is stored in a file. When you try to login, the login program prompts you for your password. It takes the password you enter and

hashes it, and then compares this hash value with the one stored in the password file. This is a pretty slick system, which protects passwords while they're being stored, and also provides a way to protect them as you login.

This is a super important concept, so here's a couple of quick questions to help you check your comprehension:

1. Assume you want to buy a new guitar for your favorite teacher. You login to an online auction site that requires a password so you can bid on a guitar. How is your password protected at the online auction site?
    a. The password is encrypted before it's stored.
    b. The password is hashed before it's stored.
    c. None of the above.

2. Assume you are the highest bidder in the guitar auction. As you complete the purchase you enter your credit card information. How is this information protected while it is transmitted to the auction site?
    a. The information is encrypted before it's transmitted.
    b. The information is hashed before it's transmitted.
    c. None of the above.

The answers are **b**. for question 1, and **a**. for question 2. Hopefully this makes sense. But if not, don't worry too much as you'll be learning a lot more about hashing and encryption in later classes.

There are dozens of different encryption and hashing algorithms, and the differences between them probably won't make much sense at this point. (But don't worry, you'll learn all about them if you take the Cryptology course.) So once again, the main thing you need to remember is that hashing is used for passwords, and encryption is used for almost everything else.

The last key point I think you need to grok is that none of the current encryption or hashing methods are completely invulnerable. They are difficult to break if implemented correctly, but everything can be broken given enough time. The entire history of encryption is littered with methods that were once considered unbreakable, and were very difficult to crack by hand. But someone, some very, very smart someone, found a way to crack them. Our current encryption and hashing systems are also not unbreakable. The time needed to crack something may be measured in millions of years, but everything can be broken sooner or later. And, you may be aware that quantum computers are able to break classic computing encryption schemes much more quickly, at least in theory.

The other thing to be aware of is that the encryption and hashing schemes must be implemented correctly to provide protection. For example, you may remember the WEP protocol for wireless security. It was the standard when wireless routers first became popular. It used an encryption scheme that allowed different length keys, but a very short key was

chosen for the actual implementation. This made WEP traffic very easy to crack. Another more recent example is the way that Samsung implemented security on their Android phones[1].

The technical details behind these implementation problems can be difficult to understand, so let's use this analogy. This is kind of like saying we have a box that you can lock your messages in. The lock on the box will only open if the correct password is provided. You can set the password to be anywhere between 2 characters and 200 characters in length. If a 200 character password is used, someone could possibly find the correct password given enough time, but it will probably take them hundreds of years before they stumble onto the correct combination of characters. But if a 2 character password is used it can be guessed fairly quickly. The locks will be very secure if they're implemented correctly, but they can also be almost worthless if they're set up wrong.

Applying What You've Learned to Your Home System(s)

This brings us to looking at practical things you can do to protect your own devices. There are a few things regarding cryptography that you can apply to your personal security. As we've talked about several times previously, you need to use strong passwords and you'll learn more about password strength and how to check your passwords in a later chapter. The other thing you might consider is using encryption to protect your files, folders, or even your entire drive. The reason I say you might want to do this is that using application encryption or full disk encryption requires remembering passwords, and if you forget your passwords you will lose access to your data.

I've made a few videos that will show you how to use a couple different methods to encrypt files, folders and entire drives on Windows systems. You'll learn about when it would be appropriate to use the various methods as well as how to actually perform the encryption and decryption. The only you'll have to try is EFS encryption, which you'll use for one of the assignments for this section.

You also might want to consider encrypting the files on your phone. I believe that iPhones are encrypted by default. I don't own an iPhone, but I did a little research and it looks like this is the case[2]. Android phones used to have a full phone encryption feature, but decided that it had the potential for causing issues with things like updates or emergency calls, and have moved to an optional feature called Secure Folder that can be used to encrypt the files of your choice. The following web page has information about Secure Folder and instructions for setting it up. If this page no longer exists, or you want to learn more you can do your own search for something like Android Secure Folder.

https://www.androidauthority.com/samsung-secure-folder-908758/

[1] https://www.techrepublic.com/article/100-million-samsung-phones-affected-by-encryption-weakness/
[2] https://www.lifewire.com/encrypt-iphone-5193023

Ways to Check Your Comprehension

The test over this chapter won't be for a few weeks, so I suggest that you use the review questions at the end of the chapter to check your comprehension. I don't have the review questions loaded in Canvas, so you'll have to just read them and figure out your answers on your own. Or you could try and connect with some other students in the class and drill each other using these questions.

The material in this section will be included in Test 1, which isn't due for a few weeks. If you look in the Test 1 Canvas Module you'll see it contains a link to a Practice Test. You can take the Practice Test as few or as many times as you want. You're not required to take the practice test, but it's also a good way to check your comprehension and prepare for the "real" test.

The Activities for This Section

There are three activities for this section, two hands-on projects and a writing assignment.

Hands-On Projects

Hands-On Project #6-3

In this project you'll get some hands-on experience with a few different hash functions. You'll see what happens when you give a hash function input of different sizes, and what changing just one character can do to the hash output.

The program that you'll use for this exercise, Hashcalc, is pretty simple to use, but it's a little tricky to download. It's especially tricky if you follow the download instructions in the book. To help you navigate past some of the traps in the download process I've created a short video which I urge you to watch.

https://tonysako.com/cs150-tips-for-download-hashcalc-video/

Also, you're going to need to do some text editing. If you do this exercise on the VM you'll find that Microsoft Word is not installed on the VM. If this happens, try and problem solve on your own. That is, try and think of a different way you can edit a file from the VM. Remember that you'll run into situations like this in the workplace, and your employer will expect that you have the ability to figure out a solution. If you can't think of any alternate ways to edit a text file on your own, look at the end of this document and I will provide some help.

When you do this exercise try to keep a couple of things in mind. The first is just what the heck is a hash and why should you care about them? The answer to the second part of the question is that hashes are used to store passwords in a secure way, and they can also be used to prove that a set of data has not been changed.

Hash functions work by performing calculations on any amount of data, from a single character to all of the data on an 8 TB hard drive, and returning a single number. This single number is

called the hash signature or hash value, or sometimes just the hash. There are hundreds of different hash functions, but only a handful are useful for security. To be used in security a hash function must be something called cryptographically secure, which means it must possess these properties:

1. The hash must be a one-way function. That is, given a hash value there must be no way to calculate the data input to the hash. This is what makes hashing work so well for passwords.
2. Regardless of the size of the input, the output must always be the same size.
3. Different inputs must result in a different hash value.

The hash functions you will use in this exercise were all once considered adequate for use in cryptography. Researchers have found problems with a few of the hash functions in the list, but they can all still be used to demonstrate the properties listed above.

What to submit for Project 6-3:

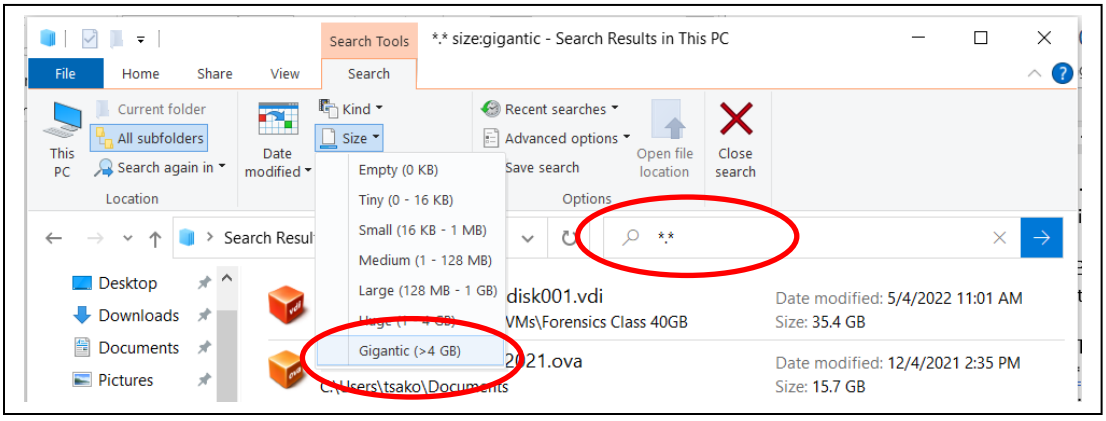
- Answer all of the questions
- Include screenshots after steps 15 and 20
- You must also complete the following steps 23, 24, and 25, and provide answers for 23 h., 24 g., and 25 b. Note that these steps are not included in the book. They are an extra bonus that I created especially for you!

23. This step you will provide another demonstration of how slight changes to the input will cause extreme changes to the hash output or signature. You will do this by creating 3 files, each of which contains a single number, and then comparing the hash values for each file. You will then see if you can use the first 3 hash values to predict what the hash of the next number will be.
    a. Create a text file that contains the single number **1**. Save this file with the name **1.txt**.
    b. Create a text file that contains the single number **2**. Save this file with the name **2.txt**.
    c. Create a text file that contains the single number **3**. Save this file with the name **3.txt**.
    d. Create a text file that contains the single number **4**. Save this file with the name **4.txt**.
    e. Reopen Hashcalc if necessary
    f. Create the MD5 hash for files 1.txt, 2.txt, and 3.txt. Save each hash value to a text file as you create them.
    g. Inspect the 3 hash values and see if you identify any patterns or sequences that will allow you to predict what the hash signature will be for the 4.txt. If you find a pattern, add your prediction to the file of hash values.
    h. Create the MD5 hash for the file 4.txt. Were you able to predict this value? That is, was there any pattern you could detect in the hash output for the first three

files that would allow you to predict what the hash output for the 4th file would be? Or does the hash output for each file seem to be random even though the characters in the files are sequential?

24. This step will demonstrate that a specific hash function will create the same size output regardless of the size of the input. In this exercise you will hash two more files, one that is 41 MB, and another that is the largest you can find on your computer. You will then compare the size of the hash output you created in the previous step, for a file containing a single character, with the hash output for the 2MB file and the largest file you could find.

    a. In this step you will copy the MD5 hash output for any of the files from the previous step and paste it into a text file. You will then create MD5 hash outputs for two larger files, and compare the size of each output. Start by copying and pasting the hash output from one of the 1 character files from the previous step. If necessary open Hashcalc, load any of the files from the previous step, calculate the hash, highlight the hash output value, copy it, and paste it into Notepad, or any text editor.
    b. Download the file **gc.jpg**. Notice that it is ~41MB in size. Use Hashcalc to create the MD5 hash for this file, then copy and paste the hash to the same document you created in step A.
    c. Find the largest file you can on your computer. The easiest way to do this is to open Windows Explorer then type the string *.* in the Search Box. Each * in this string is a wild card that will match anything, so this is telling Windows to search for any file. It actually doesn't matter what string you type initially, you can type any characters as you're just doing this to get the **Search Tools** tab to appear. Typing *.*, or any text in the search box will open up the Search Tools tab in Windows Explorer.
    d. If you didn't type the string *.* in the search box, do it now. Go to the Size drop down in the Search Tools tab, and select Gigantic. This will tell Windows to search for files with any name that are bigger than 4 GB.



e. Windows will search for large files, which may take a minute or so, and display the results. If your system doesn't have any Gigantic files, search for Huge files or Large files until you find some. Note the path to the largest file on your computer.
f. Return to Hashcalc and open the file from step e. Calculate the MD5 hash for this file, and copy the hash output to your text document.
g. Compare the size of the hash output strings for the three hash values. Does the size of the input have an effect on the hash output, or are they the same size? Assume that you calculate the MD5 hash for a 3 TB hard drive. What will the size of the hash value be? Write your answer as the length of the hash value, or the number of numerals in the hash value. For example, if the hash value is 4A7B the length would be 4.

25. This step is designed to demonstrate that hashes are one-way functions, which can't be reversed. It has also been designed to provide you with experience cracking a hashed string.
    a. Assume that you know that the hash value for a string is:

       `a87ff679a2f3e71d9181a67b7542122c`

       Can you tell me what the original string was? That is, what string was hashed to create this output? Do you see a Dehash button anywhere in Hashcalc? The answer to all these questions is No. Remember that encrypted files can be decrypted, but hashed values can NOT be dehashed. Hashing is a one way function, which means there's no way to reverse the process.

    b. While you can't dehash a string or file, you can crack a hash value. Cracking is the process where discover an input string by hashing other strings, then comparing the hash outputs. If they're identical you know the string you just hashed is the same as the string that created the hash value you're trying to crack. In this example, I will arm you with the knowledge that the original input was a single number or numeral between 0-9. To check each number, start Hashcat then go the **Data Format** dropdown and change it from **File** to **Text String**. Next, enter the number **0** in the Data box, then click the Calculate button. Check the MD5 hash output, and compare to the hash value in step a. If they match you know that the number 0 was the original input. If they don't match, enter the number **1** in the Data box and recalculate the MD5 hash. Repeat this process until you find the string that recreates the same hash value listed in step a. Write the following in your homework, substituting the original number you found for *x*:

       The original number was *x*

26. [5 points Extra Credit] What was the original input used to create the following MD5 hash output?

    `d2ddea18f00665ce8623e36bd4e3c7c5`

    Here's a hint – the original input was a number between 0 and 99. If you want the extra credit points, write the following to your homework substituting the original number you found for *x*:

    The extra credit number was *x*

Hands-On Project #6-4

Note: There are a few things that may make it difficult for you to complete this exercise, so I've created a video to assist you. I've also provided written instructions below, but I strongly urge you to watch this video before starting the exercise:

https://tonysako.com/home/cs150-introduction-to-computer-security/cs150-3-configuring-efs-and-starting-service/

Hands-On Project 6-4 will not work with the Home version of Windows. If your computer is running the Home version of Windows you will have to use the Virtual Machine to complete the assignment. If you use the VM you'll have to create a plain text document instead of Word document, since Word is not loaded on the VM.

The point of some of the questions in this exercise is that when you access a file that's been encrypted it has to be decrypted before you can open it. You should see if there's any noticeable latency or delay while the file is being decrypted. When I think about all of the math and calculation that has to occur to decrypt a file it seems to me that it should take a significant amount of time. But you'll get to check the delay for yourself …

What to submit for Project 6-4:

- Answer all of the questions.
- Include a screenshot after step 8

When you try and turn on EFS on the VM you may get an error message about not being able to start the service, as explained in the video. But, you may also have trouble starting the EFS service from the Services App. If that happens use the following process to start the EFS service:

1. Login to the VM with the Admin account

2. Do the search for the Services App and start it. You can either just start it, or right-click and choose Start as Administrator.
3. Find the EFS service on the list. Right-click on it and choose Properties.
4. Find the Startup Type box, and change it from Disabled to Manual or Automatic.
5. You'll now be able to select Start in the Service Status section.

Also note that Hands-On Project 6-4 only shows you that EFS is easy to turn on, and works very quickly; so there's very little impact on your work flow. But how do you know the encryption is actually working? If you really want to test the EFS encryption you'll have to login as a different user and try to open an encrypted file. If you want to try this you can use the following steps. These steps aren't required and won't be graded. They're just provided in case you want to really test EFS.

1. Turn on the EFS as directed in 6-4. (To make things easier later, you might consider creating the EFS file in the root of the C: drive, instead of in your Documents folder.)

2. Create a different Windows User

3. Log out of Windows, then login as the user you just created

4. Find the file you encrypted with EFS. (This is why it's easier to create it in the root of the C: drive.) Try and open the file. Are you able to open it?

Case Project (Writing Assignment)
Choose one of the following Case projects:

6-1, 6-2, 6-3, 6-4, 6-5 or 6-7.

Research the topic then write a paper consisting of at least two to three paragraphs ensuring that you answer all of the questions asked. Make sure to include all of the required items. For a full list of required items and details on how your paper/report will be graded you can refer to:

https://tonysako.com/home/cs150-introduction-to-computer-security/cs150-class-introduction/cs150-guidelines-for-writing-assignments/

Ways to Edit a File on the VM

There are a few different ways to edit a text file on the VM, two good ones, and one not so good. First, you could use Notepad or Wordpad as these utilities come with every Windows installation. Second, you could use a cloud based editor such as Google Docs or Microsoft Office Online (or whatever Microsoft is calling it now). Or the third solution, which is not so good, is to edit files on your PC and then put them on a cloud storage server or email them to yourself, then login to the cloud server or your email and download the files.