Encryption: The History and Implementation

by

Jacob Levinsky

Submitted to the Department of Mathematics and Computer Science
School of SUNY Purchase
in partial fulfillment of the requirements
for the degree of Bachelor of Arts

Purchase College
State University of New York

May 2022

Sponsor: Dr. Knarik Tunyan

Second Reader: Dr. Athar Abdul-Quader

Table of Contents

Contents

Abstract

This thesis seeks out to look back at the history of encryption and see how it has evolved. The research will first look at what encryption started out as and as time went on, see how encryption has as well. The research will show how at each step of the development of our encryption methods, how secure were they as well as the steps taken to keep the information away from those who aren't supposed to have it. Along with the research, I will create my own encryption software and write about what steps were taken to ensure the security of encryption. From all of these points of research and development, I will conclude how cryptography and encryption are used to provide security, as well as how important it is for today's age.

Keywords: Encryption; Cryptography; Software; Security

1. Introduction

Throughout history of humans, there has been one form or another of humans keeping text. As time went on, people started to hide their texts so that others couldn't know what they were, and that process has evolved. Encryption in its most basic idea is cryptography, which is just a technique in which messages can be secured. Back as far back as 1900 BC with the Egyptians, they used hieroglyphics to write their texts and some evidence shows that there are some different hieroglyphics that are not same as others suggesting the use of cryptography. Throughout history, evidence of this continues, even before what we call the current era, and we can see that civilizations have their own versions of cryptography and encryption. What I will be researching and looking at is that history of encryption and cryptography from its early stages to how it is used today.

1.1. History of encryption and cryptography

Back as early as 1900 BC, Egyptians kept their texts in their own language that was universally used called hieroglyphics. Back then, we have seen how Egyptians used hieroglyphics to keep their texts, communicate, create plans, and just about any and everything else. There is some evidence on how some of the hieroglyphics used in their texts are different then some of the typical hieroglyphics presenting the theory that it was used as a cypher to create cryptography. Another civilization that has evidence of the use of cryptography are the ancient Spartans. Back in 500 BC, the Spartans had a device that those theorize that it was used to messages. What the Spartans had was a device that was "called a scytale to send secret messages during battle" (Thales). The device had a leather strap that had letters on it, and the "letters on the leather strip are meaningless when it's unwrapped, and only if the recipient has the correctly sized rod does the message make sense" (Thales). Later in the more modern era, around 1917, there is an American named Edward Hebern who invented an "an electro-mechanical contraption" that "uses a single rotor, in which the secret key is embedded in a rotating disc" (Sidhpurwala) that was called the Hebern rotor machine. The device was used to encrypt messages by having the key encode "substitution table and each key press from the keyboard resulted in the output of cipher text" (Sidhpurwala). This was an early encryption device that later the Enigma machine built itself upon. In Germany during WW1, around 1918, a German engineer named Arthur Scherbius started work and invented the Enigma machine. The Enigma machine used 3 or 4 more rotors and how it works is that the "rotors rotate at different rates as you type on the keyboard and output appropriate letters of cipher text" (Sidhpurwala). There are more examples of the progression of the history of cryptography and encryption devices that will be explored further, as well as the examples brought up.

1.2. The difference between cryptography Vs encryption

There is not much of a difference between cryptography and encryption except that one goes into the other. Cryptography is "the science of concealing messages with a secret code" (Thales) which is typically used by a cipher. A cipher is a "method of transforming a message to conceal its meaning" (Britannica). Encryption is "the way to encrypt and decrypt the data" (Thales). To encrypt a message or data usually used the use of cryptography and the use of a cipher. Some examples of different encryption methods and ciphers are the Caesar Cipher, Hill Cipher, and Reverse Cipher. All of these have different levels of "security" depending on how complex the

cipher is. By using the Reverse Cipher, all that is done is that the alphabet is reversed so the A becomes Z, B becomes X, and so on. By taking the plain text and putting it through the cipher, all is needed to be done is reverse the process. Similar is done with the Caesar Cipher, which is the alphabet, but the letters are moved 3 spaces down so that A becomes D, B becomes E, and so on. Another similar cipher is the Hill Cipher where the alphabet is associated with its corresponding number, so A becomes 1, B becomes 2, and so on. Cryptography has a lot to do with encryption, especially now when it comes to algorithms.

1.3. Algorithms

As cryptography evolves and advances, so does its methods. Today, most cryptography and encryption methods use algorithms. An algorithm is a process that is used in calculations typically by a computer. Today's encryption methods use these algorithms, typically with software, and they are based on the old cryptography methods. One kind of algorithm is called the Symmetric-key algorithm and it can also be "referred to as a secret-key algorithm" and it "transforms data to make it extremely difficult to view without possessing a secret key" (Turner). Another kind of algorithm is called the Asymmetric-key algorithm, also known as the "public-key algorithms" and it "uses paired keys (a public and a private key) in performing their function" (Turner). Just like in the encryption methods where they have different ciphers, algorithms also have their own version of a cipher, which is called the key. Without the ciphers, the encrypted message won't be able to be seen, just like without the key whatever the message is won't be able to either. Algorithms are a huge advancement and importance to cryptography because they are used for things "such as data encryption, authentication, and digital signatures" (Spies). Just about everything in today's life revolves around algorithms either controlling or being a huge part of how things work. Without them, anyone could access bank accounts, steal personal information, log into people's emails, just about do anything.

2. Early cryptography

Looking back to how cryptography was used, the Spartans and other early civilizations like the Egyptians were early adopters of them. The Egyptians had their hieroglyphics which was used for their texts. Hieroglyphics showed things such as how they lived their lives, their rules, and just about everything else. One of the ways that Egyptians used their cryptic hieroglyphics were by monks and they used "non-standard hieroglyphics to keep anyone outside of their inner circle from understanding what was being communicated" (Fuerst). Spartans also had their own version of cryptography that was used during war time which was called the scytale. Just like modern ciphers, it uses a type of key to decipher the messages. How the scytale works is that it "used a cylinder base, such as a stick or baton, wrapped with a leather or parchment strip wound spirally around it" (Fuerst), and the stick or baton would be the key. The message would be on the leather strip and without the correct size stick, the message would just be random letters and wouldn't make sense. The key in this case is crucial to decipher the messages, just like we need the cipher keys in encryption to know what the messages are.

2.1. The Spartans

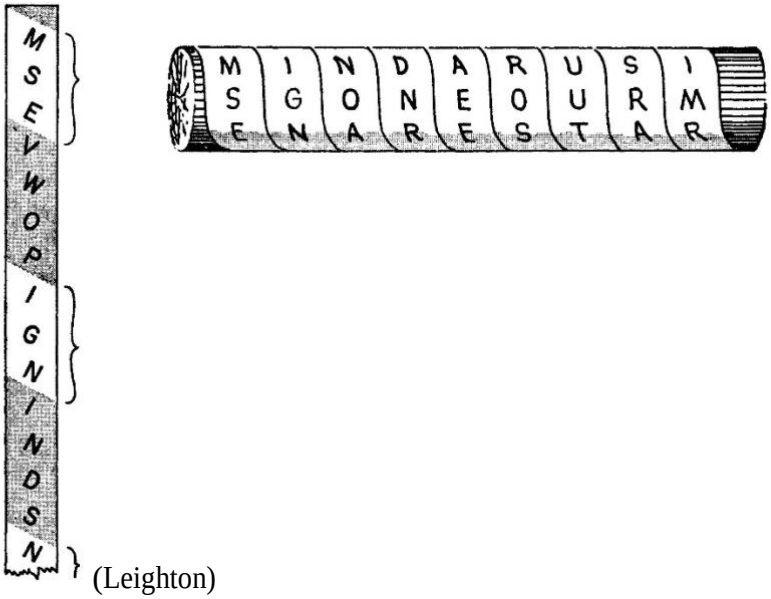
One of the early civilizations that used cryptography methods to send messages between each other were the Spartans civilization, around 500 BC. Cryptography is very useful for many things such as masking messages so that one those who need to know what the message is can only see it. The Spartans had one very useful cryptic device that more modern devices and methods can be based on today. The scytale is an early cryptic device that the Spartans used to relay messages during times of war. Like it was said before, how the scytale works is by using "a cylinder base, such as a stick or baton, wrapped with a leather or parchment strip wound spirally around it" (Fuerst). The scytale was used a lot during times of war and it "was a practical means of secret communication" (Leighton). Since the scytale was comprised of two parts, it could be determined to be a very easily lost or broken method, but one way the Spartans would combat this was by having "a split tally-stick, half of which was carried by the general" and "the other half was to be retained by the Spartan authorities" (Leighton). This made it so that since part of the key was in pieces, the two had to be brought together to make it so that the message could be read. One of the drawbacks that the scytale was the strip that held the message had to be full otherwise it wouldn't be as affective. To be properly made, the "rest of the scytale is filled in with random letters (null)" (Leighton) and that would make it so that if a person is just looking at the strip and tried to read it, they would be confused. Since the scytale was greatly used but had its downfall, there was one Roman that is attributed to creating another method called the Caesar Cipher.

2.2. Julius Caesar

Around 100 BC, Julius Caesar was a Roman who a general and during times of war thought of a new way of sending messages secretly. While the Spartans had used the scytale to send messages secretly, Julius is attributed to a new cipher that was used during their time of war. How the Caesar Cipher worked was by the "plain text is replaced by the letter standing a predetermined number of places from it" (Leighton). As said before, the cipher takes the alphabet and shifts it a certain amount, so if the shift is 3, the whole alphabet moves 3 so that 'A' becomes 'D', 'B' becomes 'E', and so on and so forth. This type of cipher is called "the letter shift, or substitute cipher" (Fuerst) and what a substitute cipher means is that it is a method in which the character in the message is replaced with the characters in the ciphertext. The cipher could be thought of as a better cryptography method than the scytale, but both have their pros. Both the scytale and the Caesar Cipher, if someone looks at the message that is being delivered it will just look like a bunch of random letters that have no real meaning. One of the differences between the two is how they are deciphered; both need a key but the method in which it is done are two different ways. The scytale needs the stick that makes it so that the message can be properly looked at and the Caesar Cipher needs the key in which the message got shifted by. As said before, for the scytale's key, a split tally-stick, half of which was carried by the general" and "the other half was to be retained by the Spartan authorities" (Leighton), and for the Caesar cipher, it was "rumored that he would tattoo the key onto the shaved head of the messenger" and when their " hair grew back, the key would not be visible" (Fuerst).The Caesar cipher was mostly used for the secrecy that it provided and since the cipher just shifted the letters in a message, it could be easily broken. One of the methods in which these types of ciphers could be broken would "by using the frequency of letters in the language" (Sidhpurwala) and by doing that, you could tell how much the letters got shifted. Just as these cryptography methods were widely used in these times, mostly is times of war, there have been many more to evolve and grow.

2.3. The Secureness Behind Old Cryptography

After looking at some different cryptography devices such as the scytale or Caesar cipher, they aren't that secure when it comes to their encryption. With the scytale, the secureness of that cryptography device was dependent on the stick that was being used as the key. The message itself has its own secureness since the message is scrambled between other random letters. The message is only visible when it is wrapped around the stick, otherwise it looks like bunch of unimportant letters. The stick must be used to do so, and "the key was the diameter of the cylinder base" (Fuerst).



(Leighton)

From these figures from the text by Leighton, it shows how the scytale was used. When the strip that has the message is unwrapped it is just a bunch of random letters that don't seem to follow any visible pattern, but after it is wrapped around the proper stick, they encryption key, then the message is revealed. With the Caesar cipher, there is a different secureness level compared to the scytale. Since the cipher works by shifting the alphabet, and therefore the message, it could be easily cracked. When it comes to the Greeks for where the Caesar cipher was developed, there are only 24 letters in the alphabet so one could "simply guess up to 25 times to decrypt the message" (Fuerst) which is a big flaw. Another flaw that the Caesar cipher proved to have would be the structure of the message doesn't change, therefore looking at the message structure could "reveal patterns to easily break the code" (Fuerst). After looking at all these flaws in these different encryption methods opened flaws that I would have to overcome in my own encryption algorithm.

3. Modern encryption

In more modern encryption, things are still physical like how cryptography methods were in the first iteration but with a touch of modern technology. Machines such as the Enigma machine was a "rotor-based cipher machine" (Fuerst). How the Enigma machine worked was it used the rotors to turn each time a letter was pressed, which in the process would assign that letter a new random letter. As the person using the machine typed out the message, after one letter was pressed a "electrical current went through a series of rotors and a light was illuminated on a lamp panel" (Fuerst) and the person would take note of what was lighting up. The new letter that was assigned becomes part of the encrypted message, but to decipher it the person must also know how the Enigma machine was set up before the message was typed. The Enigma machine itself is both the key as well as the encryption machine.

3.1. The Enigma Machine

The Enigma machine is one of the modern inventions for encrypting a message. The Enigma machine uses the “Enigma cipher device developed in the mid 20th century” (Brown) and it was developed during WWII. Just like the scytale was used to send secret messages, the Enigma machine was used by the Germans to send encrypted messages. How this was done was with rotors that would change after each letter was inputted into the message. There were wires that would pass through the rotors that would scramble the lettering so that no letter input would be the same. To be able to decrypt the message that was created by the Enigma machine, the user had to know the original starting positions of each rotor. The Enigma machine was extremely proficient in its encryption where it " confused protected information by breaking expected language rule” (Brown) therefore making it extremely hard to break. This advantage for those who used to Enigma machine to encrypt their messages because if their message got intercepted, the interceptors won’t be able to decipher the message without the original knowledge. This made the Enigma machine one of the best encryption devices during its time and including to today’s era even though its cipher has been broken. The level of encoding that the Enigma cipher had created made the pathway for future encryption methods.

3.2. End-To-End Encryption

In today’s day in age, we rely heavily on our technology for our everyday tasks such as banking, money transactions, and messaging each other. WhatsApp is very popular messaging application that is used by millions of people around the world, and they have been stated that they have an end-to-end encryption, also called E2EE, for all the messages. When it comes to being able to send private messages to someone through a medium such as WhatsApp, it is important that only you and the person that you are trying to talk to is able to read your messages. WhatsApp’s E2EE does just that, make it so that only you and the person intended for the message able to see the message. Now this type of encryption is not new, but due to the popularity of WhatsApp, “the company has installed end-to-end encryption ‘because as a company they believe in your right to have private conversations when you use their service’ (Burrell, 2017, par 33)” (Santos). How E2EE works is by encrypting the message as it is doing out, then when it is received on the other end it is then decrypted. E2EE is known as an asymmetric encryption which means it uses two keys for the encryption instead of one key. This allows a deeper level of security since the data can only be read by the sender and receiver. While it is being sent over, the information is encrypted and since you need both keys to be able to read the information, that also adds more secureness to the encryption. What’s different from symmetric encryption is that symmetric encryption uses the same key for both ends, the sender and receiver.

4. My Project

After doing all my research, I took what I have learned into making my own encryption algorithm and application. I took inspiration from 3 major aspects of my research; Caesar cipher, which was developed around 100BC, the Enigma machine which was developed during WWII, and symmetric encryption. I also took inspiration of the drawbacks of each aspect that I researched and used those to further improve my own algorithm. How my algorithm works is that it takes in a key and the message that the user wants to encrypt, cycles through each of the letters of the message and shifts the letter, just like the Caesar cipher, according to what the key has. After the message has

been shifted, I added one more level of the encryption by base64 encoding the message. This made it so that instead of the message having the same structure, it is broken down into binary then converted into ASCII characters. The decryption works just as the encryption, just in reverse.

4.1. Python, My Language of Choice

When I decided to write my algorithm, I chose to write in the language Python. Python is one of the most used programming languages that is used in today's age, and I wanted to use that language to build my project. I have never used Python before, so I had to teach myself how to write in this language, but what is Python? Python is a general purposed and is "an interpreted, object-oriented, high-level programming language with dynamic semantics" (Python). Python is very good at application development which is what I was exactly looking for since for my project, my goal was to create an algorithm and use it as an application for other uses. Another thing Python is good at is for databases and holding and reading data. The future goal for my project is to be able to store data, so Python was used to future proof just that.

4.2. The Algorithm

How my algorithm works is in three parts: generating the key, encrypting the message, decrypting the message. I first had to think on how I was going to break down the message. I have done this by running through the message string and index a new variable at each point. To further understand the code, a string is a set of letters or other characters, and a variable is a reserved memory location to store values.

```
    for char in message:
        if char.isupper(): #check if it's an uppercase character
            if i >= len(key):
                i = 0
            char_index = ord(char) - ord('A')
            # shift the current character by key positions
            char_shifted = (char_index + shiftKey[i]) % 26 + ord('A')
            char_new = chr(char_shifted)
            encrypted += char_new
            i += 1
        elif char.islower(): #check if its a lowecase character
            if i >= len(key):
                i = 0
            # subtract the unicode of 'a' to get index in [0-25) range
            char_index = ord(char) - ord('a')
            char_shifted = (char_index + shiftKey[i]) % 26 + ord('a')
            char_new = chr(char_shifted)
            encrypted += char_new
            i += 1
        elif char.isdigit():
```

```
            if i >= len(key):
                i = 0
            # if it's a number,shift its actual value
            char_new = (int(char) + shiftKey[i]) % 10
            encrypted += str(char_new)
            i += 1
        else:
            # if its neither alphabetical nor a number, just leave it like that
            encrypted += char
```

Breaking down my code, it starts out with a “for loop”, which is a way of iterating over any sequence. For this “for loop”, it starts by creating a variable called “char” which is a placeholder for a segment of the message that it is passing through. The first thing the “for loop” checks for is if that certain character in the message is capitalized. If that character is capitalized, then it applies the Caesar cipher by shifting that place in the alphabet by whatever the key states. If the character is not capitalized, then it goes to the next check in the “for loop”, which checks if it is lowercase. If it is, it does the same as it does in the first check. The next check is for checking if there is a number in the message, and if there is a number it gets shifted by whatever the value is. If the shift key is bigger than 10, it divides it and gets the remainder which means if it is 11, the shift value would be 1. The final check is if there are any special characters such as a period, exclamation point, or anything else. If there are any special characters, there is no shift in the character, and they just get added into the new encrypted message. Inside of each check, the key gets iterated through as well. The key is set up as an array, which is list of similar data types, and the index of the key is used as the shift key. If the key has reached the end of the length of the list, it resets the index number and starts from the beginning. How the encryption works within each of the checks is by first creating a temporary variable called char_index and set that to the current character in the message to the integer that corresponds to where the character is in the alphabet. Then it creates another temporary variable called char_shifted. This variable is the new shifted character, which is set by calculating the character index integer and the what the shift key index is, then by dividing everything by 26, which is for the 26 letters in the alphabet, and using that remainder. Then the final conversion is changing the integer into the corresponding character. So, with everything working together and following the algorithm, if the character we are changing is ‘a’ and the shift is going to be 5, first ‘a’ is converted to 1, then you add 1 and 5 to make it 6, then that gets reconverted to the corresponding character which is now ‘f’. Then the final step is to add the character to the new encrypted message. It is all the same through each check, just changes between if the character is an uppercase letter, lowercase letter, or if it is a number. Now that there is a way to encrypt the persons message, there needs to be a key that is generated

```
    self.secureness = secureness
    i = 0
    while i < self.secureness:
        x = random.randint(1,25)

        self.key.append(x)
        i += 1
```

```
    return self.key
```

The key is generated by taking in a value from the user which determines the length of the array. Once the length of the array has been determined, there is a “while loop”, which is another loop just like the “for loop” except it will keep looping while a certain statement is true or false. In this “while loop”, while the size of the array that the user has determined has yet to be met, a random number between 1 and 25 will be added to the array. The reason the random number stops at 25 is due to the fact there are only 26 letters in the alphabet. If the random number stops at 26, that would mean if the number 26 is added then there would be no shift since it would shift to the same character that was originally being encoded.

The final portion of my algorithm is the decryption part, and for the Caesar cipher to decrypt the encrypted message it just needs to de-shift. Because of that, the decryption code is just like the encryption code.

4.3. Creating the User Interface

My next step for my project is to make it accessible to the user with an interface. What I chose to do with my project is to make it reachable with a website. Since I was writing my project in Python, I decided to use Django as the framework for my website development. Django is a free and open-sourced framework for Python web development, and it allows rapid development of secure and maintainable websites. How Django works is that it follows a model–template–views architectural pattern, which means when the website is being developed it follows URL paths that is determined. The start of my web development began with focusing on the interface for encrypting and decrypting. For the encrypt page of my website, it allows the user to create their own key which will be used for the encryption algorithm, input their message and the key, and then output the encrypted message. For the decrypt page, it has the same elements as the encrypt except there is no key generation portion, just the encrypted message input, the key input, and then the decrypted message output. One problem I ran into when it came to integrating my Python code into the website, so I had to modify my original code.

```
def key(secure):
    key = []
    i = 0
    num = int(secure)
    while i < num:
        x = random.randint(1,25)
        key.append(str(x)+ ' ')
        i+=1
    return key
```

Starting with the key generation function, it takes in the same values as the original, but the output had to change. First thing that was modified was changing the values inside of the array from integers to strings. This is done by the “str()” function which inside of the parenthesis takes in a parameter, in this case the random number being generated, and converts the number into string. This is done because inside of html, which means HyperText Markup Language and is designed for web development, it doesn’t like displaying integers. The next modification was done for the array itself because when the array is being displayed, the numbers get all bunched up together and there is no differentiating between each number. A

space is added in front of each number to allow for this. The final modification was converting the input number from a string into an integer, because once again inside of html it has everything as a string.

```
def encryptMess(key, message):
    encrypted = ""
    i = 0

    keyInput = key.split(' ')
    shiftKey = list(map(int, keyInput))

    for char in message:
        if char.isupper(): #check if it's an uppercase character
            if i >= len(shiftKey):
                i = 0
            char_index = ord(char) - ord('A')
            # shift the current character by key positions
            char_shifted = (char_index + shiftKey[i]) % 26 + ord('A')
            char_new = chr(char_shifted)
            encrypted += char_new
            i += 1
        elif char.islower(): #check if its a lowecase character
            if i >= len(shiftKey):
                i = 0
            # subtract the unicode of 'a' to get index in [0-25) range
            char_index = ord(char) - ord('a')
            char_shifted = (char_index + shiftKey[i]) % 26 + ord('a')
            char_new = chr(char_shifted)
            encrypted += char_new
            i += 1
        elif char.isdigit():
            if i >= len(shiftKey):
                i = 0
            # if it's a number,shift its actual value
            char_new = (int(char) + shiftKey[i]) % 10
            encrypted += str(char_new)
            i += 1
        else:
            # if its neither alphabetical nor a number, just leave it like that
            encrypted += char
    messageBytes = encrypted.encode('ascii')
```

```
base64Bytes= base64.b64encode(messageBytes)
encrytpedMessage = base64Bytes.decode('ascii')
return encrytpedMessage
```

The next functions that needed to be modified were the encrypt and decrypt function, and since they do the same thing, I will be displaying the encrypt function as reference. The modification that needed to be done was the change to the key. First, when the modification was made to the key generation function to add spaces to the key when it displayed, those spaces needed to be removed. The next modification was then to convert the key into a new array of integers instead of strings. This is done because when the algorithm iterates through the key to shift the message within the cipher, it only accepts integers as an acceptable value. The final modification was to add another level of encryption is to add base64 encoding. This is done by converting the encrypted message into a bytes-like object using the string's encode method and store it in the variable called messageBytes. Then the base64 encodes messagBytes and stores it in the variable base64Bytes and using the base64.b64encode method, the string representation of the base64 conversion by decoding the base64Bytes as ASCII. All these modifications are the same for the decryption algorithm, except the base64 encoding is done in the beginning and that the message can be decoded from base64 and then run through the cipher.

The next step after modifying my original code is to make my website able to communicate with it. How that was done was with JavaScript and help with Django. First, I had to setup Django to make it so that my website can pull the information that is being requested. That is done with setting up special URL links that only the website can see, and inside of the URL link it calls in the certain information. After the Django portion gets set up, it was getting the JavaScript set up. The functions that were created obtain the information and values that the user has put in, and then after the algorithm runs, it outputs the values that are created. After all these modifications were finished, my website is fully up and running with my project.

4.4. The Security Behind My Project

After completing my research on the elements that I used for the creation of my algorithm, there were some security faults that I had to overcome for my final project. The first thing I had to change was with the Caesar cipher's fault, how the message structure stays the same and the shift only changes one letter. To first deal with the shift key fault with the cipher, I took inspiration from the Enigma machine. How I added another level of secureness by having the characters in the message get shifted by each number in the key. This is done by iterating through the key, and this adds a level of security because the only way to decrypt the message is by having the exact key. Another fault that I needed to fix was the message structure after the message gets encrypted. I fixed this by adding a base64 encoding to the message after the message is encrypted with the cipher. After doing that, it changes the message structure making it more secure by making it unreadable by the naked eye and web safe. Another level of security I added was to where my project is hosted, the website itself. Another way that there is a level of security is with the key generation itself. The key generation is all unique meaning the change of two keys being the same is highly improbable with every level of secureness of the key, or the length of the key. Since my project is hosted on AWS, Amazon Web Service, and therefore can be accessed by anyone. To add another level of security to that, I got an SSL, secure sockets layer, certificate to the website. What the SSL certificate does is encrypts the data that gets sent between the browser, the server, and the user. The final layer of security for my project is the project itself. Since everything is done locally, meaning nothing gets sent over any connection, there is no way that the message or key can be stolen. The only way someone can steal the unique key or message would be by user error.

5. Conclusion

Upon asking my research question about encryption, a lot of new information was presented in front me. First, cryptography has been around for thousands of years and was widely used by many civilizations. Cryptography has many forms and was important for many civilizations. With the scytale from the Spartans and the Caesar Cipher from the Romans, these were early predictors of encryption. With these early predictors, this allowed further advancements to what we call our more modern encryption methods such as the Enigma machine, and from that E2EE with the idea for further implementing keys. From all this research and taking inspiration of these devices and methods allowed me to create my own encryption algorithm. While I was creating my algorithm, I learned new things such as how to write in Python as well as web development. As I was creating my project, I had a lot of problems that needed fixing and barriers that I had to overcome. The final project can be found at project.jacoblevinsky.com.

Bibliography

Brown, & Rutkowski, J. L. (2021). Enigma Machines of WWII and Oral Implantology Research Papers. The Journal of Oral Implantology, 47(3), 181–182. https://doi.org/10.1563/aaid-joi-D-21-Editorial.47.3

Fuerst, N. (2021, November 11). *Who should we thank for modern cryptography? the Egyptian monks.Neal Fuerst*. Entrust Blog. Retrieved December 14, 2021, from https://www.entrust.com/blog/2020/12/who-should-we-thank-for-modern-cryptography-the-egyptian-monks/

Leighton. (1969). Secret Communication among the Greeks and Romans. Technology and Culture, 10(2), 139–154. https://doi.org/10.2307/3101474

Python. (2022). What is Python? Executive Summary. Retrieved 17 May 2022, from https://www.python.org/doc/essays/blurb

Santos, & Faure, A. (2018). Affordance is Power: Contradictions Between Communicational and Technical Dimensions of WhatsApp's End-to-End Encryption. Social Media + Society, 4(3), 205630511879587–. https://doi.org/10.1177/2056305118795876

Sidhpurwala, Huzaifa. "A Brief History of Cryptography." Red Hat Customer Portal, 19 Mar. 2019, https://access.redhat.com/blogs/766093/posts/1976023.

Spies, T. (2017, May 19). *Public key infrastructure*. Computer and Information Security Handbook (Third Edition). Retrieved December 14, 2021, from https://www.sciencedirect.com/science/article/pii/B978012803843700048X

Thales. "A Brief History of Encryption." Thales Group, 18 Apr. 2016, https://www.thalesgroup.com/en/markets/digital-identity-and-security/magazine/brief-history-encryption

Turner, D. M. (n.d.). *Summary of cryptographic algorithms - according to NIST*. Cryptomathic. Retrieved December 13, 2021, from https://www.cryptomathic.com/news-events/blog/summary-of-cryptographic-algorithms-according-to-nist