Visualization of Encryption

Vibha Pandurangi

5 December 2016

Abstract

Cryptography is the study of processing, storing, and transmitting data into a particular encrypted form that only those for whom the message is intended will be able to decode and read the message. While the first recorded use of encryption dates back to Ancient Rome with the Caesar cipher, encryption has developed far beyond simple alphabetic substitutions.

For my project, I studied cryptography and explored the visualization of different encryption methods through graphics using VPython. First, I learned about ancient cryptography, including the Caesar cipher method and the poly-alphabetic cipher. At this point, I then created a visualization of the Caesar cipher and the poly-alphabetic cipher by creating spinning wheels to display the encryption shifts. Then, delving further into more modern encryption methods, I studied public-key encryption in an attempt to visualize this as well.

Through my project in visualizing encryption, I hope to make learning and understanding the concepts behind various encryption methods less abstract and easier to comprehend.

Chapter 1

Introduction to Encryption

1.1 Encryption

Encryption is the process of converting data to an unrecognizable, hidden form in order to protect sensitive information. The goal of encryption is to convert the data into a form that only those for whom the message is sent can read the information.

1.2 Caesar Cipher

The Caesar cipher is a simple shift cipher used by Julius Caesar in 58 B.C. to protect sensitive military information from interception by enemy forces. This ancient method of encryption is a *mono-alphabetic substitution*, meaning the shift replacement is constant throughout the encryption process. Messages are encrypted through sliding the letters in a given message a set shift value down the alphabet. For example, if a shift value of 3 is chosen, the string “ABCDEF” would become “DEFGHI” with ‘A’ shifting down 3 letters to be substituted with ‘D’, ‘B’ with ‘E’, and so on.

In order to ensure that the message is encrypted and decrypted correctly, a shift value must be agreed on in advance. For Julius Caesar, this shift value was often 3. For example, if both the sending and receiving party, calling them Party A and Party B respectively, have both agreed on a shift value of 3, and the message “HELLO” is to be sent, Party A will send the message shifting each letter in the String by 3. By applying this shift to each character in “HELLO” gives the message “KHOOR”. When Party B

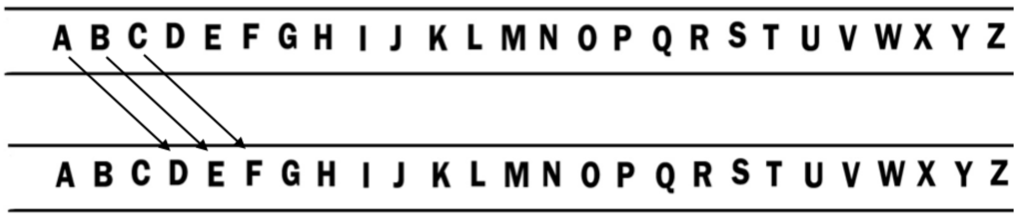

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

Figure 1.1: Shift value of 3

receives this message, they will simply have to apply a shift of -3 to attain the original message. If an arbitrary Party C were to intercept the message in the process of transmitting the message, it would read, "KHOOR", which is incomprehensible unless the secret shift value was known by Party C.

Despite the simplicity of the Caesar cipher, this encryption method was used by military leaders for hundreds of years after Caesar and was considered to be a strong method of encryption; the Caesar cipher was not broken until nearly 800 years later by Arab mathematician Al-Kindi[1]. He discovered the method of *frequency analysis*, scanning a text or book and recording the letter frequencies. The English language has a consistent pattern of letter frequencies, with the letter 'E' as the most frequently used letter (Figure 1.2).

Thus, to break the Caesar cipher, Party C can record the frequency of the letters that appear in the intercepted message and compare the frequencies of the letters in the encrypted message to that of the original frequencies. For example, if the most commonly used letter in the encrypted text was 'J', a shift value of 5 is likely because 'E' has the highest original frequency. The shift value can be found by comparing the letter frequencies of the encrypted message with that of the original set of frequencies, and thus, the encryption is broken.

1.3 Poly-alphabetic Cipher

The poly-alphabetic cipher, also known as the Vigenére cipher[2], uses multiple letter shifts, rather than only one like the Caesar cipher. Instead of choosing

[1]https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/caesar-cipher

[2]http://user.it.uu.se/ elenaf/Teaching/Krypto2003/vigenere.html

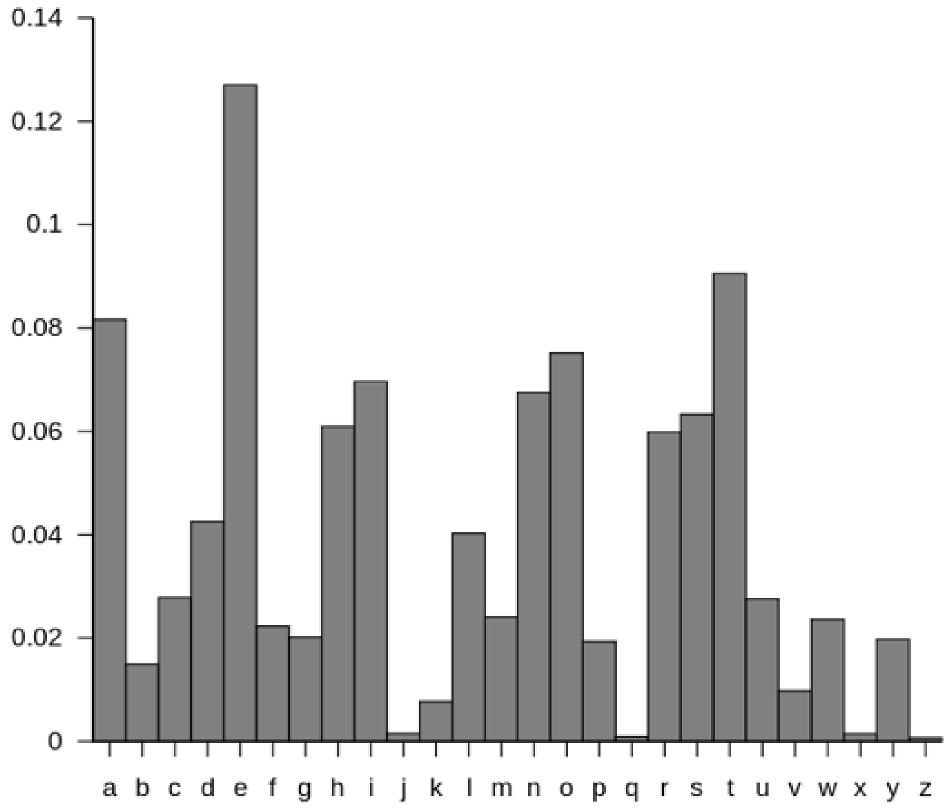


Figure 1.2: Graph of the letter frequencies of the English language

a shift value, a shift word is chosen. Each letter of the word corresponds to a number in the alphabet with ‘A’ starting at 0 to ‘Z’ at 25. For example, if the shift word is “CAT”, this would correspond to 2, 0, 19 in the alphabet. These numbers become the shift values for the message so if the message “HELLO” was to be sent, ‘H’ would be shifted 2, ‘E’ shifted 0, ‘L’ shifted 19, ‘L’ shifted 2, and so on. The end result of using the poly-alphabetic cipher on the word “HELLO” with a shift word of “CAT” is “JEENO”. Essentially, the poly-alphabetic cipher is multiple Caesar ciphers, whose shift value is determined by the corresponding letter in the key word. The poly-alphabetic cipher can also be depicted using a chart, like the one pictured (Figure 1.3). To use the chart, one must first find the letter of the message to be encrypted, then the corresponding letter of the shift value, and see where they match up on the table.

The key to breaking a poly-alphabetic cipher lies in the differential in partial frequencies in the letter frequency distribution[3]. In the case of the poly-alphabetic cipher, there is a code word, whose shifts are repeated throughout the message. This gives the leak of information that allows the poly-alphabetic cipher to be broken. The first key step to breaking a poly-alphabetic cipher is to determine the length of the key word. One must check the letter frequencies of the message at various intervals. The first frequency distribution graphed would be the entire message to see if the shift is simply a Caesar cipher. Otherwise, every second letter needs to be graphed by letter frequency and compared to the original graph, and so on until the frequency distribution of the interval approximately matches that of the original distribution. For example, if the shift word has a length of 3, when the frequency distribution of every third letter is checked, the length of word will be revealed because this will approximately match the original letter frequency graph. Once the length of the code word is found, all that remains is to break the simple mono-alphabetic ciphers individually. With a shift value of 3, 3 separate Caesar ciphers must be broken by the method described above.

The poly-alphabetic cipher is more secure than a mono-alphabetic cipher because the frequency distribution of the letters with the poly-alphabetic cipher is flatter. The goal of encryption is to get a frequency distribution graph that is uniform, meaning each encrypted letter has an equal chance of being any letter originally. With the poly-alphabetic cipher, the distribution is flatter because there are multiple shifts being used and thus, the original frequencies do not remain constant after encryption, unlike the Caesar cipher.

Because the key clue in breaking the poly-alphabetic cipher is the repetition of the shifts because of the repeating key word, a method of strengthening the poly-alphabetic cipher is to increase the length of the key word. By making the key word longer, there is less repetition, which decreases the differentials in the frequency distribution. In addition, a longer key would result in more intervals that are required to be checked to determine the length of the word, and thus, it is more time-consuming and tedious. All in all, using a longer shift word allows for a more secure poly-alphabetic cipher.

[3]https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/polyalphabetic-cipher

1.4 One-Time Pad, Randomness, and Perfect Secrecy

The *one-time pad* encryption is one that is purely based on randomness[4]. It is a long list of random shifts as long as the message that is being coded. For example, to encode the message "HELLO", five different random shifts would be selected. By choosing the shift completely randomly, there is a uniform frequency distribution and no frequencies differentials. The shifts do not have repetitive patterns because they are random, unlike the poly-alphabetic cipher which has a key word whose shift values are repeated throughout the message.

Thus, with the one-time pad, each encoded letter has an equal probability of being any letter in the alphabet originally, creating perfect secrecy. In order to decrypt the message, the receiving party must know the shifts applied so they can be reversed.

The problem with the one-time pad lies in that is it extremely impractical and difficult to exchange a completely randomized key before transmitting a message. The one-time pad is the goal for encryption because of its expansive number of combinations for encrypt a message. The Caesar cipher, to act as a comparison, has only 25 possible encryptions/shifts, which if necessary can be broken by brute force (meaning trying out each possible shift individually until message is decrypted). The one-time pad, however, has n^{26} combinations, with n representing the length of the message.

Perfect secrecy is the result of randomness, and as discussed, the problem with this lies in that long keys have to be shared in advance. This leads to the development of creating *pseudo-randomness*. One such example is the middle squares method. First, a random number is selected, which is called the seed. The seed is then squared, and the final output is the middle of the product. Then this output is squared, and the middle of this product is the next output, and so on.

The difference between true randomness and pseudo-randomness is that once the initial seed is established, there are many sequences that then cannot occur because the next "random" number depends on the previous "random" number, which depends on the seed. Only the seed is randomly selected, and the seed determines the rest of sequence in pseudo-randomness so once

[4]https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/one-time-pad

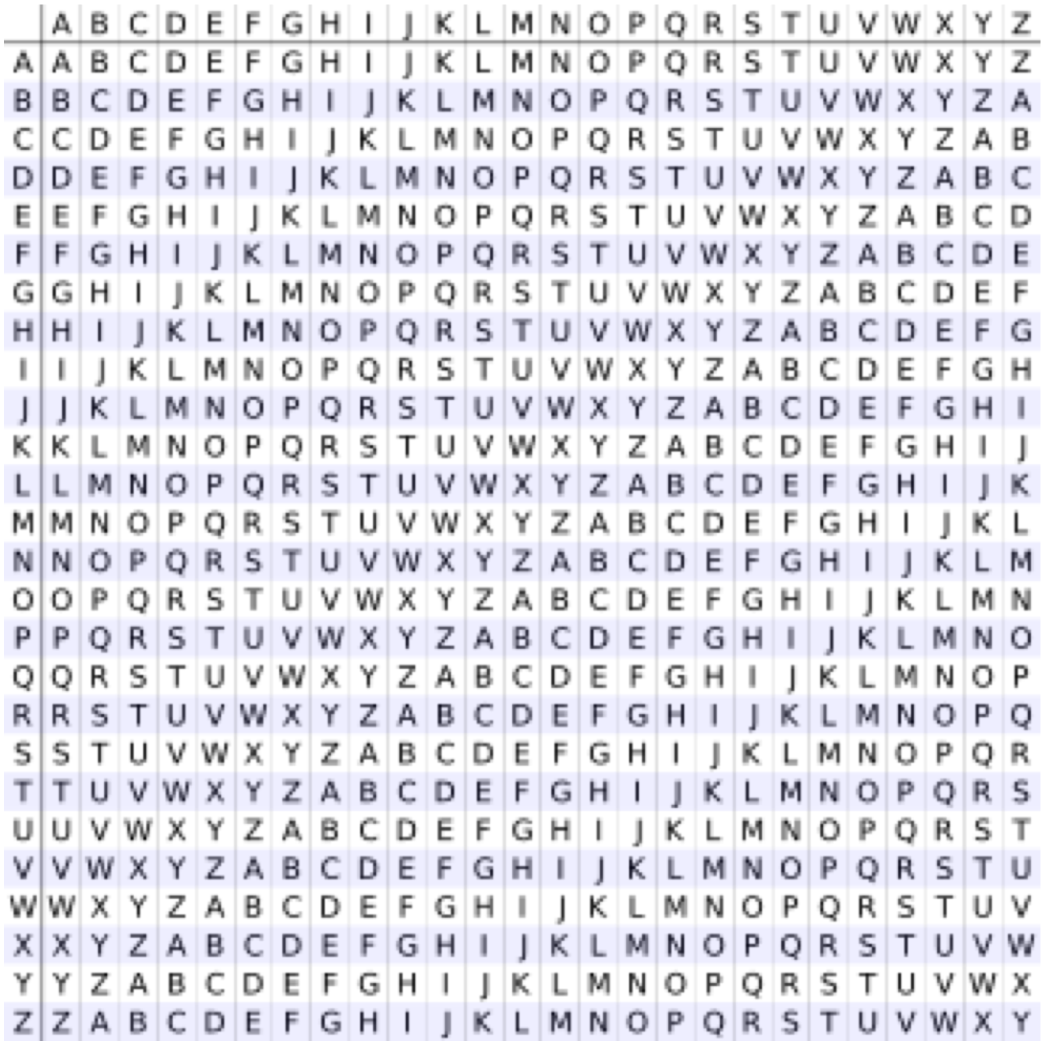
the seed is selected, the total possibilities, called the *key space*, is instantly smaller. Thus, to successfully utilize pseudo-randomness in encryption, the seed must be chosen such that it is not practical or time-efficient for a computer to search through all the seeds to find the one used in the encryption.

1.5 The Enigma Machine

The Enigma Machine is an encryption machine used by Germany in the second World War[5]. The goal of the Enigma Machine was to automate the one-time pad so that surprise attacks on the enemy could be planned and executed quickly and secretly. The ideal machine would accept the input, apply a random shift, and then output it. Because all machines perform specific, previously defined operations, the process from the initial state to the output is predictable; thus, perfect secrecy and pure randomness are not possible. Instead, the Germans had to produce multiple identical machines which would use long lists of shifts that would take too long to break. For this to work, the two machines would have to agree on an initial starting point, which is called the *key setting*, from which the machine would perform identical operations to get identical outputs. Thus, for the Enigma Machine, if the key setting was released, the encryption would be entirely compromised because the key setting determines the way the machine would process through the list of shifts. The Enigma Machine was a rotor encryption machine, that initially started with three rotors whose order could be rearranged. To increase the *key space*, the total number of key settings possible, a fourth rotor wheel was added to the machine. Because the key setting determines the remaining sequence for the machine to process through, the Enigma Machine's security was based on the size of the key space and the randomness of the initial state of the machine. If the key space was small, the enemy could process through each key setting and determine the shift more easily, and if the key setting had a pattern, the enemy could determine the pattern and predict the key setting. While the Enigma Machine was an attempt in mechanizing the one-time pad, it was not perfect. In fact, the design itself was one of the reasons the Enigma Machine's encryption was cracked. The machine was created so that a letter would never be encrypted to itself; however, to attain a uniform frequency distribution, each letter

[5]https://www.khanacademy.org/computing/computer-science/cryptography/crypt/v/case-study-ww2-encryption-machines

must be equally likely to be encrypted to any letter. Thus, by creating the Enigma to never code a letter to itself, the uniform frequency distribution was broken. In addition, the machine required the operators of the machines to randomly choose the initial position of the machine, but humans are unable to imitate true randomness. Humans tend to favor certain outcomes over others and deem some outcomes less likely than others but in random events, every outcome is equally likely. As a result, the uniform frequency distribution was disrupted even further. As a result, the Allies were able to create the Bombe, a machine that could check each rotor position and find the possible key settings, breaking the encryption of the Enigma Machine.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

Figure 1.3: Poly-alphabetic cipher table

Chapter 2

Public Key Encryption

2.1 Introduction

Public-key encryption, also known as asymmetric encryption is a modern-day method of encrypting data effectively because it does not require the two parties to have shared a key in advance as required by both the Caesar cipher and the poly-alphabetic cipher. It is asymmetric because the key used to encrypt the information is not the same as the key used to decrypt the message once it is received. In this type of encryption, there are private keys, which are kept secret between the transmitting party and the receiving party respectively, and a public key, which is not hidden to anyone.

2.2 One-way functions and prime numbers

Public-key encryption is based on *one-way functions*, which are functions that are easy to compute but are difficult to break down inversely. One such one-way function is prime numbers, and the basis for RSA encryption lies in multiplying prime numbers. All numbers are created from smaller primes, and all numbers can be broken down to their prime numbers, known as the *prime factorization* of a number. The fundamental theorem of arithmetic states that "every positive integer (except for the number 1) can be represented in exactly one way apartment from rearrangement as a product of one or more primes." [1] In other terms, each number has a unique prime factor-

[1]http://mathworld.wolfram.com/FundamentalTheoremofArithmetic.html

ization. For example, if someone is given two prime numbers, say 5 and 13, and is told to multiply them, it is easy to get the product of 60. However, if the person is given the product 60 and is told to determine those two prime numbers, it is far more difficult, exemplifying a one-way function.

2.3 Modular arithmetic

Modular arithmetic, also called clock arithmetic, deals with remainders. For example, $67 \bmod 9$ is the remainder of $67 \div 9$, which is 4. A key element of modular arithmetic is evident in looking at exponents. For example, in looking at 7^2 mod13, the answer is 10, and one would expect the answer to increase if the exponent were increased, but with modular arithmetic, this is not the case.[2] 7^3 mod13 results in 5, while 7^4 mod13 results in 9; the lack of order in seen in these results facilitates public key encryption.

2.4 Diffie-Hellman Key Exchange

The Diffie-Hellman Key Exchange allows the two parties to share messages without having had shared a key in advance using modular arithmetic. Calling the transmitting party, Party A, and the receiving party, Party B, the two parties agree publicly on prime number we will call P and a base number N. Then, Party A will choose a private number A, and Party B will choose a private number B, which are secret to both the public and the other party. Then Party A will perform the following operation:

$$J = N^A \bmod P$$

and Party B will perform the same operation with its private key, resulting in number $K = N^B \bmod P$. Then J is sent to Party B and K to Party A openly. Once J and K are swapped, Party A will perform the following operation:

$$S = K^A \bmod P$$

[2]https://www.math.cornell.edu/ mec/2003-2004/cryptography/diffiehellman/diffiehellman.html

where

$$K = N^B \bmod P$$

so

$$S = (N^B)^A \bmod P = N^{BA} \bmod P$$

to get a number S while Party B performs the following operation to get the number T :

$$T = J^B \bmod P$$

where

$$J = N^A \bmod P$$

so

$$T = (N^A)^B \bmod P = N^{AB} \bmod P$$

It can be seen here that the results of these operations performed by Parties A and B, S and T, are equal, and thus, Parties A and B now have a common, private key that can be used to communicate information without having shared one in advance.

2.5 Why This Works

When using large numbers for P, Q, and N, public key encryption is extremely difficult and time-consuming to break because of the sheer number of possibilities for P. In order for a Party C to intercept and decrypt the message, Party C would have to list the powers of N mod P because as seen previously, there is little pattern or order in computing modular arithmetic with powers.

Chapter 3

Visualizing Encryption Project

3.1 Proof-of-Concept and RTICA with Caesar Cipher

For a proof-of-concept of my project, I created a visualization of the Caesar cipher using two rotating wheels with letters. The inner wheels spins, aligning the outer wheel's letters with new letters on the inner wheel, displaying the shift. My RTICA allows the user to press a key to start the animation of a phase shift of three, which is a standard shift value for the Caesar cipher. The user is then able to use the shifted wheels to easily encrypt a message since the original letters are aligned with the ones shifted three letters over. To decrypt a message with a shift of three, the user must simply match up the encrypted letter on inner wheel letter to the letter on the outer wheel to decrypt it. If a key is pressed again, the inner wheel shifts back to its original position. This RTICA will allow the user to not only easily encrypt and decrypt messages with a shift value of three, but also allows the user to visualize the cipher.

3.2 Poly-alphabetic Visualization

For the poly-alphabetic cipher, I wanted to convey the idea that the poly-alphabetic cipher is simply multiple Caesar ciphers. Adding on to the Caesar cipher idea, this RTICA features multiple spinning wheels with various shifts with a default key word. The user presses a key to start the RTICA. Once a key is pressed, the four wheels spin to reveal the corresponding shift letters according to the shift values of the default word, “MATH”, which are displayed in the RTICA. This will allow the user to encrypt any message using the default key word since the shifted letters are shown aligning in correspondence with the letter in the original message. More importantly, it breaks down the poly-alphabetic cipher into the individual Caesar ciphers, simplifying the multiple shifts into a more comprehensible form.

3.3 Public-key Encryption Visualization

I found a clever way of visualizing one-way functions of public-key encryption when researching on the internet[1], which I then attempted to recreate and code in VPython. Mixing colors is an example of a one-way function that is easier to visualize than modular arithmetic. Mixing two given colors to produce a new color is easier than doing the inverse, easier than being given a color and being told to find its exact color components. Using this concept, I created a visualization of the concepts behind public key encryption, and displayed how a private key can be created without sharing a key or a shift secretly beforehand. The steps shown in my visualization of public key encryption are as follows:

1. Party A and Party B decide on public key (yellow)
2. Party A and Party B each choose their own secret private keys (red and blue respectively)
3. Party A mixes their private red with the public yellow to get orange
4. Party B mixes their private blue with the public yellow to get green
5. Party A and Party B now have orange (red+yellow) and green (blue+yellow) respectively
6. The two parties now publicly switch the orange and green, keeping the red and blue still private
7. Now, Party B has the orange (red+yellow), and Party A has the green (blue+yellow)

[1]https://www.khanacademy.org/computing/computer-science/cryptography/modern-crypt/v/diffie-hellman-key-exchange-part-1

8. Party A mixes their private red with the green (red+green = red+blue+yellow)
9. Party B mixes their private blue with the orange (blue+orange = blue+red+yellow)
10. The resulting colors from the two previous steps are the same, allowing parties A and B to use this color as their private key for communication

Thus, in the end, Parties A and B have the same end secret color, without having shared their private colors to each other. While the yellow, orange, and green in this example are private, it is extremely difficult to determine which exact colors and their shades composed the orange and green, allowing for the public-key encryption to be extremely secure.

Chapter 4

Future of Project and Further Developments

The purpose of my project is for the user to better understand how encryption works through a visual depiction of the different methods. When I was learning the Caesar cipher in my computer science class, I found it difficult to comprehend simply because I could not visualize it effectively. Often times, encryption is not understood because of how abstract it is in concept, and through visualizing simplified versions, it becomes easier to comprehend. Through my project, I look to making encryption easier to break down for the user.

The long-term purpose of my project in terms of the study into the public key encryption part of my project is to deepen my understanding of encryption, and using this knowledge, try and create a visualization or graphic to depict public-key and RSA encryption. Overall, I would like my project to help the user understand both ancient and RSA encryption, through text and visualizations. Currently, my project displays the concept of arriving at the private key, but does not encompass the entire process of encrypting the information, so the next step would be to find a way to visualize the encryption of the information.

In terms of further developments of my project, it would be interesting to see these animations written in JavaScript to allow them to become more user friendly and interactive. For the Caesar cipher RTICA, a development and improvement would be to allow the user to select a shift value themselves, rather than have a default of 3. For the poly-alphabetic cipher, an improvement would be along those same lines—to allow the user to be able to choose their own shift word, rather than being able to see the animation of the shifts of the default word used in the current RTICA. It would also be interesting to see an interactive table (Figure 1.3) that would allow the user to select two letters and see the paths leading to the encrypted letter. These improvements would be facilitated by transitioning to JavaScript, and recreating the current RTICAs in JavaScript.

In terms of the purpose of allowing the user to understand and see the concepts of various encryption methods, I believe this project was successful; however, there are many improvements and expansions that can be made in making the current RTICA's more user friendly and interactive in JavaScript as well as different visualizations.

Chapter 5

References

1. *Commentary on Modular Arithmetic*. University of Illinois. 20 February 2012. Web. 5 December 2016.

2. *Journey into cryptography*. Khan Academy. Web. 28 November 2016.

3. *Primes, Modular Arithmetic, and Public Key Cryptography*. Cornell University. 15 April 2004. Web. 5 December 2016.

4. *The Vigenere Cipher – A Polyalphabetic Cipher*. n.p, n.d., 1 December 2016.