Drake Sims

Hiram College

Computer Security IRC

The Evolution of Encryption

# Abstract

For centuries, cryptology has been used to disguise messages from one person to another. Throughout this time several methods have been used for encrypting ranging from the Caesar Cipher in 100 BC to modern encryption algorithms such as RSA. This paper will explore the topics of three different encryption and decryption algorithms, the Caesar cipher, the Enigma machine, and RSA. The purpose of this paper is to go over the history, the algorithms, and how these algorithms were used in the project associated with the paper.

# Introduction

Cryptography is defined as, “a method of protecting information and communications using codes, so that only those for whom the information is intended can read and process it.” (<https://www.techtarget.com/searchsecurity/definition/cryptography>). Cryptography has been around for several centuries dating all the way back to 1900 BC, when it was used by the Egyptians as a part of inscriptions in a tomb. However, within cryptography there is the process of encryption and decryption. Encryption is defined as, ““encryption is the process of encoding information. This process converts the original representation of the information, known as plaintext, into an alternative form known as ciphertext.” (https://en.wikipedia.org/wiki/Encryption). Once this ciphertext reaches its target, it must be decoded. This is where the process of decryption comes in. To decrypt, the target of the ciphertext must know or have a key to decrypt the ciphertext. This project aims to show how the Caesar cipher, the Enigma machine, and Rivest Shamir Adleman (RSA) are all used in the encryption and decryption process.

# Symmetric Versus Asymmetric

There are two types of encryption and decryption algorithms, symmetric and asymmetric. It is important to understand the difference between the two types of encryption processes. In a symmetric encryption, the same key is used for encrypting and decrypting. This, however, could cause issues in terms of protecting the information in the given message. If this key was somehow intercepted or got into the hands of someone else, they too would be able to read the encrypted message that was intended for a different target. An asymmetric encryption and decryption method, however, uses two separate keys to encrypt and decrypt a message. This gets rid of the need for trying to transfer the same key to the intended recipient, allowing for a safer way to encrypt and decrypt.

# The Project

The project is a text-based program written in Python that allows a user to choose if they would like to encrypt or decrypt a message. Upon selecting, the user can choose between three different algorithms, the Caesar cipher, the Enigma machine, or RSA. After selecting what algorithm the user would like to use, the user is prompted to enter their message, whether a message they would like to encrypt, or an already encrypted message that the user would like decrypted. A screenshot of a computer program

Description automatically generated

Figure 1 The Project Code for Selecting

# The Caesar Cipher

## History

The Caesar cipher dates back to roughly 100 BC and was used by Julius Caesar to send encoded messages to his army generals. Julius Caesar used the Caesar cipher to help prevent opposing militaries from being able to read his messages that would often include plans, and other writings that could jeopardize his militaries. If one of Caesar’s messages were intercepted while in route to his generals, typically opposing armies thought gibberish was written, and these plans were likely never known. Due to the fact that the Caesar cipher is a form of symmetric encryption the key to decrypt the ciphertext had to have traveled with the message to ensure the message could be decrypted. It is said that Caesar would typically use a shift value of three to encode his messages, however, there are rumors stating, “he would tattoo the key onto the shaved head of the messenger” and when their hair grew back, the key would not be visible” (Levinksy, Jacob).

## How the Caesar Cipher Works

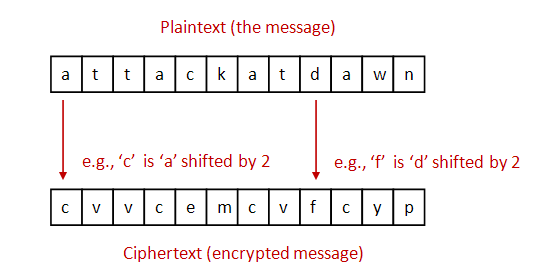
Although the Caesar cipher is seen as trivial today, at the time of its use, the cipher did a good job of encrypting due to the fact the cipher was not widely known. The Caesar cipher is a form of a substitution cipher which uses a fixed set of characters. A shift value is then applied to each character within the cipher, this shift value shifts the characters over however many characters the shift value called for and that character is now replaced. 

Figure 2 Example of Caesar Cipher

Figure 2 provides a visual example of how a shift in the Caesar cipher works. It is important to note that if the shift makes a character reach the end of the fixed set of characters, the characters then wrap back around to the beginning of the fixed set.

## Implementation in the Project

In the project, when a user selects that they would like to encrypt using the Caesar cipher, they are first prompted to enter the shift value they would like to use in their encryption. After providing a shift value, the program asks the user to input the message they would like encrypted. Lastly, the encrypted message is returned. In my program rather than the alphabet being the fixed set of characters, I have coded the use for spaces, upper and lowercase letters, numbers, and symbols. This differs from the original Caesar cipher, as typically their alphabet was all that was used.A screenshot of a computer screen

Description automatically generated

Figure 3 Caesar Cipher Encryption in the Program

After the user encrypts a message, they can then try decrypting their ciphertext by selecting to decrypt. The program then asks for the ciphertext and asks what shift value was used for encryption. Figure 4 shows the ciphertext being decrypted, and the original message being returned. A screenshot of a computer screen

Description automatically generated

Figure 4 Caesar Decryption

# The Enigma Machine

## History

The Enigma machine is, “an electro-mechanical rotor cipher machine which was first developed in Germany to provide security to commercial communication among companies and banks” (Prasad and Kumari). The Enigma machine was first patented in 1918 by a German businessman, however, in 1924, the German military began using the Enigma machine to encrypt messages. The Enigma machine continued to be used by the German military all the way up to and through World War 2. The Enigma machine became popular because it was known to be one the most secure ways to encrypt a message with over, “150,000,000,000,000,000,000 possible solutions” (<https://www.cia.gov/legacy/museum/artifact/enigma-machine/>). The Enigma machine would go through several evolutions to prevent the allies from cracking the machine. Eventually in 1941, the Enigma machine would be cracked by Alan Turing, an English cryptanalyst, allowing for thousands of encrypted messaged by Nazi Germany to be decrypted. This is seen as a huge aid to the allies’ victory over Nazi Germany in World War 2.

## How the Enigma Machine Worked

The Enigma machine featured several electrical and mechanical parts to make it work.

A machine with buttons and dials

Description automatically generated

Figure 5 an Enigma Machine

The Enigma machine shown in figure 5 is one of the later evolutions of the machine. To start, the Enigma machine typically featured three rotors and a reflector. A plugboard was a later evolution to the Enigma machine, which allowed for millions of new outcomes for the encrypted characters. The plug board worked by attaching pairs to the characters that would be switched when that character is pressed. For example, if a and z were a pair, if a user typed “a” the machine would encrypt as if the character was “z”. The rotors would have each character in the alphabet located on them. Each one of these characters had an electrical contact on them. When a letter was pressed to be encrypted the letter an electrical current would be sent, the plugboard would then change the letter, the electrical current was then sent to the first rotor which would rotate. After being sent through the first rotor the current would run through the next two rotors and hit the reflector. The reflector would then send the electrical current back through the rotors, further mixing the character up. After making it back through all three rotors, the ending letter would be sent to the plug board, and the other letter in the pair would light up on the lamp board. Messages would be encrypted character by character, depending on what light came on to represent the character on the lamp board. Figure 6 provides a great visual of the flow of the electrical current in the Enigma machine, and how the characters became encrypted.A diagram of a diagram of a lamp

Description automatically generated

Figure 6 Example of the Encryption Process in the Enigma Machine

However, because the Enigma Machine is a form of symmetric encryption, the German military would send out sheets with the settings of the rotors for each day of the month. This was useful because without the proper rotor settings for decryption, it was impossible to get the same message back, as characters could not be the same. This caused an issue, because if an ally was able to find one of these sheets naming the rotor settings each day, the allies were then able to decrypt of message that Nazi Germany would put out.

## Enigma Implementation in the Project

Due to the project being a text-based project that resets after running, the rotor settings had to be set to a constant to ensure messages could be encrypted and decrypted properly. In the project, users are able to type the message they would like encrypted, and it will return the encrypted message using the projects version of the Enigma machine. The projects version of the Enigma machine would be considered one of the early versions of the Enigma machine that did not use the plug board yet. However, the project enigma uses five rotors and a reflector for encryption rather than three rotors and a reflector. Figure 7 shows the output to the message “Hello” using the Enigma machine in the project. A screenshot of a computer screen

Description automatically generated

Figure 7 Enigma Encryption

The machine works properly, as the two “l” in the word hello become encrypted to two different letters. Upon decrypting the ciphertext the user will receive the original message.

A screenshot of a computer program

Description automatically generated

Figure 8 Enigma Decryption

# RSA

## History

RSA was first introduced in 1977 by three MIT colleagues named Ron Rivest, Adi Shamir, and Leonard Adleman, hence the name RSA. These colleagues aimed to tackle the issue of how keys can be transferred to the correct recipient without risk of being intercepted. This issue was first covered in a paper by Whitfield Diffie in 1975, however, Diffie was not able to come up with a function that met the requirements of the algorithm he came up with. This was until 1977 when Rivest, Shamir, and Adleman came up with the RSA algorithm. RSA is still commonly used throughout modern day web browsers, VPNs, and email, however, due to the resource consumption and the time it takes a computer to run the encryption method, it is common for the first message to be encrypted by RSA, and then switch to a different encryption method after keys are established.

## How RSA Works

RSA is a form of asymmetric encryption because it requires a public key and a private key. The public key in RSA is used to encrypt the data being sent by a user, while the private key is used by the recipient to decrypt the data that was encrypted by the public key. However, to generate the public and private keys, there is an algorithm that must be followed. To start, two prime numbers must be selected. In modern day RSA, these prime numbers are extremely large, ranging from about 1024 to 2048 bits. These prime numbers are then multiplied together to get your product. You then must calculate your totient by subtracting one from each of your prime numbers and multiplying those two numbers together. Now you can select your public key. Your public key must be prime, less than the totient, and must not be a factor of the totient. Now that your public key is satisfied, you can select your private key following these requirements. The private key does not have as many requirements, but it is more complicated. To select your private key, the key when multiplied by the public key and divided by the totient, must result in a remainder of one. After a public and private key are selected using the algorithm above, a message can now be encrypted by raising the message to the power of your public key and modulus by the product of your two prime numbers. To decrypt this message with the private key, you now take the ciphertext raised to the power of the private key and modulus by the product to get the original message. Figure 9 provides an example of what RSA looks like when being used.

A close-up of a computer screen

Description automatically generated

Figure 9 Example of RSA

## RSA Implementation in the Project

The RSA implementation in the project is a pretty simplistic version of RSA. When called for RSA, first the public and private keys are set using the prime numbers of 7 and 19. Due to the project being text based, these numbers were set as a constant to ensure the same outcome when encrypting and decrypting. When a user selects that they would like to encrypt using RSA, the user is prompted to enter the message they would like encrypted, and a list of integers is returned. In modern RSA these integers would be characters combined to look like the messages in Figure 9. Figure 10 shows the encryption and decryption process within the program. With more time it would be important to increase the size of the prime numbers, along with the encrypted message, in hopes of replicating RSA better. A screenshot of a computer

Description automatically generated

Figure 10 RSA in the Program

# Conclusion

## Future Work and Challenges

This project proved to be a challenging task but allowed for a ton of knowledge to be gained. With that being said, there were several challenges that were faced in the creation of the project. The Caesar cipher proved to be trivial, and there was not much issue faced regarding that. However, the Enigma machine proved to be a decently tough challenge due to the lack of understanding of the machine. If there was more time, I would like to truly build the World War 2 version of the Enigma machine. Due to the program being text-based, the rotors were not able to be coded correctly and had to be set to constants. Another thing that would be added with more time would be the plug board. Due to time constraints, I was not able to implement the plug board. The RSA implementation in the program is extremely simplistic, especially in regard to what is returned. With more time, the RSA program would return a string of characters rather than a list of integers, to better replicate RSA. Another thing that would be changed is the size of the prime numbers used for the public and private keys. The prime numbers used in the implementation are extremely small, and with more time would become more complex. Overall, with more time I would like to increase the complexity of the Enigma machine and the RSA implementation. Along with an increase in complexity, I would also like to better polish the code resulting in a better program. This involves making the program more than just text-based and creating classes for each algorithm.

## Final Thoughts

This project aimed to show the evolution of the encryption and decryption methods used in society. Overall, the project was a success as the program was able to encrypt and decrypt using the Caesar cipher, the Enigma machine, and RSA. Although the project could have been more complex and polished, everything worked, and showed an evolution in the algorithms used. This project allowed me to learn a lot about all sorts of encryption algorithms, but also taught me a lot of history. If I were to restart this project, I would choose the topic of encryption from the beginning of the term and focus on making the RSA and Enigma machine more complex.

# Works Cited

Calderbank, Michael. *The RSA Cryptosystem: History, Algorithm, Primes*, 20 Aug. 2007, math.uchicago.edu/~may/VIGRE/VIGRE2007/REUPapers/FINALAPP/Calderbank.pdf.

*Central Intelligence Agency*, Central Intelligence Agency, www.cia.gov/legacy/museum/artifact/enigma-machine/. Accessed 4 May 2024.

Ellis, Claire. *Exploring the Enigma - plus Maths*, Mar. 2005, plus.maths.org/issue34/features/ellis/2pdf/index.html/op.pdf.

“Encryption.” *Wikipedia*, Wikimedia Foundation, 8 May 2024, en.wikipedia.org/wiki/Encryption.

*The Enigma Cipher Machine and Breaking the Enigma Code*, www.mpoweruk.com/enigma.htm. Accessed 4 May 2024.

Kotas, William. “A Brief History of Cryptography.” *A Brief Hist A Brief History of Cryptography*, 5 May 2000, trace.tennessee.edu/cgi/viewcontent.cgi?article=1398&context=utk\_chanhonoproj.

Levinsky, Jacob. “Encryption: The History and Implementation.” *SUNY Open Access Repository (SOAR)*, Dr. Knarik Tunyan, May 2022, soar.suny.edu/handle/20.500.12648/12073.

“Module 14: Science & Engineering Applications III - Cryptography and Computational Biology.” *Introduction to Software Development*, www2.seas.gwu.edu/~simhaweb/cs1111/classwork/module14/module14.html. Accessed 9 May 2024.

Perendi, Daniel, and Prosanta Gope. *The Language’s Impact on the Enigma Machine*, 13 Nov. 2015, eprint.iacr.org/2021/1434.pdf.

Prasad, Kalika, and Munesh Kumari. “A Review on Mathematical Strength and Analysis of Enigma.” *arXiv.Org*, 17 Apr. 2020, arxiv.org/abs/2004.09982.

Richards, Kathleen. “What Is Cryptography? Definition from Searchsecurity.” *Security*, TechTarget, 13 Mar. 2022, www.techtarget.com/searchsecurity/definition/cryptography.

“What Is RSA Algorithm in Cryptography?: Simplilearn.” *Simplilearn.Com*, Simplilearn, 13 Feb. 2023, www.simplilearn.com/tutorials/cryptography-tutorial/rsa-algorithm.