CryPic: A Web App for Securely Disguising Messages as Pictures

Brendan Van Allen

Marist College, Poughkeepsie, NY

brendan.vanallen1@marist.edu

Abstract

The combination of cryptography and steganography has been studied and implemented in a variety of fashions. These two disciplines share many commonalities in principle, but take unique approaches to solving the problem of protecting information. CryPic is an application created to simplify the merger of these two philosophies, making it approachable to even the least technical of users. In this paper, the underlying themes of both cryptography and steganography are presented to establish the basis for CryPic, and explain how the application accomplishes the goal of bringing the two together to achieve stronger security than each discipline on its own.

Introduction

Cryptography and steganography are separate disciplines that have the same goal: to conceal information from third parties. Cryptography achieves this goal by encrypting the information, rendering it incomprehensible even if the information is transported in plain sight. Much of cryptography is grounded in mathematics and takes advantage of the inabilities of humans and our technology to efficiently solve mathematically complex algorithms. On the contrary, steganography relies on the fact that people and computers can be relatively naive.

The focus in steganography is not on protecting the contents of a message, but is on protecting the knowledge of there being a message at all [1].

Each of these disciplines have vulnerabilities that are not present in the other. In cryptography, encrypted messages are unreadable to the human eye, but this may make an important message stand out amongst plaintext messages. If a malicious third party comes into possession of a message that is encrypted with a relatively weak algorithm, or also obtains even part of the key for a strong algorithm, it could take as little as a few minutes to crack it due to the abundance of resources available for nearly all major encryption techniques [2]. Steganography does have its fair share of resources pertaining to it, but is not nearly as widely studied as cryptography. Furthermore, messages hidden using steganography are meant to be indistinguishable from other information. However, since the content of these messages is hidden in plain sight, it only takes one cunning individual to discover the use of steganography, and reverse engineer the data to reveal the hidden information [3].

The goal of creating CryPic was to combine cryptography with steganography in a way that is easily accessible, highlights the advantages of both disciplines, and uses each one to overcome some shortcomings of the other. Concealing the message as an image prevents it from drawing attention to itself, while the encryption protects the message even if it is able to be extracted from the image.

Related Work

The idea of combining cryptography and steganography is not novel, and is the subject of a number of publications (see references). The majority of the previous body of work in this area is aimed at large-scale implementations that protect highly sensitive information. An example of this type of implementation is given by Atee, Hayfaa, and Robiah in [4]. With CryPic, I wanted to take a simpler approach that targets normal communications amongst regular people. The algorithm proposed by Sharma and Kumar in [5] was nearest to the complexity that I sought. While this application is not intended to protect critical information such as nuclear launch codes, it does provide an easily accessible way for anyone who wishes to add an additional layer of security other than cryptography or steganography alone.

Methodology

The process CryPic uses to combine cryptography with steganography is not actually a combination, but rather uses each approach in sequence. The high-level view of the sequence is as follows:

1. The user inputs their message, and either inputs a key, or chooses to have a random key generated
2. The key is then used to encrypt the message using AES to produce ciphertext
3. Each character of the ciphertext is converted to its ASCII value, and this value is inserted into a triple that represents an RGB pixel value
4. The resulting array of pixel values is used to create a new image

The first two steps of this process represent the encryption phase, and the output of this phase (ciphertext), is fed into the steganography phase to further reduce the risk of a third-party being able to access the message. In order for the intended recipient to view the message, CryPic simply reverses the process:

1. The recipient inputs the key given to them by the sender, and uploads the image
2. The pixel values of the image are put into an array that is processed to extract the relevant ASCII values
3. The resulting list of values is converted to an ASCII string that represents the ciphertext
4. The key is used to decrypt the ciphertext and the plaintext message is presented to the recipient

This layered approach taken by CryPic provides a level of security that is higher than that of either cryptography or steganography alone.

Application

The actual functionality of CryPic is presented as a web application. This application was created using the Python Flask microframework to create a simple, local web server, on which python scripts can run that pass information to and from HTML web pages.

Pycrypto is the module used to provide encryption and decryption using AES in the cipher block chaining (CBC) operation mode. CBC was chosen as the mode of operation in order to allow messages larger than 128 bits to be encrypted by the application. The current version of the application supports AES with a key size of 128 bits, but the user may input any key that has a length greater than 0. The reason that this is allowed is because the key specified by the user is not directly used as the key for encrypting using AES. Rather, the key given by the user is put into a hash function that always produces 128-bit blocks, and the output of the hash function is then used as the key given to AES. There are two primary reasons behind this design choice. The first is due to vulnerabilities that exist in pycrypto. Hashing the key before passing it to the pycrypto module helps protect against these vulnerabilities. The second reason is that I wanted to make this application as easy to use as possible for users of all experience levels. Some users will not be familiar with the mechanisms behind AES, and even more users won't be familiar with encryption at all. Removing any key length requirements allows these less experienced users to use the application while thinking of the key as a simple password, and providing a wrapper of sorts around the interaction between the user and the encryption module.

After the message is encrypted in the application, the steganographic phase begins. As stated previously, the technique used to encode the string of ciphertext as pixels is based on a simplified version of the algorithm proposed by Sharma and Kumar in [5]. Essentially, each character of ciphertext is converted to its decimal ASCII value, and this value is placed into a triple that represents an RGB pixel. For my implementation, random values are generated for the red and green pixel values, and the ASCII value of the character is used as the blue value. After processing the entire string of ciphertext, the resulting array of pixels is passed to the Pillow module (Python Imaging Library), and a PNG image is created.

The decryption process of the application is fairly straightforward: the steps of the steganographic phase are reversed to reproduce the ciphertext string, which is then passed to the pycrypto module for decryption. The Pillow module is used in the decryption process to provide the ability to pull the individual pixel values from the image.

Results

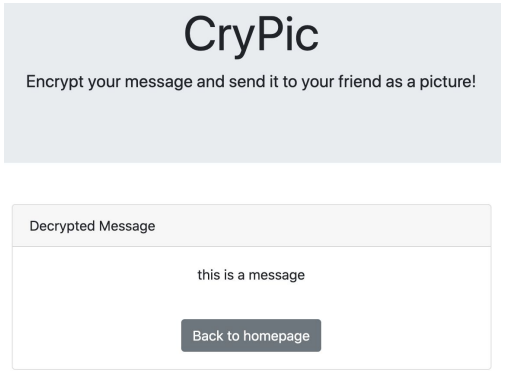
The images produced by CryPic are relatively unique every time. Due to the randomness of the red and green values, any two pixels that store the same ASCII character value as the

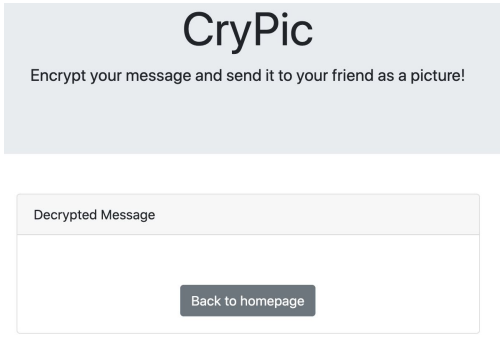
blue value have just a $1/256^2$ chance of also having the same red and green values. Scaling this probability out to represent the chances of any two images that store the same ciphertext string (this would mean the same message is encrypted with the same key on two separate passes through the code) having the exact same RGB values for all of their pixels is as simple as multiplying the previous probability by the length of the ciphertext string. For a ciphertext string with 64 characters, the chances of the exact same image being reproduced on separate passes through the program is just 1 in over 4 million. A simple proof of this notion is given by the two images below:

In both images, the key used is “this is my key123”, and the message given is “This is a test of a long string that is at least 50 characters”. As you can tell, the pixel values that make up each image are unique.

In a typical encryption system, if an attempt is made to decrypt a cipher with an invalid key, the output of the decryption is not null and is not equivalent to the plaintext message that was originally encrypted. In a system of this kind that encrypts character strings, the output of decryption using an invalid key is a character string that is insignificant in meaning. The following example shows how the same situation is handled in CryPic:

The image was created from encrypting the message, “this is a message”, with key, “testkey”.





The image on the left shows the application when the image from above is decrypted using the correct key. The image on the right shows the application when the image is decrypted with an incorrect key. For this example, the incorrect key given was “testkey1”.

The decision of how CryPic handles erroneous key input for decryption is based on the principles laid out in [6]. When this situation occurs, the output of the decryption is null. This is intended to withhold any information about the underlying mechanisms behind CryPic from a malicious third-party. During my design of the application, I considered the alternative option of displaying a message if an invalid key was given, but decided to let the decryption fail silently, in hopes of possibly discouraging the attacker from repeatedly attempting to guess the key.

Conclusion

The closely associated, yet distinct disciplines of cryptography and steganography set out to achieve the same basic goal: protect information from third-parties. Each discipline takes its own approach to accomplishing this goal, and both can be quite successful at doing so. These disciplines also have their own unique limitations and vulnerabilities. When creating CryPic, I wanted to build an application that synergized the security aspects of both cryptography and steganography, while also using the strengths of each one to overcome some drawbacks of the other. The final product reflects this design goal, while also keeping the application intuitive enough for users of all experience levels to use. The level of security provided by CryPic surpasses the levels that either discipline can achieve alone, and that is the defining aspect of my project.

References

[1] T. Morkel, J.H.P. Eloff and M.S. Olivier, "An Overview of Image Steganography," in *Proceedings of the Fifth Annual Information Security South Africa Conference, ISSA2005, Sandton, South Africa, June/July 2005* (Published electronically)

[2] A. Al-Shaaby and T. Al-Kharobi, “Cryptography and Steganography: New Approach,” *Transactions on Networks and Communications, 2007*.

[3] H. Wang and S. Wang, “Cyber Warfare: Steganography vs. Steganalysis”, *Communications of the ACM,* vol. 47, no. 1, p. 76-82, January 2004. [Online]. Available: ResearchGate, http://researchgate.net. [Accessed April 29, 2020].

[4] H. Atee and R. Ahmad, “Combining Cryptography and Steganography for Data Hiding in Images,” *Proceeding of the 13th International Conference on Applied Computer and Applied Computational Science, ACACOS2014, Kuala Lumpur, Malaysia, April 2014.*

[5] V. Sharma and S. Kumar, “A New Approach to Hide Text in Images Using Steganography,” *International Journal of Advanced Research in Computer Science*, vol. 3, no. 4, p. 701-708, April 2013. [Online]. Available: ResearchGate, http://researchgate.net. [Accessed April 2, 2020].

[6] F. Piper, "Basic principles of cryptography," *IEE Colloquium on Public Uses of Cryptography*, *London, UK*, *May 1996*.