A tool that encrypts and hides data in PNG files

Sunday 6th January, 2019 - 22:45

Justin VINANDY
University of Luxembourg
Email: vinandy.justin@gmail.com

Benot-Michel COGLIATI
University of Luxembourg
Email: benoit.cogliati@uni.lu

***Abstract*—This BSP will first give an overview of the project related subjects, namely cryptography and steganography. Then the objectives and limitations of the project will be defined. In Background I will state that which I had to understand or learn before I could properly begin the project. The I will, in detail, explain the scientific and technical parts of my project. And finaly I will give my personal assessment to the whole BSP.**

1. Introduction

This is the bachelor semester project paper written by the first year bachelor in Computer Science student Justin VINANDY, who was aided and supervised by the tutor Benot-Michel COGLIATI.

The goal of this project is to build a tool that, given a password, a file, and a PNG picture, outputs a new picture that is similar to the first one, and also contains the encrypted bits of the file. The tool should also be able to extract the file when given the correct password. This work involves basic steganography and applied symmetric cryptography.

2. Project description

2.1. Domain

The project has mainly 2 scientific domains and a technical domain.
The scientific domains include:

2.1.1. Stenography:. The art of hiding information in another piece of information (such as a file) so that even if someone receives the file they wont suspect hidden information in it. In this project the domain/type of stenography used will be that of hiding a message inside of a png file by manipulating the individual pixels of the png to contain the information of the message without the image looking tampered with (at least not by the naked eye).

2.1.2. Cryptography:. The art of encrypting information, scrambling the data in a way such that, if someone receives that file and knows that there is information there they cant decipher it (at least not easily). In this project it is the message that will be hidden that should be encrypted.

The technical domain is:

2.1.3. Python:. Python is the programming language used to write the program as it offeres a large collection of libraries to aid in this project as well as the fact that I was currently learning this language during the duration of this semester.

2.2. Objectives

The objective of this project is to create a program that when given a PNG picture, message and a password, gives back a special PNG picture that looks (almost) identically to the first picture but has the message hidden and encrypted inside of it.
The way it will work is that the program will encrypt the text message by using the given password and turning the password into an encryption key that gets used in the encryption function. The then encrypted message will be turned into binary. Then the PNG file will be opened using another python library, showing the binary value of each individual pixel of the picture. Then the encrypted binary text will be used to replace the last binary value of each pixel of the picture. So, by the end I will receive a picture that looks almost identical to the first one but has the encrypted message hidden in it, in each individual, slightly altered pixel. It is most important for the project that the information hidden is securely encrypted an that the produced image looks identical to the first one to the naked eye.
Then the program should also be able to take a PNG picture and a password and then be able to give you back the message if one has been hidden in the picture beforehand and the given password is equivalent to the one used to create that special picture.
The way this will work is that the program will receive the PNG picture and ask for the password. The program will then get the information from each relevant pixel of the image by reading the last binary value of them. Then it will try to decrypt the information using the key derived from the given password. Now if it turns out that either the password was wrong or that the picture didnt have a

message encrypted in it to begin with (or it was corrupted or tampered in some way), then the program wont return some garbage/random message but instead just an error. In the end if no problem arises then the program will output the message that was in the picture.

2.3. Constraints

For this project there were no constraints considered, as in there were no properties or requirements that had to be fulfilled before having started the project, since this is, after all, the first semester project. The only things discussed before the project really began were certain limitations, as in some things that were either impossible, unreasonable or very difficult to do for the project.
Those limitation consisted of me not having to write any encrypting or decrypting program myself for the project as doing so in my level would take too much time and could be considered a project in of its own. So instead I would be using a python library that offers these tools from GitHub.
Another limitation is that image editing, an important part of my project, would also be done by a library, again because trying to edit a picture and writing the code for that by myself would be unnecessary and needlessly difficult if I could simply find something on GitHub, a website that offers python libraries for users to share and use.

3. Background

Since there were no constraints before the project, meaning no prior knowledge needed before starting the project, there is also no background needed for the project. All subjects and domains related to the project, such as stenography and cryptography, were to be studied and understood during the project and the explanations and reports of those subjects will be found in section 4.

The only thing one might consider as prior knowledge needed, is knowledge in some programming language. But since Im already learning python during the semester that requirement will also be fulfilled after the project has started and isnt considered as a required background.

In the next part I will briefly explain the scientific aspects or the project 4.

3.1. Scientific

Steganography:
Steganography, in the domain of informatics, is the practice of hiding any sort of file or message within another file or message with the intent being to conceal the contents of the first file and to conceal the fact that there is anything hidden at all in the second. It comes under the assumption that if the information is visible then the point of attack is evident so the goal is to hide the very existence of the embedded data. The benefit of steganography compared to cryptography alone is that steganography doesnt look suspicious at first glance unlike when one sees an encrypted message, while one might not be able to decipher it, still reveals the fact that there is something to inspect. In the past physical examples of steganography include things such as writing secret messages in semi invisible inks or having the first letter of each sentence make a message. Modern steganography came with the inception of the computer. Examples of modern steganography would be to hide a message in a frame of a video or, like in this projects case, concealing messages in the lowest bits of noisy images. The type steganography used in this project specifically is digital image steganography. What it is, how it works, the exact method used as well as its functionalities will be discussed further in the report as it is part of the scientific deliverable.

Cryptography:
The second scientific aspect will be the cryptography, the art of securing/encrypting a message so that even if it is seen by others, they wont be able to (easily) figure out/decrypt the message. In this case the specific type of cryptography I will be using is AEAD or Authenticated encryption with associated data, a type of encryption that provides confidentiality, integrity, and authenticity assurances on the data given. Cryptography has been used in the past, as an example, as a way to saftly relay messages via radio during the second world war. While these messages could be picked up by the enemy, they couldn't do anything with them as they were encrypted. Only the allies that had the same key could decipher the messages. Now in the digital age, cryptography is wildly used everywhere to secure almost any data shared online, in order to prevent third party onlookers from stealing and abusing that information, as well as making sure that the information shared hasnt been tampered with.

3.2. Technical

In the technical area, the program used for my project is Python and various downloaded python libraries that will provide me with the necessary functions to construct the program. The reason I used python is because it is currently the programming language, I was learning during the winter semester and because it has a simple and intuitive layout and easy access to many libraries.

4. A tool that encrypts and hides data in PNG files

4.1. Requirements

The project had several requirements to be satisfied in order to be considered completed. These requirements can be split into two parts: the requirements needed to comprehend how to make the program successfully, and thus create a report and understanding of the scientific part of the BSP and the requirements that the program itself had to successfully fulfil, meaning that the program could do what it was required to do.

First the student will have to research on two subjects:

Cryptography, what it means, what its used for and how it is applied in this program. Then he will have to decide on an encryption method/type that is secure and will satisfy the objectives and requirements of the program and of course implementing it correctly.
Steganography, more specifically digital image steganography and what method is used to hide information in a PNG picture.

Then the program itself has several requirements:

First the program should be able to do both the encryption process and the decryption process (while of course also hiding the message in a PNG file), meaning that all functionalities needed are in one program. The user should then be able to choose if they want to hide a message or decrypt a message from a picture. That should be done simply by a question prompt that the user will answer and depending on what the user wants; the program will then move on to the desired functionality. The program will be split into two parts: the encryption part and the decryption part.

For the encryption part it is required that the user can input a message that will be encrypted, a password that will be used for said encryption and a PNG picture that will then have the encrypted file hidden within it. It is important that the encryption is strong and has integrity, authenticity and confidentiality built into it. The exact meaning of those requirements will be explained in Design. Then the program will encrypt the given message using the password in some way and then hide the encrypted data inside of the picture. Finally, the program will save a new picture under a different name that will have the data hidden inside of it. It is required that the outputted picture looks identical to the initial picture given, at least so that a human wouldnt be able to tell the difference.

Then for the decryption part the user will have to input a picture (that the user thinks has an encrypted message hidden in it) and also the same password that was used to encrypt the message as is required to decrypt it. Then only if the picture had a message hidden in it and if the given password was correct, then the program will output the message for the user to read. If the picture didnt have a message hidden in it or if the password was incorrect the program will then give an error message instead of trying to decrypt garbage data and output a faulty, nonsensical message.

4.2. Design

In this part I will explain the scientific concepts that had to be learned during the project as well as explain the process that was undergone to find out how to satisfy the objectives and requirements. The tools and libraries used for the program will also be explained here.

Cryptography:

The first part was to decide what type of cryptography is needed for the project since there are several types, the two main types being Symmetric-key cryptography and public-key cryptography. Briefly explained: Symmetric-key cryptography has the same key being used for the encryption and decryption while public-key (or asymmetric key) cryptography has different keys being used for encryption and decryption. For this project the user hiding the message and the person receiving the message should have the same password/key to encrypt and decrypt, so Symmetric-key cryptography is the one thats relevant.

AES (Advanced Encryption Standard)

AES is a very popular worldwide used encryption method that is a subset of the block cipher. A block cipher is a deterministic algorithm that operates on a fixed length set of bits, called blocks, where the transformation is specified by a symmetric key. It is used in block cipher modes of operation, an algorithm that uses block cipher, to provide services such as confidentiality and authenticity. The block cipher by itself is only useful for transforming a block, while the mode of operation describes how to repeatedly apply the block cipher operation. The exact details of how AES works was not necessary for the project but the different modes one could use with AES were important to understand as they provide the integrity,confidentiality and authenticity needed for the project. Those different modes include:

Electronic Codebook (ECB)

ECB is a very simple mode for encryption where the message is divided into its blocks and then encrypted separately. The problem with that though is that, since the block cipher will always transform a block in the same way. So, if the blocks are the same at the beginning then after the encryption they will be encrypted but both blocks will still be the same. This makes it susceptible to attacks since each block gets decrypted in exactly the same way as well. So, this mode is not suitable for the program.

Authenticated encryption (AE) and authenticated encryption with associated data (AEAD)

While most modes of encryption do offer confidentiality and authenticity, they do not immediately offer integrity, meaning they need to be combined with a hashing algorithm, and they need IVs (initialization vectors) to randomize the encryption. But AEAD provides confidentiality, authenticity and integrity to the data that needs to be encrypted all under a single easy to use programming interface. It also allows a header to be added that has integrity and authenticity but no confidentiality, but this wasnt necessary for the project.

Integrity is used to make sure that nobody changed parts of the data during the transfer between two parties. Normally to do so a HMAC (hashed message authentication code) is used for this. A HMAC is just an algorithm that used some initial value and a passphrase that both parties share.

Confidentiality is used to make sure that nobody was able to read that data that was sent in between parties. To achieve this encryption algorithms are used. In this case, Symmetric algorithms allow encryption and decryption with the same key.

Authenticity is used to make sure that the message has indeed been sent by the correct party. Normally pre-shared keys/passwords are used for this.

The way the AEAD programming interface works is for: Encryption Input: plaintext, key, and optionally a header in plaintext that will not be encrypted, but will be covered by authenticity protection. Output: ciphertext and authentication tag (message authentication code).

Decryption Input: ciphertext, key, authentication tag, and optionally a header. Output: plaintext, or an error if the authentication tag does not match the supplied ciphertext or header.

Since AEAD satisfies the requirements and is fairly simple to use, it is the one I chose for the project.

On Github I found a python library simply called cryptography that has the AEAD method in it. In the library I specifically chose to use the Galois/Counter Mode (GCM) variant of AEAD since it is a widely adopted and efficient symmetric mode of operation. I also used the librarys PBKDF2 (Password Based Key Derivation Function 2) function that is able to derive a key from a password.

Steganography: The type of steganography that will be used for the project is digital image steganography. It takes advantage of the fact that images, digital pictures, contain a lot of data to hide messages in and that images by themselves are also inconspicuous, as in there is nothing suspicious if one shares a picture with another.

For PNG files, each pixel uses 9 bits to represent each of the colour values and alpha(transparency) values. Meaning that the blue alone has 2 to the power of 9 different levels of blue intensity. The difference between 111111111 and 111111110 in the value for blue intensity is likely to be undetectable by the human eye. Therefore, the least significant bit can be used more or less undetectably for something else other than color information. If that is repeated for the green, red and alpha elements of each pixel as well, then it will be possible to encode one byte of data in 2 pixels.

The reason PNG pictures will be used for the project is that the pixels of PNGs are fairly easy acces, PNGs dont get easily compressed and degrade like JPGs and on average PNGs have more data then JPGs.

In order to open images in python I found an image editing library on github named Pillow that uses an older image library PIL (python Image library). I use Pillow in order to open images and go through each pixel of the image and read its red, green, blue and alpha values.

Now these pixel values are read as integers, and the cipher text that the encryption function produces is a string of bytes. I will need something that can convert both into a mutable array of bits, that can then be turned into a binary string. While it is technically possible to do so in python it is unnecessarily complicated which is why I found another library called BitString that allowes me to turn everything into a BitArray and then convert that into its binary form and then back to its integer or byte string form. Using those three libraries and now understanding the scientific aspect, I wrote the program that is the technical deliverable.

4.3. Production

In this part I will explain in detail the final result of the technical deliverable, being the program itself. I will explain the way the program is structured and how it works as well as the different variations the program went through during the process of programming and refining it, in order to compare versions and explain certain alterations and aspects of the final program.

The technical deliverable produced is a singular python program capable of two things, and thus the description of the program will be split into two parts.

At first though, the user will be asked by a simple question prompt if they want to do the decryption part; meaning decrypting the file that they put in with their password and then having that data be hidden in the PNG file that they chose; or if they want to do the decryption part; meaning the user gives a PNG picture and inputs a password to receive the hidden file that was in the picture. Depending on the users answer (which has been abbreviated to either be E for encryption or D for decryption for ease), the program will go to the relevant part. If the user gives a nonsensical answer the question will simply loop. At the end of each part the program also asks the user if they want to continue using the program, in which if the user answers with Y (for yes) then the program will jump back to the beginning, if the user answers N (for No) then the program will end. This whole process is simply done with a few if else functions and a while function that checks if the user does want to continue.

First part The encryption:

When the user has chosen this option, the program will first create the variables needed for the encryption, those variables being the backend,salt,aad,nonce and the kdf hash.sha256 algorithm. The backend will be the default one as there is no point or benefit in changing it, the salt will also be fixed to a certain bitstring. The salt has to be fixed because if it is different for the encryption process and the decryption process then the whole decryption part wont work since the variables used for both are not the same. The nonce is a random 12 byte long string that will then be turned to a mutable BitArray. The nonce value is random so that the encryption has more security built into it (since always reusing the same nonce isnt recommended, security wise) and it is turned into a BitArray in order to also be hidden in the picture. The nonce value itself doesnt need to be encrypted though, since it is a random string after all. The aad (authenticated but unencrypted data) variable will kept blank as there is no use or need for it in this program. Then the program will ask the user to input a message in form of a file, meaning that the user will have to type in the file name and type (an example would be mymessage.txt). The program saves the last three characters of the string in order to later include the file type in the data that will be hidden in the picture. The file will then also to read by the program in terms of bytes. The kdf algorithm is prepared in order to turn the password that the user will input into a secure key, that will later be used for the encryption. Then

the user will be asked to input a password. The password itself can be a string of any size, even left blank if the user wants to. Since the password will be turned into a suitable key through the kdf hashing function it will still be secure (even though it is still better to have a longer password as the security will be stronger then). The created key is then used with the file data, add (which has been left blank) and the nonce string, in order to produce an encrypted cyphertext in byte form. This cyphertext is then converted into a bitArray and that bitArray can then be converted into a binary string of 1s and 0s.

The length of this string will then be combined with the file type in another variable typ. The purpose of this is to be able to tell the program afterwards, in the decryption phase, how mmuch it should look for in the picture as well as what type of file the read data should be. If the data length werent specified then the program wouldnt know how long the intended message data is and would instead scan through the entire picture which would require a long amount of time and would make the scanned data faulty, as in the data scanned cant be decrypted by any password. Since the length of the data binary string can vary a lot the length of the created variable would also vary, which would create a problem in the decryption phase as the program wouldnt know how much of the scanned data relates to the message length and how much doesnt. So, the length of the typ variable gets lengthened to be 128 bits long, a size big enough to be sure that no data length will be bigger. It the data length were bigger, then the whole program wouldnt work in the first place as the size of the message file would be bigger than any picture available to hide the information in (and even then it would take a very long time to put all that data in the picture). Now that the typ data has a fixed length it can also be encrypted using the same key from before, but a different also random nonce(that also gets saved for later). The typ data should be encrypted because it does reveal important information about the message, namely what type of message is hidden as well as in how many pixels that message is hidden in the picture. So for security reasons it gets encrypted as well.

Now that the encryption phase is over the program moves on to the steganography phase. The user will be asked to type in what image should be used to hide the information in. It is important that the picture selected looks fairly busy and that the PNG picture is large enough. The picture should look busy in order to mask the fact, that the pixels have been altered, better. The picture also has to be large enough so that, in case the message file is somewhat big (around 100 kilobytes, what you can get if you want to hide another picture in a picture or a small audio file), the picture is large enough to fit all the data in its pixels. If the picture isnt big enough then all the data wont be able to be put in the picture which will make the decryption process impossible (since the encrypted message isnt the same anymore). The image the user selected then gets oped by the program using the pillow image library. Using a couple of for loops the it goes through the picture pixel by pixel, getting the red, green,blue and alpha values of ech pixel. Those values are then converted, using BitArray, into binary strings. Since the length of both nonce values and typ data are fixed, the program can reserve a set amount of pixels for that data. The data then gets turned into a binary string. One after the other the program replaces the last binary value of each pixel channel(r,g,b,a) with a binary value of the nonces and then of the typ. In order to do so it keeps track of each pixel number to then use that number to determine what bit of the data is used to replace the bit of the pixel channel. After that is done it starts doing the same with the larger message data and stops the process when all of that data has been hidden in the picture.

At the end the for function is stopped and the newly created image is saved under a different name.

Second part the decryption: This part of the function takes advantage of the fact that in the encryption phase the data is hidden in fixed predetermined lengths of pixels. First the backend, salt and kdf are prepared, similar to the encryption phase. Then the user is asked to type in the image name that they want to get the message out of as well as the password that was used to encrypt it. That password is then converted into a key using the kdf algorithm, and since the salt is the same for both the encryption and decryption, the derived key is also the same.

Then the program goes through each pixel similar to the encryption process and reads the last binary letter of each pixel channel instead of replacing it. That then gets added to its respective variable. Now the program doesnt yet know how many pixels it should search in to read the message data. In order to determine the remaining number of pixels that the program has to look into (which will be the data length divided by 4 since each pixel can hide four 1s and 0s in the red green and blue channel) we start decrypting the scanned typ data using the second unencrypted nonce and the password key. We then receive the message data length and message type and can continue to scan through the pixels.

In the end once all the data needed has been read from the image we can decrypt the message cyphertext using the password key. That data now by itself is still just a string of bytes, so we create a new file with the same type as received fro the typ data and write the byte string in that new file and save it under a new name hiddenmesssage.xxx .

In previous versions of the program the decryption part took a surprising amount of time (usually more than a minute), which seemsed weird since the encryption time was shorter when usually it is the decryption part that should be faster. What caused that problem was that the message data was continuously added to the message BitArray, meaning that for every pixel the program had to create more space in the BitArray for the added information. Since the program had to constantly add more space it slowed down a lot. The solution was to set the length of the BitArray at the very beginning so that all the necessary space was created in the beginning. Afterwards the decryption process was reduced greatly and became much faster then the encryption process. For the encryption process though there is no easy way to speed up the process other that hiding more bits in

the pixels. The problem that created thought was that the difference between the original picture and the produced picture became noticeable, meaning that a human could see a distinct blurriness at the point where the pixels had been altered, which contrasted with the otherwise sharp picture.

4.4. Assessment

To summarize the main requirements that were set in the beginning of the project: Create a program that when given a message, password and image will produce a image that has the encrypted message hidden inside of it. Then the program should also be able to, when given a password and the special image, produce the hidden message at the beginning.

The initial goal that was set at the beginning of the semester has been fulfilled. The knowledge required to understand that process has also been learned by the student. The encryption is secure and delivers on integrity authenticity and confidentiality. The message is also successfully hidden in the picture, in a way that one doesnt notice the difference. The user also has the choice if they want to decrypt something or encrypt something first.

The final program has a few features implemented into it that were not required at the beginning. The program is now also able to encrypt not just txt files but also jpg, png and mp3 files. Meaning that it is able to encrypt most files that are small and have a 3 letter long file type. Of course this comes with the downside that any file type that is not 3 letters long wont work. And the code in general could possibly still be refined and made to look more organized.

But all in all, I would say that the requirements have been met and that the end deliverable is satisfy able.

Acknowledgment

I would like to first thank my Project Assistant Tutor ,Benot-Michel COGLIATI, who helped me during the project, explaining the parts of the scientific aspect that I didnt understand and guiding me on how the program should be structured as well as how to make it more secure. Then I also want to thank Nicolas Guelfi who answered other questions about the project and report details. In general I want to thank the whole BSP management team.

5. Conclusion

In conclusion, the BSP project and deliverables were successful. The report might be a bit rough but I think its OK.

References

[1] BiCS Bachelor Semester Project Report Template. https://github.com/nicolasguelfi/lu.uni.course.bics.global. University of Luxembourg, BiCS - Academic Bachelor in Computer Science (2017).