Applied Cryptography (CSE 539)
MCS Portfolio Report

Chirag Bhansali
Arizona State University
Tempe, US
cbhansal@asu.edu

I. Introduction

This is a portfolio report for Applied Cryptography (CSE 539) course taken in Fall 2019 semester. For this course, we had to complete 6 projects/ assignments, each required us to use topics and concepts which were taught during the regular classroom hours.

For project 1, we were suppose to create a new/ modified algorithm and implement the same using either C, C++ or Java for both encrypting and decrypting the 32bits block of plaintext data (txt, pdf or jpeg format) using S-box (substitution) and P-box (permutation), so as to obscure the relationship between the plaintext and cipher and use brute force technique to try and uncover the key used for doing transformation assuming we know the plain and ciphertext. We were also required to record the time needed by the brute force technique to find the key and to comment on whether there is any relationship between the time taken and the type of file present, as well as any weaknesses which could result in breaking the encryption faster.

For project 2, we were provided with the code written in C language for encrypting and decrypting a file, we were to comment on the design of the algorithm used for encryption, whether its a self-contained algorithm or not and to work on two different techniques (brute force and cryptanalysis) to discover key used for the process when we only know the type of file (jpg, pdf, txt) which we are trying to decrypt, report weakness in the encryption process, steps that can be taken to possibly fix that and record the runtime for the two techniques for comparison.

For project 3, we were provided with 10 hashes for which we had to get the passwords for given that the password is unsalted, is alphanumeric and is 4 characters long. We were asked to use two approaches, which are also very commonly used by the hackers to crack the hash of a password i.e. Brute force and Rainbow table attack. We were suppose to write a program each for the above two techniques, record the runtime for them and comment on the implementation steps involved in creating the rainbow table, the size and the reduction function which was used, along with any shortcoming in the approach used for cracking the hash.

For project 4, we were asked to use the previous given encryption algorithm from project 2 and hashing algorithm from project 3 to create a MAC (message authentication code), which are used to maintain the integrity and authenticity of the message. If any intruder tries to modify the message, the hash of the entire data changes and thus the receiver gets to know whether the data is genuine or not, since the hash that is sent and the one that is computed using the MAC process aren't similar. It was C language program that we were to write where the command line required two inputs, the path of the file, which contained the message and the key used for the encryption.

For project 5, we worked with digital certificates which are issued by the CA (Certificate Authority). Using the Python3 libraries for cryptography we had to obtain the subject's public key and modulus for communicating securely between 2 parties, verify the subject's certificate, whether its valid or not, extracted information like subject name, serial number, encryption algorithm, Public key modulus,Public key exponent, private key exponent from both the root and the subject's certificate. Lastly, we had to encrypt a message using the public key present in the subject's certificate and verify it by decrypting the cipher using the subject's private key.

For the last project, we developed a CPP program for implementing the Challenge Response scheme which is used to authenticate whether the party that is been challenged knows the correct answer to a specific challenge or not, test the given RSA classes by providing arguments to its constructors as well as the Blind signature where a sender gets a document signed from the receiver without revealing their identify to them. We are provided with steps for executing each of the above technique.

II. Explanation of the solution

A. Project 1

For encryption part, we modified the Vigenere cipher, which is a poly alphabetic substitution system which uses a key and a double entry table, instead of using alphabets, our cipher uses the hexadecimal characters, so that the size of the table is reduced. We then made the key and the message length to be the same, by repeating the key until they both are of the same length. After going through the substitution cycle,where the plaintext and the key are added and modulus 16 is taken character wise, the intermediate cipher goes through 3 different P-Boxes, to make the cipher more obscure and finding the pattern between the cipher and the plaintext difficult.

For decryption, we pass the cipher through the 3 P-Boxes, and then via the modified vigenere cipher, where the key and the ciphertext values are subtracted and a modulus is taken character wise to get the resultant plaintext back.

After coming up with the algorithm, the logic was translated into CPP language program for both the encryption and decryption. For the brute force part, we modified the decryption code, and since we know both the plain and the cipher text, we started off with key as 0x00 and incremented by 1 until the resultant value from the decryption and the plaintext arent the same.

B. Project 2

We performed the cryptanalysis on the encryption program code, by going through each and every steps in it and understand what is their purpose and how they are transforming the plaintext and the key into the ciphertext. We also analyzed the working of the MD5 (message-digest) code, which was provided, to understand its purpose. On doing the analysis, we found that its not a self contained code and that it "cheats" by not using the MD5 for the encrypting of plaintext.

For the brute force, we modified the decryption code, to start off with the key value as 0x00 and incremented the value by 1 until the XOR value of the first 4 characters of the ciphertext and the key didnt produce the specific required hex file signature matching with the one for which we are trying to find the decrypted version of. And for the result from the cryptanalysis we simply created a program which did the XOR operation between the the first 4 characters of ciphertext and the specific hex file signature to get the desired key value.

C. Project 3

We first generated all the 32 bit long alphanumeric passwords, stored them in a file. For doing the brute force, we read the password containing file, read each password one by one, used the hashing function which was already given to us with the project description and compared it with the given hashes and printed the passwords for which a match was found.

For the rainbow table, we chose table size to be of dimension 62 * 238328, i.e. we chose 62 different passwords, ran the chaining process with the hashing and the reduction function for 238328 times and then stored the resultant hash and the password in a file. We then took the given hashes, one by one, used the reduction function and the hashing function, ran the process for a max of 238328 times or until the intermediate hash generated doesnt match with any of the 62 hash values which we stored. Once, a match is found, the stored password is taken, the chaining process is done and ran until the given hash is not found and thus the password for which we got the given hash is the result we want.

D. Project 4

For creating the intended mac.c file, we as mentioned in the description, we first receive the path of the 1024-bytes file and the key from the command line as arguments. We start with encrypting the given 1024-byte file data using the encryption function from the project 2 using the 32-bit long key,creating a 1028-byte result, we removed the first 4 bytes and concentrated it with the key and took the hash of the concentrated value using the hash function from the project 3 to get the resultant MAC.

E. Project 5

For creating the Python file, we created functions to gather and display various asked information from the given certificate of the subject and root along with the subject 's private key, we used Python3 cryptography libraries like `load_certificate`, `get_signature_algorithm()`, `get_notAfter()`, `get_notBefore()`, `load_pem_public_key`, `dump_publickey`, etc to get the asked information. We also encrypted a message using `encrypt()` and decrypted the cipher using the `decrypt()` function.

F. Project 6

For the CPP file, we followed the instructions which were given for RSA, where we created instances of the RSA class provided with the project to check the encrypt and decrypt functions, by providing either 0 or 1 or 2 parameters to the constructor along with checking whether it works only for prime numbers or not by provided non-prime numbers as arguments to the constructor.

For Challenge-Response scheme, we created two instances of the RSA class and assign the public key and modulus of an instance to another one. Generated a random message, encrypted it using the public key and decrypted it using the private key, and checked if both the random number and the decrypted values are similar or not.

For Blind Signature, we followed the steps as given in the project description, created instances of the RSA classes as Alice and Bob, who communicate with each other with Alice sending a message to be signed by Bob, without Bob knowing the identity of Alice.

III. Description of the result

A. Project 1

With the help of the modified vigenere cipher, we obtained different values for the same character because of the key and the shift changes, making it resistant against the frequency analysis based attacks, and with the help of the 3 PBox we mitigated the issue of key length prediction and made the cipher as obscure as possible when compared with the plaintext.

For the brute force attack, it took about 50 minutes to go through each and every 32 bit long keys and we found no relationship between the file type and the algorithm and all the 3 types took the same amount of time for the brute force. One of the weakness that was there in the algorithm was that the key length is small and thus the brute force process took less time and was easier to decrypt the cipher.

B. Project 2

Firstly, we found that the encryption process isnt using the MD5 algorithm given with the project for the plaintext but was instead encrypting the key and for generating the ciphertext ,the program was doing an XOR between the plaintext and the key.

We found that the time taken via the cryptanalysis is far less than the time taken by the Brute force and that for different file, there are different file signatures which are really useful for decryption process and to get the desired decrypted file and thus each file format took different amount of time to get the decrypted file. For removing the weakness, we suggested to do the MD5 operation on the plaintext instead of the key or to use an operation which generates a bit more complex results than the XOR operation.

C. Project 3

For this project, we found all the passwords for the given hashes while doing the brute force , but for the rainbow table attack, we didnt find the password for 1 of the given hash. The execution time for the brute force is almost 10 times less than the time for the rainbow table and we found that for small sized passwords in this case 4 character long, the results were computed in less than 2 minutes and thus its important to have the password length to be greater than 6 characters for the two attack to be infeasible to perform.

D. Project 4

For this project, we receive the MAC value for the message as the result, which is formed by taking the hash of the concentrated string containing the key and the encrypted message.

E. Project 5

Using the various python3 libraries we verified the subject'certificate. printed the various data about the certificate like subject name, issuer, serial number, encryption algorithm, not valid before, not valid after, public key modulus(n), public key and private key exponent, the hex signature. Lastly, we encrypted a message using the subject's public key. for decryption, used subject's private key.

F. Project 6

We see that the encrypt and decrypt functions only work when prime numbers are given as arguments to the constructor. For the blind signature, where Alice encrypts the random number obtained from Bob with the public key and multiplies it with the message, on which Alice wants to get the signature on and sends it to Bob, Bob decrypts the cipher, and sends it Alice, who multiplies it with the inverse of the random number to get the desired signed message.

IV. Description of my contribution

A. Project 1

In this project, I was responsible to creating the CPP program and the algorithm, design for the modified vigenere cipher and the 3 Pbox which were used for the encryption and decryption process along with the report writing part of the same.

B. Project 2

Modified the decryption code, which was used for getting the desired output key, wrote the report and recorded the outputs for both the brute force and the cryptanalysis along with suggesting to use a stronger algorithm which would be produce results more obscure than the one from the XOR operation.

C. Project 3

Did the precomputation for both the attacks, by generating all the passwords and for generating the table for the rainbow table attack.I came up with appropriate reduction function, which helped in covering majority of the hashes along with the code modification for the hash function, so that it could read the file in case of brute force, and compare the computed hash value with the given hash and in the case of rainbow table, chaining process and finding out if any the intermediate hash is a match within the 62 stored hashes.

D. Project 4

Created the C language program, where the data from a file is encrypted using project 2's encryption function and sending combined value of the resultant string and key to the hash function and get the MAC for the message.

E. Project 5

Did research about which all libraries to use, their implementation examples, and how to use them for the project. Created different python functions which are called for each part and wrote the readme for how to use and gather the necessary information from the program.

F. Project 6

Created the CPP program, where for each part, I created instances for the RSA class and created methods to check whether the results from the decryption function and the plaintext (random number) are similar or not. Created the function for challenge-response and for blind signature, created separate functions showing the communication between Alice and Bob.

V. New skills acquired

During the entire course and project execution, I learnt quite a few new skills and tricks which were really useful for the completion of this course and for using them in real-world scenario.

1) Encryption and Decryption process, steps involved in doing so, about the use of S-Box and P-Box, using them to create an reversible decryption algorithm, the purpose

of doing the encryption and decryption of the data and lastly, about Brute force, the time and effort, it takes to decipher the cipher message, considering we know the algorithm and plaintext and are trying to find the key.

2) Difference between the Brute force and Cryptanalysis techniques, why brute force might not effective and how code, cipher-text, plain-text are analysed to find pattern in them and to use those patterns to come up with flaws and defects in the cryptography algorithm, exploit them and get the resultant symmetric keys used for communication.
3) Rainbow tables, the technique which is used by majority of the hackers to crack passwords given its hash. Learnt about the creation and utilization of the table and key difference in terms of time and memory required for the brute force attack and for rainbow table along with the importance of choosing the right reduction function, for the rainbow table so as to cover maximum passwords and their hash.
4) MAC (Message authentication code), its importance, how it helps in maintaining the integrity and authenticity of the message by using the previous learnt techniques of hashing and encrypting the message.
5) Python3 libraries and their implementation for verifying digital certificate given root and subject's certificate, extracting information from them. Along with using the PKI (public key infrastructure) i.e. the public and private key for encrypting and decrypting the message.
6) RSA encryption, which are used for securely transmitting data, Challenge Response, a technique used for authentication. As well as Blind Signature, which is used for getting the signature of a person without them knowing the sender.

VI. Team Members

All the projects were done in a group of 2. Other team member was Shivank Tiwari.

References

[1] Tutorialspoint.com. Public Key Encryption - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/cryptography/public_key_encryption.htm

[2] Tutorialspoint.com. Feistel Block Cipher - Tutorialspoint. [online] Available at: https://www.tutorialspoint.com/cryptography/feistel_block_cipher.htm

[3] Pyopenssl.org. crypto — Generic cryptographic module — pyOpenSSL 19.1.0 documentation. [online] Available at: https://pyopenssl.org/en/stable/api/crypto.html

[4] X.509 Reference. Available at: https://cryptography.io/en/latest/x509/reference/

[5] En.wikipedia.org. Blind signature. [online] Available at: https://en.wikipedia.org/wiki/Blind_signature