

Encryption Algorithms for Secure Communication over the Internet

BICS student: Desislava MARINOVA*, PhD Student Asya MITSEVA†, Prof. Dr. Thomas ENGEL‡
*Faculty of Science, Technology and Communication, University of Luxembourg,
Luxembourg*

*Email: * desislava.marinova.002@student.uni.lu, † asya.mitseva@uni.lu, ‡ thomas.engel@uni.lu*

Abstract—This document is a template for the scientific and technical report that is to be delivered by any BiCS student at the end of each Bachelor Semester Project (BSP). The Latex source files are available at:

<https://github.com/nicolasguelfi/lu.uni.course.bics.global>

This template is to be used using the Latex document preparation system or using any document preparation system. The whole document should be in between 6000 to 8000 words and the proportions must be preserved. The other documents to be delivered (summaries, ...) should have their format adapted from this template.

1. Introduction ($\pm 5\%$ total words)

We are living in the information age enabled by computing and communication technologies, an age dense with electronic connectivity, eavesdropping and fraud. For all that, when we send messages around the world instantaneously, we do not want them to be intercepted or stolen. Thus, information is an asset that has precious value like any other. Future needs for data security is vital, given society's reliance on a changing panoply of information technologies and technology-enabled services, the increasing global nature of commerce and business, and the ongoing desire to protect freedoms. We all use the Internet, all the data is becoming congested, therefore, how do we maintain privacy, integrity, online-identity? Network based attacks spring up every day, remoteness is not a problem anymore since data has been digitalized, not to mention the issue of privacy being collected, processed and misrepresented, all conducting the vulnerability of Internet security. For this reason, the disciplines of Cryptography and Network Security have matured. They protect financial flows and medical records, defend state secrets and the corporate sort, and make the e-commerce industry possible. Without cryptography, credit-card details, bank transfers, emails and the like would zip around the Internet unprotected, for anyone to see or steal. From obscure science, Cryptography today is one of the major pillars of modern cyber reality. To develop a security mechanism (algorithm or protocol) one must always consider potential attacks, take countermeasures, act like an intruder, bet against ones own intuition and exploit eventual weaknesses in the algorithm.

In order to understand the principles of cryptography, we dive into the terminology that governs our topic in the next paragraph. The word *cryptography* comes from two Greek words meaning *secret writing* [?]. Cryptography is the art and science of concealing meaning, of keeping the enciphered information secret and designing and analyzing encryption schemes. Mathematics is the foundation on which modern encryption rests. Cryptanalysis is the breaking of encoded data and uses statistical and mathematical approaches. The areas of cryptography and cryptanalysis together are called cryptology. The basic component of cryptography is a cryptosystem which is based on substitution and permutation. The former changes characters in the plaintext to produce the ciphertext; the latter is an ordered sequence of elements in a finite set with each character appearing only once. In order to make information secret, we use a cipher a cryptographic algorithm that converts plaintext (data in readable form) into ciphertext (a message in unreadable protected form), unless we have a key that lets us convert back the cipher. The process of making text secret is called encryption, and the reverse process is called decryption.

There are two types of encryption: symmetric and asymmetric. For this project, we shall focus mainly on the symmetric, also called *conventional*. It comprises a single key for both encryption and decryption and it is the preferred choice when we need to encrypt large amounts of data because it is faster and less power hungry mainly on the decryption side. Some of the encryption algorithms that use symmetric key encryption are: DES (Data Encryption Standard), Triple DES, AES (Advanced Encryption Standard), Blowfish, Twofish. One of the disadvantages is related to transferring secure files because the same key has to be used for encryption and decryption. The sender must find a secure way to provide the recipient with the key so that the latter is able to decrypt the files. Otherwise, they risk placing the key in wrong hands, leading to decryption of the encrypted files. Not to mention the high impossibility to distribute a key when the file transfer environment involves multiple users dispersed around the world.

On the other hand, asymmetric cryptography, also called *public key* and *two-key*, makes use of two keys: public and private. The public key is used to encrypt the message, whereas the private key is used to reverse the encryption.

Most widely employed asymmetric key algorithms are RSA and DSA. However, this method is computationally costly [?], thus, cannot deal with large quantities of data due to: the runtime of supporting processes employed; the more storage space required and the more time needed to move over a link. However, it is more secure when we want to transfer over an insecure channel. Moreover, it does not have the issue with multiple user key distribution, as long as the private key is kept secret.

It is not easy to compare the cryptographic strengths of both key encryptions since having longer key lengths in asymmetric encryption does not infer impossibility to break. Nevertheless, the choice to be made on which system to use for encryption and decryption depends on the purposes, requirements, ease of distribution and on the functions of each algorithm. Modern file transfer systems employ a hybrid versions of both symmetric and asymmetric key cryptography as they have their own advantages. For example, we classify SSL and SSH as hybrid cryptosystems.

If someone wants to break the ciphertext, knowing the algorithm but not the specific cryptographic key, we present 3 main types of attacks as there are others as well. And a good cryptosystem protects against all three types of attacks [?]

- Ciphertext only attack : ciphertext known, the goal is to find the corresponding plaintext. The key may be found as well if possible. (A Caesar cipher is susceptible to a statistical ciphertext-only attack)
- Known plaintext attack : both ciphertext and plaintext and known; the goal is to find the key
- Chosen plaintext attack : specific plaintexts are enciphered in order to find out the corresponding ciphertexts and the goal is to find the key.

We often stumble upon impending security threats such as information leakage, integrity violation, denial of service, illegitimate usage, trojan horses, insider attacks, problems related to access or authentication control. Thus, there are instilled security requirements of utmost importance to mention just a few: confidentiality, integrity, authentication, availability, access control, non-repudiation, often intermingled to provide further security. Hence, one of the mechanisms promising to ensure security is encryption, along with digital signatures and hashing.

2. Project description ($\pm 10\%$ total words)

2.1. Domain

The report draws on a variety of disciplines. However, it is impossible to appreciate the vastness and significance of the topic in a limited amount of content pages. Nevertheless, we attempt to make the report self-contained by providing the reader with an intuitive understanding of our survey and application results. The domains that are associated with our project are:

- Internet security: a branch to Computer Security, related to the Internet, comprised of measures to protect transactions done over a collection of interconnected networks
- Computer Security: the process of preventing and detecting unauthorized use of our computer
- Cryptographic primitives: well-established, low-level cryptographic algorithms frequently used to build cryptographic protocols for computer security systems
- Symmetric-key cryptography: same key is used for both encryption and decryption
- Cryptographic algorithms: ensure alteration of data in order to preserve its authenticity, confidentiality, integrity through various security mechanisms such as symmetric encryption, asymmetric encryption and cryptographic hash functions

2.2. Objectives

We aim for a comprehensive survey of both principles and practice of cryptography and network security. The objectives of this project are three-fold.

- 1) As a preliminary task, we address a highly needed background material. We give a basic introduction to the concepts and primitives of cryptography, i.e. transposition/permutation, encryption/decryption, symmetry/asymmetry, block ciphers/stream ciphers and how they obtain their security in constantly emerging attacks
- 2) Our main focus falls on symmetric encryption, taking into account both classical and modern algorithms. Therefore, we explore the following encryption algorithms: Caesar cipher, ROT13, One-time pad, DES, 3DES, AES in order to yield two types of approaches: information theoretic and computational
- 3) Lastly, we implement an encrypted chat system that could be of use to provide network security.

2.3. Constraints

Our project takes into account substitution/transposition; block/stream ciphers but we do not consider:

- Asymmetric ciphers (public-key algorithms, including RSA and elliptic curve);
- Data integrity algorithms (cryptographic hash functions, message authentication codes (MACs); digital signatures));
- Authentication techniques: key management and key distribution topics (protocols for key exchange);

3. Background ($\pm 15\%$ total words)

3.1. Scientific

In this section we present substitution, transposition, stream and block ciphers.

Substitution ciphers: Ciphers have been used long before apparition of computers. During the Ancient Rome times, Julius Caesar invented an encryption for his private correspondence. Today, it is known as the Caesar cipher. Given the English alphabet, the cipher shifts a letter from the alphabet three places further down. To decrypt the message, the other party needs to know both the algorithm and the shifting number.

ROT13 is a simple monoalphabetic substitution cipher, a special class of Caesar cipher, that encodes a certain letter with another letter that is 13 positions after it. Only those letters which occur in the English alphabet are affected. Numbers, symbols, whitespace, and all other characters are left unchanged. It is an example of a cipher providing weak encryption, since both operations encryption and decryption are identical. Hence, this cipher is its own inverse. Because we know there are 26 letters in the English alphabet, if we wish to apply twice 13 it would give us one shift of 26. Thus, it leads us back to the original text. Moreover, the direction of the shift is of no importance, since it will always give the same output [?]. ROT13 is used in online forums as a means of hiding spoilers, punchlines, puzzle solutions, and offensive materials from the casual glance. Furthermore, it has inspired a variety of letter and word games online [?].

The most popular secure example of substitution ciphers is one - time pad, designed in 1917. It is employed by Russian spies, the CIA covert operations and the *hotline* Washington-Moscow. A pad (secret key) is never reused. Instead, it utilizes a random key which has an equal length to the message. The random key is also used in encryption and decryption of the message, and afterwards is discarded [?]. If we want to generate a new message, a new key is required. Thus, this scheme leads to an output unrelated to the plaintext. If we try to decipher a ciphertext using this method, and let us suppose we have succeeded in finding the key, our attempts would lead us to fairly different decrypted outputs. For this reason, it would be difficult to decide upon the correct decryption. According to Stallings, there are two fundamental complexities related to the one-time pad: (1) making large quantities of random keys; (2) key distribution and protection (since a key of equal length is needed by both sender and receiver). Hence, if we ought to employ the one-time pad, it would be best for low-bandwidth channels requiring very high security [?]. Nevertheless, this cryptosystem is regarded as the sole exhibiting perfect secrecy [?]. Cipher machines, presented below in this report, attempted unsuccessfully, first mechanically, then electronically, to create approximations to one-time pads (OTPs). Many snake oil algorithms claim inability to break by claiming to be OTPs, hence, giving a rise to pseudo-OTPs providing pseudo-security [?].

In order to produce the ciphertext, a substitution cipher modifies characters in the plaintext. There are two types of substitution ciphers: simple and polyalphabetic. We can break the former by using letter frequency analysis since letter frequencies are preserved (e.g. E is the most common letter in English). The latter can be broken by decomposing into individual alphabets and as a consequence to treat is as a simple substitution cipher. A historical example of breaking the substitution cipher is the execution of the Queen of Scots, Mary, in 1587 for plotting to kill Queen Elizabeth [?].

Transposition ciphers: Another kind of mapping is achieved by performing permutation on the plaintext letters. This technique is referred to as a transposition cipher. For example, a simple cipher of this sort is the rail fence technique, in which the plaintext is written as a sequence of diagonals and read off as a sequence of rows.

Cipher machines: Cryptography was mechanized by the 1900s in the form of encryption machines. The basic component of a cipher machine is the wired rotor. When we get prior to the Second World War, the Germans realized that, thanks to radios, messages can be sent across the battlefield in an instant but that always meant the other side could also tap into those radio channels. Therefore high-tech encryption was highly needed. The Germans invented Enigma, whose complexity guaranteed their privacy. The Enigma was a substitution cipher, but a more sophisticated one because it used three rotors in a row, each feeding into the next. The Germans typed their messages on a keyboard that came out as gibberish on a lampboard. Then they sent it over the radio to the other side, which also has the same Enigma machine to help them decrypt the unintelligible message. The Germans distributed *cookbooks* with daily settings for the machines. Furthermore, to decrypt the message, it is assumed that the other side must know the algorithm and must have configured the machine the same way as the Enigma machine encrypting the message. The Enigma relies on a random letter generator. Hence, its encryption does not seem to follow any kind of pattern. There are 26 wires coming out of the keyboard, running through three rotors with 6 permutations, going into the lamps representing the output letter. Once the first of the three rotors hits a full evolution, it kicks the next rotor to start moving, yet again, when it completes its cycle, it transmits the process to the last rotor. The inside of the rotors resembles scrambled wiring. A rotor's side has 26 junctions or contacts, accepting the incoming wires and outputting them to the other side of the rotor. The rotor moves each time a letter is typed in. Even if the same letter is passed sequentially, the rotor would move and a lamp representing a different output letter would light up. Thus, the dynamism of the rotors accounts for the complicated encryption. Finally, a plugboard at the front of the machine allowed letters to be optionally swapped so that the machine it is interacting with is configured the same way. Alan Turing and his colleagues at Bletchley Park were able to break the Enigma codes and automate the process by developing a new machine called *the Bombe* [?]. The rotor machines were cracked because the same key had been

used over an extended period of time and due to the use of old compromised keys while encrypting. What is more, the circuits configuration showed that it was impossible for a letter to be encrypted as itself, which turned out to be a cryptographic flaw.

Stream ciphers: A stream cipher generates a stream of bytes, one for each byte of the text we want to encrypt. An example is Rivest Cipher 4 (RC4), designed in 1987 by Ron Rivest. RC4 uses 64-bit or 128-bit key sizes. Its most popular implementation is in WEP for 802.11 wireless networks and in SSL. RC4 consists of key-scheduling algorithm, which initializes a permutation of all 256 possible bytes [?]. The permutation itself is started with a length key between 40 and 2048 bits. RC4 is also composed of pseudo-random generation algorithm (PRGA) which generates the stream of bits. PRGA has two 8-bit index pointers (i and j) on which, during the 256 iterations, it executes operations such as XOR-ing, swapping, modulo. Stream ciphers are vulnerable to attacks, mostly to bit-flipping [?]. Bit-flipping attack deals with changing a bit in a ciphertext so that it results in a predictable plaintext. It is not targeting the cipher itself but a message or a series of messages on a channel. For this reason, it could turn into a Denial of Service attack. We should never reuse a key with a stream cipher. The main reasons being that we can recover the plaintext, using the keystream and the ciphertext. The keystream could be recovered as well using the plaintext and the ciphertext. Furthermore, if we use two ciphertexts from the same keystream, we can recover the XOR-encryption of the plaintexts.

Block ciphers: Block ciphers date back to late 1960s when IBM attempted to develop banking security systems [?]. The result was Lucifer, an encryption method, with 128-bit key and block size, aimed at protecting data for cash-dispensing system in the UK. However, it was not secure in any of its version implementations. When we use block ciphers, each block is encrypted independently producing a ciphertext block of equal length. Block ciphers are also called product and have a cipher structure named *Feistel*, as first described by Horst Feistel of IBM in 1973. The Feistel structure consists of a number of identical rounds of processing. In each round, a substitution is performed on one half of the processed data, followed by a permutation that interchanges the two halves. The original key is expanded so that a different key is used for each round. Symmetric block encryption algorithms are based on this structure. In general, it is accepted that block ciphers are applicable to a broader range of applications than stream ciphers.

DES

Cryptography gradually moved from hardware to software with the advent of computers. Most famous example of the block cipher design and the classic Feistel structure is the Data Encryption Standard (DES), designed by IBM under the advisement of NASA in 1977, standardized 2 years later. It was meant to encipher sensitive but non classified data. DES complexity is comprised of a simple repetition of the primitives of transposition, substitution, split, concatenation and bit-wise operation. In DES we put 64-bit block of

plaintext, DES' key (which does the processing) is 64 bit which is 8 bytes but for each byte there is one parity bit, therefore, the value in the key is only 56 bits which means there are 2 to the power of 56 different keys. The output ciphertext is a 64-bit block. From the 56-bit key, 16 bits are generated (one for each round). Each DES round works consecutively, has the same operations and uses a different key. Each round uses a combination of proper substitution, where we take some bits which are substituted with another combination of bits. Each DES round takes as an input the ciphertext produced by the previous round and outputs the ciphertext for the next round. The input is divided into a left half and a right half. The output left half is just the right half of the input. The right output is the result of XOR-ing the left half of the input and the output of the Mangler Function (which takes as an input the 32-bit right half, expands it to 48-bit (bit-wise operation) then XORs it to 48-bit key, then uses the S-Boxes to substitute the 48-bit value into a 32-bit value). The algorithm process of decryption in DES is the same as the encryption process. It uses the ciphertext as an input to DES but the keys are run in reversed order, i.e. $k = 16$ is used as the first round of decryption, $k = 15$ is used as a second round of decryption and so on, so forth. Diffusion is one of the principles in encryption. It is achieved through permutation (initial transposition). Permutation works by changing the position of the bits in DES. The mixed bits are taken to a sub-box which receives the 56-bit key and the 64-bit plaintext, once it completes processing, it outputs the 64 bits into another transposition subsection, which in turn produces a 64-bit ciphertext. The larger the block size, the key size and number of rounds means greater security. However, when there are more than 16 rounds or more than 56 keys, the security will neither be increased nor the encryption will be made any stronger. The possible attacks that often occur on product ciphers are differential cryptanalysis and linear cryptanalysis, however, according to Stallings, DES has proven to be resistant to these sort of attacks [?]. By 1999, a computer could try every possible key in a couple of days rendering the cipher insecure.

TRIPLE DES/3DES has replaced DES since the simpler version, relying on the same key for encryption and decryption, is susceptible to brute force attacks. 3DES is a more secure method of symmetric key-encryption, as it encrypts data three times than DES, i.e. the one 56-bit key becomes three individual keys rendering a 168-bit key. However, initiating three instances of DES, implies that 3DES is much slower than other methods of encryption. The text is encrypted firstly with key 1, then with key 2 and key 3 encrypts the last text. Triple DES offers a security level of 2^{112} instead of 2^{168} , i.e. with only two keys of encryption, since the 168-bit key can be cumbersome to implement. This method is known as *Encrypt-Decrypt-Encrypt (EDE)*. The process is as follows: *key1 encrypts the message; then the message is decrypted using key2, a second time, and then encrypted again using key3.*

AES The Advanced Encryption Standard replaced DES in 2000 as the US Government encryption technique to

protect classified information. The symmetric block cipher was developed by two Belgian cryptographers Joan Daemen and Vincent Rijmen. AES was designed to be efficient in both hardware and software. It supports a block length of 128 bits and key lengths of 128, 192, and 256 bits. Thus, brute force attacks, i.e. cryptanalytic attacks attempting all possible key variants to decrypt any enciphered data, are much harder to be launched against it. AES chops data up into 16-byte blocks, transferred to a *State* array, and then applies a series of substitutions (e.g. *Substitute bytes*, *MixColumns*) and permutations (e.g. *ShiftRows*), based on the key value. We note that the *State* array undergoes various modifications throughout both encryption and decryption. What is more, that way, it obscures the message, by adding diffusion, confusion and non-linearity and repeating the processes ten (for a 16-byte key) or more times (12 rounds for 24-byte key; 14 rounds for a 32-byte key) for each block. In order to *Substitute bytes*, an S-box is used to perform a byte-by-byte substitution of the block. The other substitution called *MixColumns* utilizes arithmetic over GF(28) [?]. In fact, all encryption algorithms necessitate arithmetic operations. The key is expanded into an array of key schedule four-byte words. Employing a bitwise XOR of the current block with part of the expanded key is a stage called *AddRoundKey*, solely used on the key. Therefore, the cipher is locked around this stage adding to the security of the AES encryption. At the end, the *State* of the plaintext is copied to an output matrix where the bytes are ordered in a column. Similarly, the bytes of the expanded key, which form a word, are placed in the column of the matrix. Today, AES is used everywhere, from encrypting files and sensitive data, transmitting data over WiFi with WPA2, to accessing websites using HTTPS.

3.2. Technical

One of the deliverables of this project is a Python chat system. The application encrypts messages and employs ciphers reviewed in this report. Python is a programming language created by Guido van Rossum in late 1980s in the Netherlands. It is named after a BBC comedy show called Monty Python's Flying Circus. Python is a free software and the latest version can be downloaded from www.python.org. Python is a simple and very powerful general purpose computer programming language. It is very readable, useful and easy to learn. Python is an open source language with volunteers constantly trying to improve it. This allows for the language to remain fresh and current with the newest trends. Python has libraries for just about everything. It can be used for web development, web scraping, writing scripts, browser automation, GUI development, data analysis, machine learning, computer vision, game development to name just a few. In addition, Python is an object-oriented programming language (OOP). There are four pillars of OOP: encapsulation, abstraction, inheritance and polymorphism. Before OOP there is procedural programming, a simple and straightforward programming, that divides a program into a set of functions. However, as our programs grow we end

up with *spaghetti code*, i.e. many functions all over the place that are interdependent. Thus, OOP offers a solution to this issue. We can bundle a group of related variables (referred to as properties) and functions (referred to as methods) that operate on them into an object. This grouping is called encapsulation. Using this technique, we can reduce complexity and increase reusability. In Python, objects are data and have a certain type (integer, float, list, etc.). Once we have created our objects we can manipulate and interact with them (append, sort, delete, concatenate, etc.). We can hide the details and complexity (e.g. some methods and properties) from the outside and show only the essentials. This gives us benefits such as simpler interface and reduction of the impact of change (i.e. no inner changes leak to the outside of the contained object). Inheritance is a mechanism allows us to eliminate redundant code. Polymorphism means *many forms* and is a technique that allows us to refactor long *if/else* or *switch/case* statements.

On macOS, we can open Python on a Terminal. Moreover, we can use any Text Editor or IDE of our choice. For this project, we have used the source code editor Visual Studio Code (VSC) developed by Microsoft for Windows, Linux and macOS. Our platform is macOS. VSC is a lightweight yet powerful editor. It offers support for debugging, embedded Git control, syntax highlighting, snippets, extensions, smart code completion and code refactoring.

A GitHub repository was set up to share and build our project. The link is provided in the annex of the report. GitHub is a web-based hosting service of open source projects for version control using Git.

To build the chat program for our project we need a client and a server. To establish a connection between both and be able to transmit information back and forth over the Internet, we need a socket link. Sockets aid the communication between these two entities. The client requests information from the server and the server sends a response and serves data to the client. The server is just a software, a program running and waiting for connections. Similarly, the client is a program as well. However, altogether, they are referred to as *client/server architecture*. For instance, when we visit a website, we are using a socket and accessing a port of the web server. In this case, the server has generally port 80 open, used to transfer HTTP data. Other websites have ports 21 and 20 open for FTP access, which is not very secure, some have port 22 intended for SSH. The lower number ports are specific ports, whereas the higher number ports signify general purpose rights. More often than not, questions arise regarding security when using higher number ports. To use sockets, we need to import them by typing in `import socket`. We do not need to install anything as they are part of our standard library. Next we need to specify and create a socket by: `s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)` where AF_N

Our client should send an initial request to the server's port number, *the listening port*. To keep the communication over TCP, we have the client's IP address and local port number as well as the server's IP address along with its port number. Each client that connects to the server gets unique

set of values of two IP addresses and two port numbers in a tuple, a collection of items. However, the listening port of the server remains intact.

4. BSPro - A First Bachelor Semester Project in BiCS-land

4.1. Requirements ($\pm 15\%$ total words)

Describe here all the properties that characterize the deliverables you produce. It should describe what are those deliverables, who are the actors exploiting the deliverables, what are the expected functional and non functional qualities of the deliverables.

4.2. Design ($\pm 20\%$ total words)

Provide the necessary and most useful explanations on how those deliverables have been produced.

4.3. Production ($\pm 20\%$ total words)

Provide descriptions of the deliverables concrete production. It must present part of the deliverable to illustrate and explain its actual production.

4.4. Assessment ($\pm 15\%$ total words)

Provide any objective elements to assess that your deliverables reached or not the requirements described above.

Acknowledgment

The authors would like to thank the BiCS management and education team for the amazing work done.

5. Conclusion

The conclusion goes here.

6. Appendix

All images and additional material go there.