DATA ENCRYPTION

Abdallah Chidjou

University of Indianapolis

Department of Computer Science

CSCI-491: Computer Science Capstone Project

Semester II, 2018-2019

Table of Contents:

Abstract …………………………………………………………………………………………...3

Introduction………………………………………………………………………………………. 4

Basic concepts……………………………………………………………………………………. 4

Encryption and decryption………………………………………………………………………...5

Type of cryptographic algorithms…………………………………………………………………6

Caesar cipher………………………………………………………………………………………8

Multiplication and affine cipher………………………………………………………………….10

Java Multiplicative Cipher……………………………………………………………………….12

Python Multiplicative Cipher…………………………………………………………………….21

Matrix multiplicative cipher……………………………………………………………………. 29

RSA……………………………………………………………………………………………...30

Abstract

The gradual enhancement of computer technology has evolved almost everything to be digitalized from businesses, organization or government to simple everyday task such as shopping or even paying bills.

In the ancient world as well as in this modern time, delivery and more importantly safety of information from source to the destination have always been a huge concern; being so, numerous approaches and methodology such as cryptography have been introduced to ensure information and data integrity. The purpose of this documentation is to instigate how computer technology such as cryptography help us achieve this goal: Data integrity.

I will be discussing different aspects of cryptography, its users with a broad range of cryptological concepts and protocols.

**Introduction**

Cryptography is the science or technic of using mathematic to encrypt and decrypt data; applied to communication links, storage devices, software, and messages used in a system. Cryptography has a long and important history in protecting critical systems and sensitive information.

The Germans used Enigma machines to encrypt communications during the Second World War, and the Allies went to great lengths to crack the encryption. Enigma machines used a series of rotors that turned plaintext into cipher-text and the Allies were able to decrypt the cipher-text into plaintext by understanding the position of the rotors.

This was a momentous achievement, but it took significant manpower and resources. Cracking some encryption techniques is still possible today; however, attacking other aspects of cryptographic systems such as protocols, integration points, or even the libraries used to implement cryptography is often more feasible.

This Document has two major purposes; basic concept behind cryptographic methods and provide some real examples of cryptography using powerful programming language such as python and java with graphical user interface.

**Basic concepts:**

Cryptography is a conceptualized method using computation and mathematical algorithms to ensure data security and integrity, assures that information and programs are changed only in a specified and authorized manner. This method is referred to as data encryption and decryption based on an ultimate key which may vary depending on the set of algorithms used. If the key or the data is modified, the used algorithm will produce a wrong result in another word nonsense statement. A well encrypted data is referred to as strong cryptography; weak otherwise. Strong cryptography would be impossible to decrypt; very difficult to decipher without possession of the appropriate decoding information (the key).

There are five primary functions of cryptography:

Privacy/confidentiality: Ensure that no one, except the intended receiver, can read the message.

Authentication: The process of proving the identity of one.

Integrity: Assuring the receiver that the received message has not been altered in any way from the original.

Non-repudiation: A mechanism for proving that this message was actually sent by the sender.

Key exchange: The method by which crypto keys are shared between sender and receiver.

**Encryption and decryption:**

Data that can be interpreted or understood without being modified is referred to as plaintext or cleartext. The concept or methods of disguising plaintext into an unreadable format or coded is called encryption. Plaintext encryption results into unreadable text format called ciphertext. The process of reverting ciphertext to its original plaintext format is called decryption. These methods are used to ensure data integrity from source to destination mostly across an insecure network.

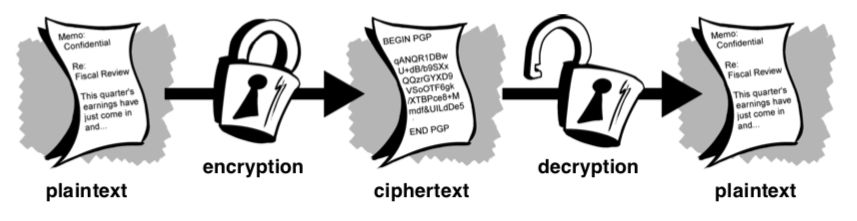


Figure1: Illustration of the process.

(<https://www.akadia.com/download/documents/intro_to_crypto.pdf>)

Using Cryptography can enable us to store sensitive data or transmit it over insecure networks such as the internet so that anyone other than the recipient won’t be able to read it.

**Type of cryptographic algorithms:**

There are several ways of classifying cryptographic algorithms. Every algorithm has specifics set of rules mainly identified by a unique key in the process of encryption and decryption. The use of key could vary depending on the algorithm. The following image illustrates a few examples of how key drives algorithms.

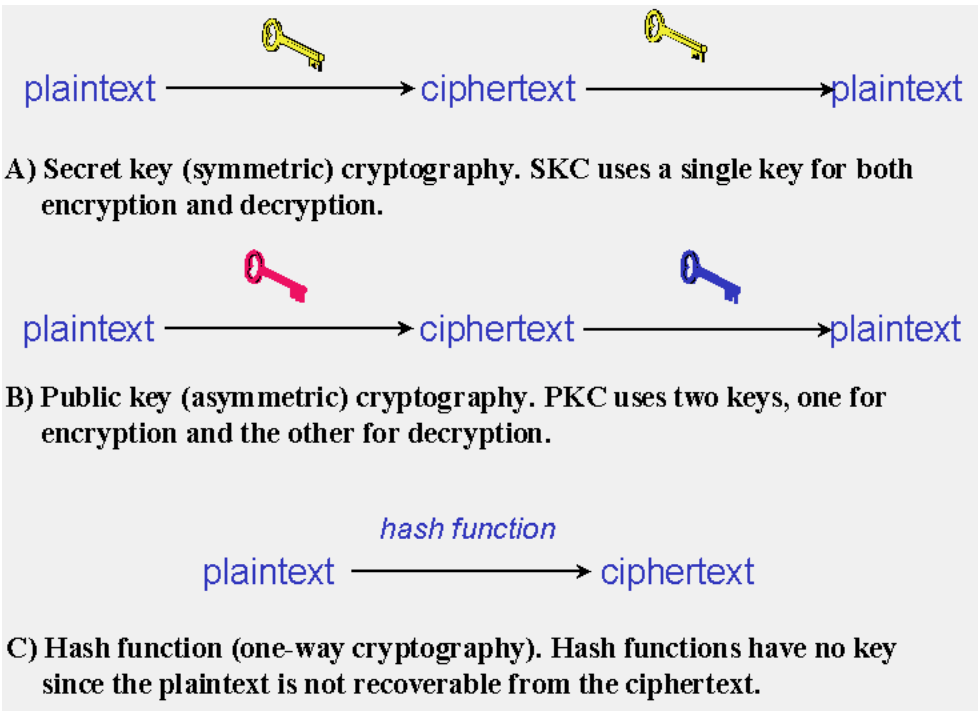
****

Figure2: Illustrate variation of Keys (Gary C. Kessler)

According to the Network Associates (1990-1999) A key is a value that works with a cryptographic algorithm to produce a specific ciphertext. Keys are basically really, really, really big numbers. Key size is measured in bits; the number representing a 1024-bit key is darn huge. In public key cryptography, the bigger the key, the more secure the ciphertext. (This decreases the chances of whomever may want or try to interrupt or corrupt the data.)

However, public key size and conventional cryptography’s secret key size are totally unrelated. A conventional 80-bit key has the equivalent strength of a 1024-bit public key. A conventional 128-bit key is equivalent to a 3000-bit public key. Again, the bigger the key, the more secure, but the algorithms used for each type of cryptography are very different and thus comparison is like that of apples to oranges. (The key size depends on the operating system architect)

While the public and private keys are mathematically related, it’s very difficult to derive the private key given only the public key; however, deriving the private key is always possible given enough time and computing power. This makes it very important to pick keys of the right size; large enough to be secure, but small enough to be applied fairly quickly. Additionally, you need to consider who might be trying to read your files, how determined they are, how much time they have, and what their resources might be.

Larger keys will be cryptographically secure for a longer period of time. If what you want to encrypt needs to be hidden for many years, you might want to use a very large key. Of course, who knows how long it will take to determine your key using tomorrow’s faster, more efficient computers? There was a time when a 56-bit symmetric key was considered extremely safe

Keys are stored in encrypted form. PGP stores the keys in two files on your hard disk; one for public keys and one for private keys. These files are called keyrings. As you use PGP, you will typically add the public keys of your recipients to your public keyring. Your private keys are stored on your private keyring. If you lose your private keyring, you will be unable to decrypt any information encrypted to keys on that ring. (In terms of encryption or decryption, there is an absolute combination allowing keys to be dependent or independent based on the algorithm)

**Caesar cipher:**

In cryptography Caesar cipher commonly refer to as shift cipher, is one of the earliest and simplest known ciphers. It is a type of switch or substitution cipher in which each letter in the plaintext is 'shifted' a certain number of places down the alphabet. For instance, with a shift of 3, A would be replaced by D, B would become e, and so on. The shift could take on any number. But in this case, the number of shifts representing the key is the same for both (encryption and decryption). The method is named after Julius Caesar, who apparently used it to communicate with his generals.

The encryption can be done using modular arithmetic by first replacing the alphabetic letters into numbers, according to the following scheme, A = 0, B = 1…, Z = 25. Encryption of a letter by a shift n can be mathematically described as illustrated in the following figure:

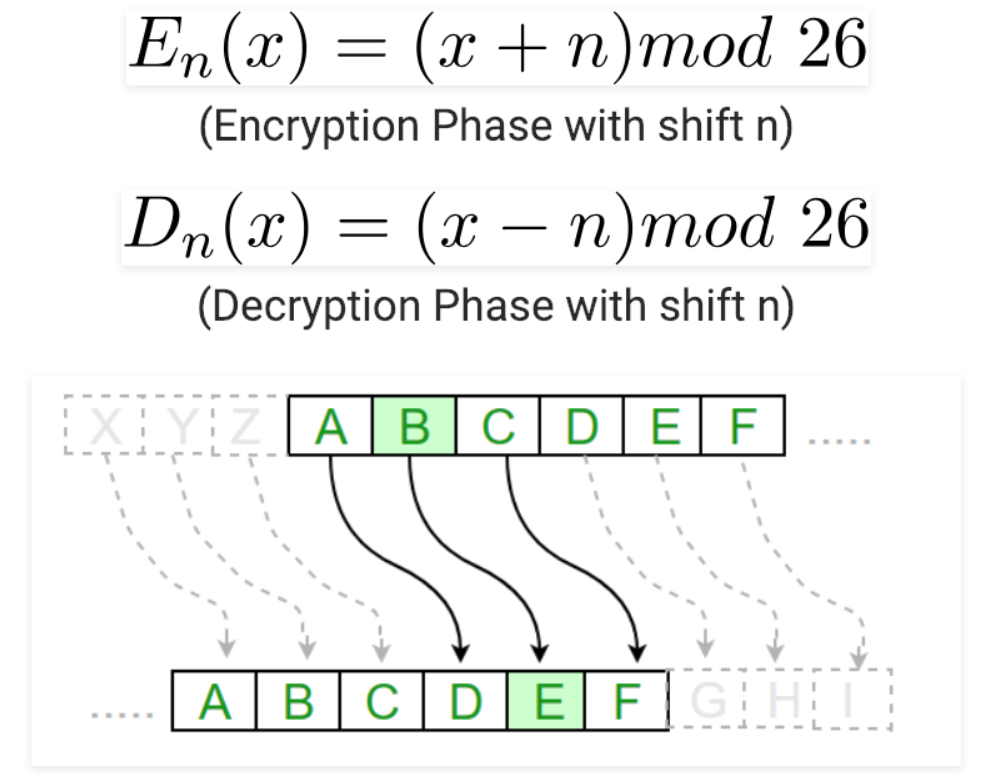


Figure 3: Mathematical description of the shift cipher. (<https://www.geeksforgeeks.org/caesar-cipher/>)

**Example 1:**

Encryption

**Text**: ABCDEFGHIJKLMNOPQRSTUVWXYZ

**Shift**: 23

**Cipher**: XYZABCDEFGHIJKLMNOPQRSTUVW

Decryption

**Text**: ABCDEFGHIJKLMNOPQRSTUVWXYZ

**Shift**: 23

**Cipher:** XYZABCDEFGHIJKLMNOPQRSTUVW

**Multiplication and affine cipher:**

Multiplication and affine ciphers are somewhat similar to the Caesar cipher (additive); except the fact that instead of adding a key to a symbol’s index in a string or plaintext, these cipher use multiplication­­­­­­. Encrypting with multiplication, instead of encoding by adding a constant, we multiply each plain letter by a secret key k. Since each plain letter turns into 0 for k=0 and remains unchanged for k=1, we start with k=2 using (MOD 26) since we have 26 alphabetic letters. It goes the same way for the affine by just adding a constant which would be consider in this case as a second key; contrary to Caesar cipher using the same key for encryption and decryption, the multiplication and affine cipher use a totally different key to decrypt the ciphertext; this is done by computing the inverse of the

original key. The inverse is generally computed using the GCD (Grand Common Divisor)

between the original key and the number used for the modulus and must be equal to one.

gcd(k, n) = 1

**Multiplicative cipher:**

**Encryption:**

C = (M \* k) mod n

with (M) the original message,

(k) the original key and n the modulus number.

**Decryption:**

M = (C \* k-1) mod n

k-1 → multiplicative inverse of k (key)

**Affine Cipher:**

**Encryption:**

C = [(M \* k1) + k2] mod n

with (M) the original message,

(k1, k2) the original keys and n the modulus number.

**Decryption:**

M = [(C - k2) \* k1-1] mod n

k1-1 → multiplicative inverse of k1 (key)

I will be illustrating on the following pages an implementation of the multiplication cipher using java and python programming language.

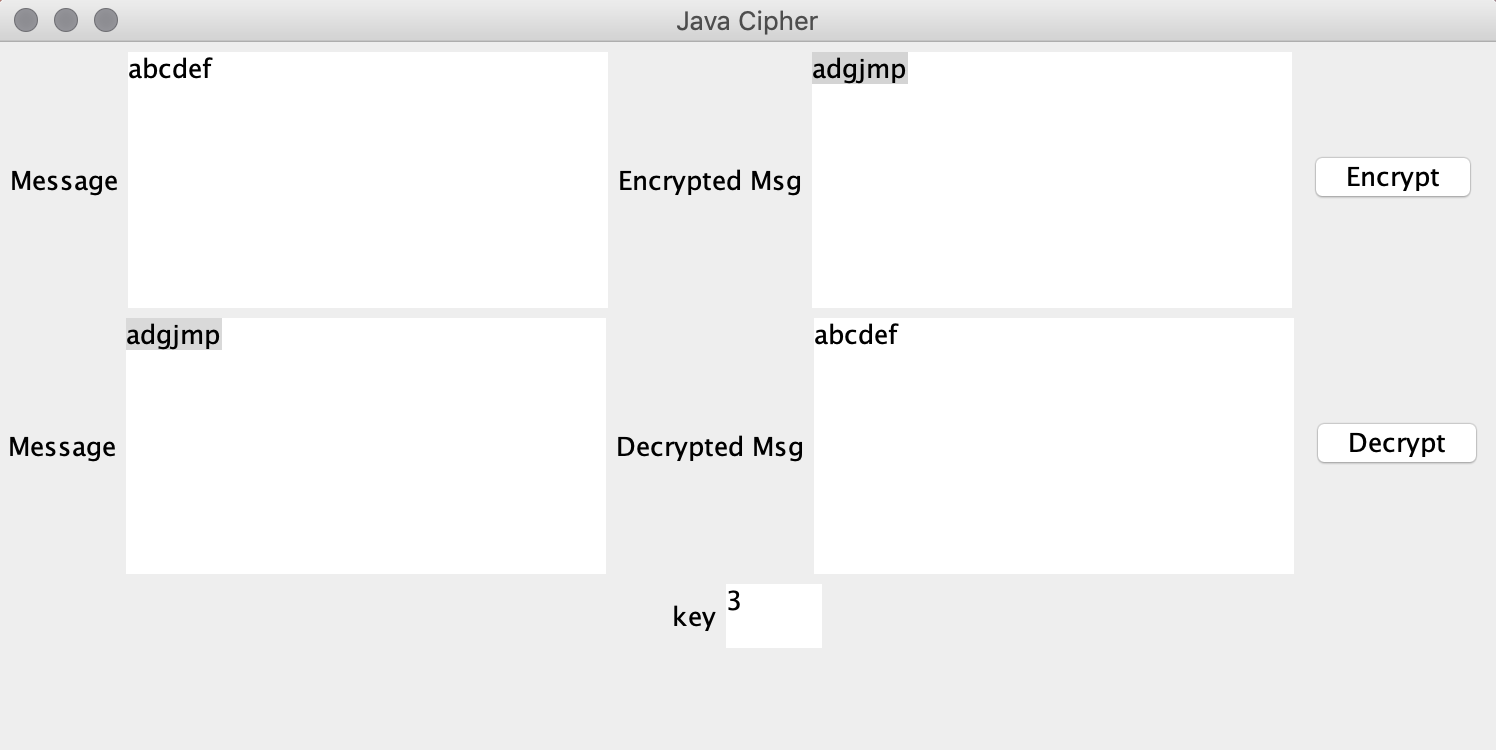
**Java Multiplicative Cipher:**

Author: Abdallah Chidjou

03/07/2019

**Output:**

Java Graphical User Interface Implementing multiplicative cipher.



**Figure 4:** Java GUI Cipher.

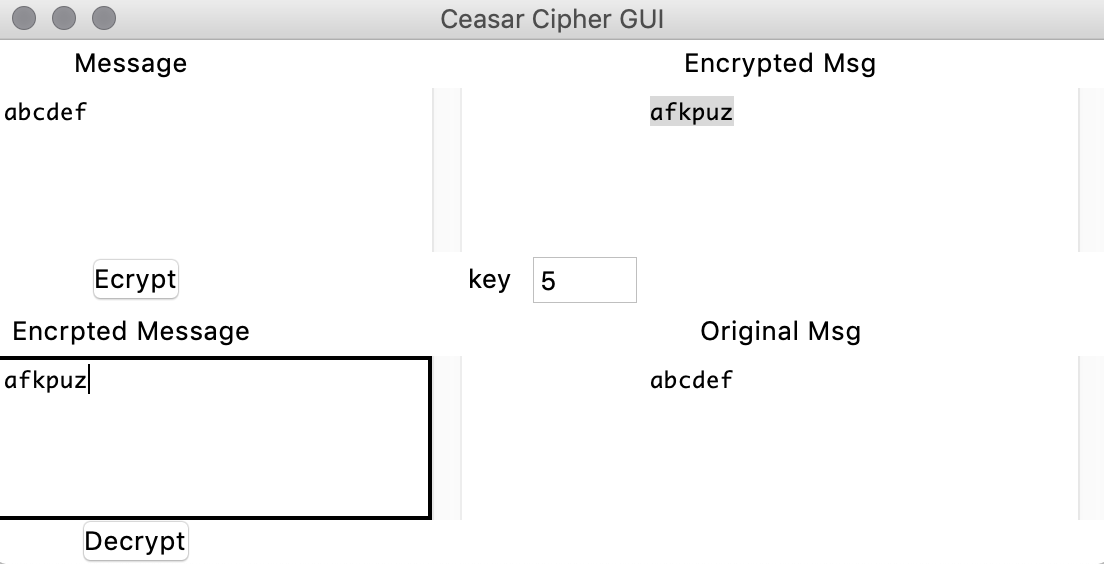
**Python Multiplicative Cipher:**

Author: Abdallah Chidjou

03/07/2019

**Output:**

Python Graphical User Interface Implementing multiplicative cipher.



**Figure 5:** Python GUI Cipher.

**Matrix multiplicative cipher:**

Just like a simple multiplicative using a unique number for the cipher; matrix multiplicative cipher uses an accurate set of number called matrix of any size; 2x2, 3x3, 4x4 and so on.

The use of matrix for the encryption in this case, however, must meet certain conditions. The chosen matrix must first be invertible; meaning the determinant of the matrix must be different from zero. And the grand common divisor gcd between the determinant of the matrix and chosen modulus N number must be equal to one.

So, having a matrix A of size nxn: gcd(det(A), N) = 1.

The decryption of the ciphertext in this case is computed using the inverse matrix of the original matrix.

**Brief Description:**

MultiplicationCipher\_Matrix(A) (mod N) \_encrypt\_2x2\_

Encryption: E = A \* M (mod N)

INPUT: abcdef (orig. msg), A= (1 2), N=27

(1 3)

OUTPUT: cdilo (encrypted msg)

Decryption: M = Inverse(A) \* E (mod N)

INPUT: cdilot (encrypted msg), B = Inverse(A) = (3 25), N=27

(26 1)

OUTPUT: abcdef (decrypted msg)

**RSA:**

RSA is an asymmetric cryptographic algorithm using essentially two keys; one public which is known by everyone and one private kept secret. The algorithm computes large integers for the keys. The generated keys can typically be 1024 or 2048 bits long; very difficult to break (Hack).

RSA Algorithm

o Choose 2 large prime numbers p & q 23 \* 57

o Compute n = (p­­­­ \* q) and z = (p - l) (q - I)

o Choose a number, e, less than n, that has no common factors (other than 1) with z. (e will be used in encryption)

o Find a number, d, such that ed-1 is exactly divisible by z (d will be used in decryption)

o Choose d such that (e \* d) mod z = 1

The pair of number (n, e) constitute the public encryption key and (n, d) the private decryption key.

Encryption:

* Message represented by bit pattern m
* Cipher Text (C) = me mod n

Decryption:

* m = cd mod n

They are many more solution and proposed idea in solving data integrity issues and yet to be discovered.

Work Cited:

Gary C. Kessler, (11 March 2019), An Overview of Cryptography (<https://www.garykessler.net/library/crypto.html>)

An Introduction to Cryptography, (1990-1999), (<https://www.akadia.com/download/documents/intro_to_crypto.pdf>)

Stallings, William. Cryptography and Network Security, Person 2017