The Implementation of Both Elliptic Curve Cryptography and Steganography to Broadcast Message through an Image

Juan Christopher Santoso - 13521116
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail : juan.csantoso@gmail.com

***Abstract*— Elliptic curve cryptography (ECC) is a key-based technique for encrypting data. As how it is named, the ECC uses the fundamentals of mathematical elliptic equations as its foundation. The ECC is frequently discussed in the common context of security. On the other hand, Steganography is a technique of hiding data within an ordinary or non-secret file. The common media to be used for steganography is an image. Therefore, the usage of an elliptic curve for data encryption and steganography for data hiding is expected to be effective in sending secret messages through images.**

Keywords—elliptic curve cryptography; steganography; security; images;

I. Introduction

In terms of developing a specific system, security is a priority substance. Developers have to make sure that the system is protected and secure, especially if the system contains private and confidential data. Thus, to strengthen their system security, developers have been using a technique to secure information and communication, which is cryptography.

Throughout history, cryptography has been a help for humanity in order to maintain the security of communication. However, cryptography itself is not absolute. Every technique and method within cryptography has its own strengths and weaknesses. Therefore, developers have been researching for a way to secure information better and better. On the other hand, hackers (or we can say breachers) have also been trying their best to crack the security of information.

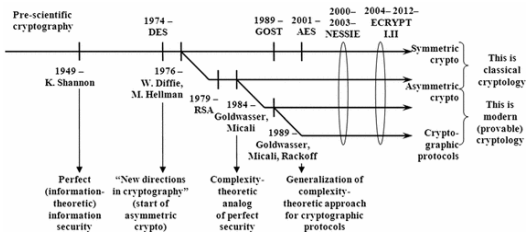

Image 1.1 The History of Cryptography
(Source: Springer Link)

Cryptography comes with many fields of study. Some of them are computer science and mathematics. Developers would like to use better equations to produce better security for information. That is when mathematics comes in. Methods like RSA (Rivest-Shamir-Adleman), DH (Diffie-Helman), and ECC (Elliptic-Curve Cryptography) have the fundamental basis of mathematics.

Cryptography also comes with various media. The most common media for cryptography is text. However, people would become more and more creative in hiding messages. Nowadays, messages can be hidden inside visual media, such as images, videos, and animations. When people become more creative, that is when the steganography comes in.

II. Theoretical Basis

A. *Steganography*

Steganography, by definition, is the technique of hiding data within an ordinary, non-secret file or message to avoid detection [1]. Unless the information arrives at the rightful destination, hiding data within an ordinary file is expected to lower suspicion. The hidden data will be later extracted at its destination.

Steganography is derived from the Greek words, *steganos*, which means hidden or covered, and *graphien*, which means to write. Therefore, steganography can be used to conceal almost any type of digital content, including text, image, video, or even audio.

Cryptography and steganography are similar yet have different objectives. Cryptography is used to hide the meaning of content. On the other hand, steganography is used to hide the existence of the content itself.

Steganography comes with various approaches. One commonly used approach in steganography is the LSB method. LSB (Least significant bit) method is a technique to hide a message inside an image by changing the least significant bit of its pixel. Since the difference is minimal, the difference itself is undetectable by human eyes [2].

Image 2.1 Visualization of LSB
(Source: Towards Data Science)

In this method, the message will be converted into binary data. The binary data will be inserted as the LSB of a sequence of pixels inside the image. Therefore, the receiver can later read the hidden message by reading the right LSB of pixels inside the image.

B. Elliptic Curve Cryptography (ECC)

Elliptic curve cryptography (ECC) is a key-based technique for encrypting data. ECC focuses on pairs of public and private keys for decryption and encryption processes [3]. The computation with ECC is expected to offer the same security level as the other algorithm with smaller keys to be used.

As how it is named, ECC uses the fundamentals of elliptic equations. The general equation for ECC is:

$$y^2 = x^3 + ax + b$$

$$with\ 4a^3 + 27b^2 \neq 0$$

The a value and b value can be randomized and up to users.

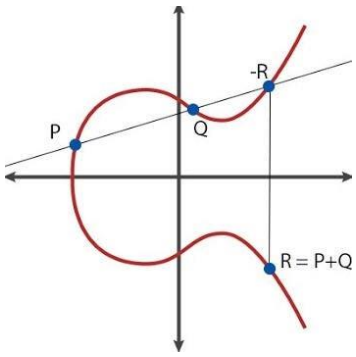

Image 2.2 Visualization of ECC
(Source: Medium)

The usage of ECC often comes with the addition and multiplication of points. All the points used within the concepts of ECC are only the points along the curve of ECC. Therefore, several concepts need to be understood in order to use ECC [4]:

1. Point Addition

   Point addition is the most common operation within the ECC concepts. Assuming there are two points of P and Q. The addition of points uses the concept of gradient where gradient (m) equals to,

$$m = \frac{Yp - Yq}{Xp - Xq}$$

   By using the value of gradient, the result point can be calculated with the formula of,

$$x = m^2 - Xp - Xq$$

$$y = m(Xp - Xr) - Yp$$

2. Point Doubling

   Point doubling has a similar fundamental concept to point addition. In this case, both points that are added are the same (P = Q). Therefore, to prevent dividing by zero when calculating the gradient, the gradient equation is adjusted to the derivation of the y value by the x value:

$$m = \frac{dy}{dx} = \frac{(3Xp^2 + a)}{2Yp}$$

3. Point Multiplication

   Point multiplication has a similar concept to general multiplication in mathematics. In order to solve a simple problem such as 3 times x, we use the logic of,

$$3x = x + x + x$$

   Therefore, point multiplication within the ECC also works the same as if it was a sequence of addition iterations. Assuming we have a point P, then the multiplication of P by 3 will be equal to,

$$3P = ((P + P) + P)$$

   Concepts such as commutative and associative are also working fine in ECC. Thus, users can solve front-to-back or back-to-front just fine.

4. Inverse Point

   Points in the ECC are symmetry to the x-axis. Therefore, two points can have the same x value while bearing a different y value. If two points have the same x but different y, then the two points must be the inverse of one another. Assume the point P has the coordinates of (x, y). Thus, the inverse point of P, which is P', has the coordinates of (x, -y).

5. Peak Point

   There is a condition where a point does not have an inverse. It is a condition when a point is located at the peak of the curve itself.

6. Infinity point

   The infinity point works as the 'placeholder' for the points that are located at the coordinates of infinity, which is not calculatable. There are several concepts regarding the infinity point:

   a. If a point is being added to the inverse point of itself, the operation will return the infinity point as the result.

b. If a peak point is being added to another peak point, the operation will also return the infinity point as the result.

c. If a point is being added to the infinity point, the operation will return the point instead.

7. Base Point

It is a random point along the curve that both users agree to use. This point will be the fundamental point for encryption and decryption process.

ECC has been developed to a concept where ECC is used within a Galois Field [5, p. 1]. Galois Field is used to limit the probability of points within the ECC. Using the ECC with the Galois Field method, the general equation of ECC itself is adjusted to be:

$$y^2 = x^3 + ax + b \ (mod\ p)$$

where p is a prime number and the values a and b should be within the range of {0…p-1}. Other equations such as the addition and gradient calculation will also be operated by the modulo of p.

C. *Elliptic Curve ElGamal (ECEG)*

Elliptic Curve ElGamal (ECEG) is the advancement as well as the combination concepts of ECC and ElGamal. It is used to exchange messages using private and public keys with ElGamal-like exchange but using the fundamental calculations of ECC [6].

The main concept of ECEG will be explained as:

1. A secret point will be generated as the 'secret message'. Another value k will also be generated for ciphering process.
2. The encryption will be done by using the public key of the receiver.
3. The encryption will result in a pair of points and will be sent to the receiver.
4. The receiver will decrypt the pair of points using the private key that they have.

The pair of points that are generated from the encryption process is:

$$Pc = [(kB), (Pm + kPb)]$$

while

k : random generated value
B : base point
Pm : secret point
Pb : the receiver public key

The receiver can later decrypt and get the secret point by doing the calculation of

$$(Pm + kPb) - (bKb)$$

with b is the receiver's private key.

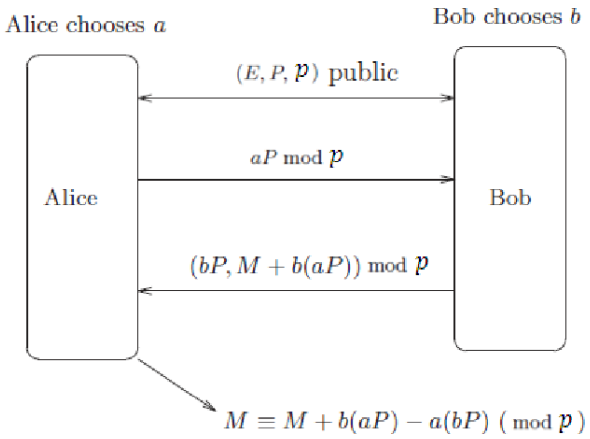

Image 2.3 Visualization of ECEG
(Source: Semantic Scholar)

III. Application

The program is implemented in the programming language of Python. In implementing the system, this program is equipped with several classes to support the performance of the system itself. The classes are:

1. Point

Here is the implemented program of the Point Class.

```
class Point:
  def __init__(self, x: int, y: int):
    self.x = x
    self.y = y

  def setValue(self, x: int, y: int):
    self.x = x
    self.y = y

  def copyPoint(self, p: object):
    self.x = p.x
    self.y = p.y

  def isSamePoint(self, p: object) -> bool:
    return self.x == p.x and self.y == p.y

  def isInverse (self, p: object, pVal: int) ->
bool:
    return self.x == p.x and self.y == abs(pVal
- p.y)

  def getPointNumberValue(self) -> str:
    return "x: "+str(self.x)+", y:
"+str(self.y)

  def getPointValue (self) -> str:
    xHex = makeNumToHex(int(self.x)).rjust(8,
'0')
    yHex = makeNumToHex(int(self.y)).rjust(8,
'0')
```

```
    return xHex + yHex

  def setPointValue (self, val: str):
    xHex = val[:8]
    yHex = val[8:]
    self.x = makeHexToNum(xHex)
    self.y = makeHexToNum(yHex)

  def changeToInverse (self):
    self.y *= -1
```

Since almost every calculation in ECC uses a point type (x, y), the author chooses to implement a Point Class in order to simplify further calculations.

There are two forms in order to define the value of a point, the first one is the original type where the point type is an object that has the attribute of x and y. The other form is a string form where a point is converted into a hexadecimal string, with each x and y value occupying 8 bytes of string.

$$Form\ 1: (35344,\ 187),$$
$$Form\ 2: 0x00008a10000000bb$$

2. ECEG

   ECEG is the main class of this program. This class controls almost every aspect including the ECC setup, encryption, and decryption. The list of crucial functions will be listed as below:

Functions	Usage
calculatePoints()	To calculate all the valid points when setting up the ECEG class.
calculateY()	To calculate the y value by an x value as the input.
calculateCoorX()	To calculate the new x value in point addition operation.
calculateCoorY()	To calculate the new y value in point addition operation.
addPoint()	To calculate point addition.
minusPoint()	To calculate point subtraction.
multiplyPoint()	To calculate point multiplication
encryptECEG()	To do encryption process based on current ECEG setup.
decryptECEG()	To do decryption process based on current ECEG setup.

Beside these classes, there are also some crucial components to control the flow of the programs. The components consist of:

1. Encryption

   This component controls the flow when users try to encrypt message using ECC and Steganography. This component includes the setup process of the ECC itself. This component generates a secret point which will later be encrypted.

2. Decryption

   This component controls the flow of decryption process by users. This includes the setup of ECEG, where it should be the same setup as the encryption process. Lastly, this component also decipher the value hidden in the image and recover the hidden secret point.

3. Image Process

   This component is the main core of the steganography process within the system. This component includes functions such as reading images, saving images, manipulating images for encryption, and searching in images for decryption process.

IV. Implementation

The implementation program will be divided into two, the processes and the result. The processes itself will also be divided into setup process, encryption process, and decryption process. On the other hand, the result will be divided into the data and the analysis.

The implementation of the program is not using the ideal number of ECC components. Instead, this program use the limit of 50,000 to 100,000 in generating numbers for ECEG setup.

1. Process

   a. ECEG Setup

```
Choose (e) to Encrypt and (d) to Decrypt.
-> e
Make sure to insert the image inside the /test folder.
Insert the image filename: img_1.jpg
Image img_1.jpg has the height of 700px and width of 1245px.
ECC Data:
 a: 92221
 b: 99923
 p: 73973
Base Point: x: 1625, y: 73899 | Hex Value: 00000659000120ab

User A Data:
 Private Key: 67783
 Public Key: x: 46191, y: 10531 | Hex Value: 0000b46f00002923

User B Data:
 Private Key: 86599
 Public Key: x: 7727, y: 50928 | Hex Value: 00001e2f0000c6f0

Total runtime of the setup process is 0.3538641929626465
```

The ECEG setup is included within the encryption process. On this step, every process is done automatically. Therefore, users only need to wait until the process is done.

   b. Encryption

```
Insert the Hex Value of the Public Key to be used: 00001e2f0000c6f0

Secret Point: x: 48162, y: 197 | Hex Value: 0000bc22000000c5
K Value: 72893

Encrypted Value Hex: 000006d100000f5c0000229c00008322
Encrypted Value Bin: 1101101000100000000000000000000011110101110000000000000000000
1000101001110000000000000000000100000110010010
Encrypted Value Attribute: 8254943

Image has been saved inside the /test folder.
Encrypted image will have the tag "-enc".

Total runtime of the encryption process is 0.4003932476043701
```

On this step, users need to input the desired public key to be used for encryption process. The rest of the process is done automatically.

c. Decryption

```
Make sure to insert the image inside the /test folder.
Insert the image filename: img_1-enc.py
Image not found! Please make sure to enter a valid image filename.
Make sure to insert the image inside the /test folder.
Insert the image filename: img_1-enc.png
Image img_1-enc.png has the height of 700px and width of 1245px.
Insert 'a' Value: 92221
Insert 'b' Value: 99923
Insert 'p' Value: 73973
Insert x Value of Base Point: 1625
Insert y Value of Base Point: 73899
Insert Private Key to be used: 86599
Insert Encrypted Attribute: 8254943

Encrypted Value Bin: 1101101000100000000000000000000011110101110000000000000000000010
0010100111000000000000000000100000110010010
Encrypted Value Hex: 000006d100000f5c0000229c00008322

Decrypted Secret Point: x: 48162, y: 197
Decrypted Secret Point Hex: 0000bc22000000c5

Total runtime of the decryption process is 0.14153170585632324
PS D:\Folder_Kuliah_Cadangan\Sems_6\Kriptografi\Kriptografi_ECC_Steganografi>
```

On this step, users need to input the all the ECEG setup that are displayed in the encryption process. A slight wrong in setting up the ECEG can result a complete incorrect result.

2. Result

This is the result of the processes by the program. First, here is the result of the LSB method of the steganography.

Original Image 1

Image with Hidden Message 1

Original Image 2

Image with Hidden Message 2

As how it is appeared, the difference between encrypted and original image is not detectable by the human eyes. However, if we look deep down into the pixel of the system, the pixel value changes by the LSB method that is generated from the encryption process of ECEG.

```
Encrypted Value Bin: 1110000110011000000000000000001010011111001001010000000000000
0101000011000101110000000000000000011011100110110 0
Encrypted Value Attribute: 26831808
Before Manipulation
 [230, 227, 234, 233, 236, 233, 234, 236, 233, 230, 234, 237, 240, 239, 239, 239,
239, 239, 238, 237, 239, 236, 236, 236, 236, 233, 232, 231, 228, 226, 225, 226, 22
5, 226, 226, 225, 226, 225, 226, 226, 225, 227, 230, 223, 223, 236, 240, 239, 239,
 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 2
39, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239, 239
, 239, 239, 239, 239, 239, 238, 237, 239, 239, 239, 239, 239, 239, 239, 239, 239,
238, 245, 206, 150, 149, 181, 207, 203, 204, 204, 204, 204, 204]
After Manipulation
 [231, 227, 235, 232, 236, 232, 234, 237, 233, 230, 234, 237, 241, 238, 238, 238,
238, 238, 238, 236, 238, 236, 236, 236, 236, 232, 232, 230, 228, 226, 225, 226, 22
5, 226, 226, 225, 227, 225, 227, 226, 224, 227, 230, 222, 223, 236, 241, 238, 238,
 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 238, 239, 238, 239, 2
38, 238, 238, 238, 239, 239, 238, 238, 238, 239, 238, 239, 239, 239, 238, 238, 238
, 238, 238, 238, 238, 238, 238, 238, 236, 238, 238, 238, 238, 238, 238, 239, 239,
238, 245, 207, 151, 148, 180, 207, 203, 204, 205, 205, 204, 204]
```

The duration of the program also exponentially increase as the limit of number generator increases. This is affected because the process of iterations and loop within the program.

Limit of Generated Number	Duration
5,000 – 10,000	0.04312s
50,000 – 100,000	0.47625s
500,000 – 1,000,000	3.80143s
5,000,000 – 10,000,000	67.1654s

V. Conclusion

The implementation of both ECC and steganography can effectively be used to hide both the meaning and the existence of the messages. Therefore, this combination can be functionally used to secure a secret message through two media, text and

image. However, the implementation of this program itself is still open to further development, in quality yet in performance. The ECC implementation in this program is still not achieve the number of bits that are commonly used in ECC. There will be also a need to improve the calculation of ECC processes. However, in terms of functionality, this ECC and steganography has passed the requirements in securing information.

SOURCE CODE

The source code of the program is accessible on: https://github.com/Gulilil/Kriptografi_ECC_Steganografi

ACKNOWLEDGEMENT

First, I would sincerely express my gratefulness to the Lord that this paper can be completed well and within the given deadline. Then, I would like to express my deepest gratitude to:

1. Mr. Dr. Ir. Rinaldi Munir, M.T. for all the cryptography lectures that have been given to me,
2. My closest friends, that has been supporting me for the time being, and
3. The rightful owner of all the references that I used in writing this paper, for every knowledge that has been passed upon me.

REFERENCES

[1] M. Semilof, "What is Steganography?" Accessed: Dec. 06, 2024. [Online]. Available: https://www.techtarget.com/searchsecurity/definition/steganography#:~:text=Steganography%20is%20the%20technique%20of,then%20extracted%20at%20its%20destination.

[2] R. Munir, "Steganografi Bagian 1," Accessed: Dec. 06, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/07-Steganografi-Bagian1-2024.pdf

[3] AVINetworks, "Elliptic Curve Cryptography Definition." Accessed: Dec. 06, 2024. [Online]. Available: https://avinetworks.com/glossary/elliptic-curve-cryptography/#:~:text=Elliptic%20Curve%20Cryptography%20(ECC)%20is,Adleman%20(RSA)%20cryptographic%20algorithm

[4] R. Munir, "ECC Bagian 2,"

[5] R. Munir, "ECC Bagian 1," Accessed: Dec. 06, 2024. [Online]. Available: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2023-2024/22-ECC-Bagian1-2024.pdf

[6] CriticalError, "ElGamal with Elliptic Curves." Accessed: Dec. 06, 2024. [Online]. Available: https://crypto.stackexchange.com/questions/9987/elgamal-with-elliptic-curves

STATEMENT

Hereby, I state that this paper is written on my own, not an adaptation, or translation from other people's paper, and not plagiarism.

Bandung, 12 Juni 2024

Juan Christopher Santoso - 13521116