ElGamal Cryptosystem with an Application

Metehan Ozten
CS293G

June 2018

1 Introduction

The ElGamal Cryptosystem is a public-private key encryption system, known as an asymmetric cryptosystem because it involves a system in which different users will have different keys used for encryption and decryption[Roe]. In a public-private key cryptosystem, the public key is distributed freely to other users and used to send encrypted messages that can only be decrypted by someone with the private key associated with that specific public key. I intend to go over the three primary components of the ElGamal cryptosystem which are the key generation algorithm used to generate keys, the encryption method which utilizes a public key to encrypt a given plaintext, and finally a decryption algorithm which requires the private key in order to decrypt a given ciphertext [Mei]. Towards the end of the paper, I will also present a novel application of the ElGamal Cryptosystem, which is a modification to the TCP protocol, which provides end-to-end encryption with some interesting properties.

The ElGamal cryptosystem also has several other unique features that I will discuss throughout the paper such as its use of an ephemeral key, and the fact that it is a form of homomorphic encryption [Aca].

2 Key Generation

Before any encryption or decryption can occur, a user must first generate their own keys (or if they are only encrypting, at least receive the public key via some communication medium) using the cryptosystem's unique key-generation algorithm. The key generation algorithm can be concisely described as follows:

1. Alice should select three parameters: $\boldsymbol{p}$, a large prime number, α (which should be less than p), and finally a random number d that falls within $2 < d < (p-2)$ [Elg85]

2. Alice then computes the following:

$$\beta = \alpha^d \pmod{p}$$

3. In this case, β would be considered Alice's the public key, with the system-wide parameters $\boldsymbol{p}$ and $\boldsymbol{d}$. In some notations, the tuple ($\boldsymbol{p}$, α, β) would be considered Alice's public key, since a user would need to know all three parameters in order to successfully encrypt a message that only Alice could decrypt [Leo].

3 Encryption

Once the relevant parties have generated their keys (any user that will be receiving data through an encrypted channel, will need a public-private key pair) and exchanged them with each other, then users can begin encrypting data for certain parties. In order to understand how encryption is performed in the ElGamal Cryptosystem, I will provide a generalized example wherein Bob wants to send an encrypted message to Alice, operating on the assumption that Bob has already received Alice's public key, which consists of the following parameters ($\boldsymbol{p}$, α, β) [Leo].

Bob will start the process out by choosing an integer K_e, which is a random integer that is less than p but greater than 0. This is known as the ephemeral key, because it is constantly changing [Aca]. It grants the ElGamal Cryptosystem an interesting property in which, the same chunk of data encrypted twice produces different ciphertexts and have an extremely small probability $\frac{1}{p-1}$ of collision, due to the random generation of K_e. The incorporation of this ephemeral key, prevents an attacker from detecting patterns in messages and message-timings by comparing cipher-texts [Leo].

Next, after choosing a sufficient ephemeral key, Bob then begins computing the first of the two components of the ciphertext for the message he wishes to send to Alice. Denote the message Bob wishes to send Alice in plaintext as $\boldsymbol{M}$, and the eventual ciphertext as $\boldsymbol{C}$, then

$$C \equiv (\alpha^{K_e} \pmod p, \beta^{Ke} * M \pmod p)$$

If the plain text is larger than what can be encoded into a single number less than $\boldsymbol{p}$, the number is then broken down into chunks and each piece is encrypted individually (which are also both less than $\boldsymbol{p}$), which is the reason why encrypted messages are two times as long as their corresponding plaintext in the ElGamal cryptosystem.

4 Decryption

The final algoritm within the ElGamal cryptosystem is decryption, in which a user with a private key that corresponds to a public key that was used to produce a given ciphertext, decodes that given ciphertext in order to retrieve it's plaintext equivalent.

Let's assume Alice just received a ciphertext $\boldsymbol{C}$ which is a tuple of messages, containing the following fields, $(\alpha^{K_e} \pmod p), \beta^{Ke} * M \pmod p$. The unencrypted

plaintext can be obtained by multiplying the first element in the ciphertext tuple, to the negative d power, by the second element in the ciphertext tuple, or in other words

$$(\alpha^{K_e})^{-d} * \beta^{K_e} * M \pmod{p} \equiv (\alpha^{K_e})^{-d} * (\alpha^{d})^{K_e} * M \equiv M \pmod{p}$$

where ***d*** is Alice's private key [Leo].

5 Homomorphic Encryption

The ElGamal Cryptosystem contains an interesting property in which the encryption function (E) and the plaintext messages M_1 and M_2, which is a group homomorphism from the cyclic group G to the group $G \times G$, obeys the following property:

$$E(M1) * E(M2) \equiv E(M1 * M2)$$

[Elg85]

5.1 Application

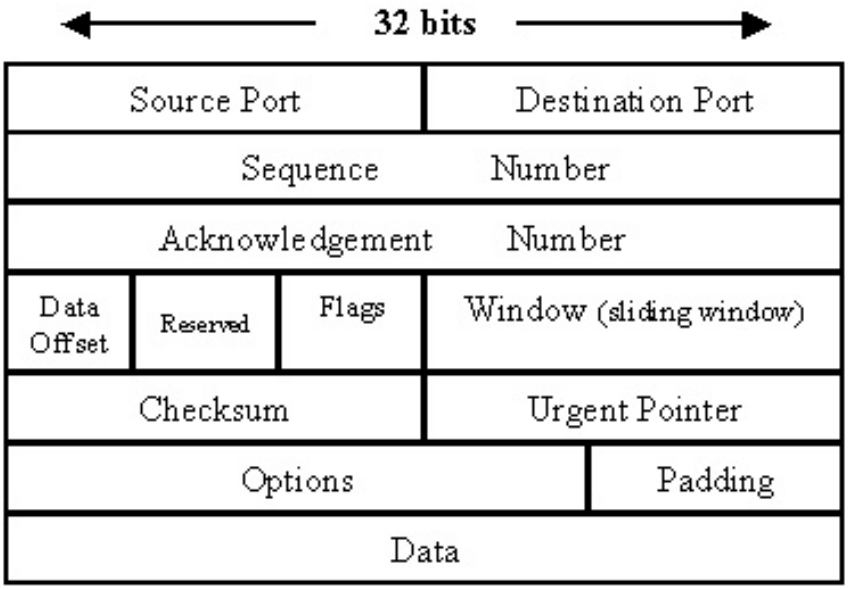

Figure 1: TCP Packet Structure [Mul]

The TCP communications protocol is the most widely used transport layer protocol for the vast majority of applications, with few exceptions. Within Figure 1, shown above, the TCP packet structure is displayed with its internal fields labeled. The application that will be discussed exists as an optional feature within the TCP protocol. Besides existing as an option within the protocol, the use of this option will add an additional checksum to each packet, located at the beginning of the payload section, which will provide additional error-detection capabilities.

5.1.1 During 3-Way Handshake

During the initial 3-way handshaking of the TCP protocol, each end of the TCP channel will indicate in the 'options' portion of their handshaking packet (shown in Fig. 1) that it wants to use encrypted data transport [Ine]. If both users indicate that they want to use encrypted data transport then each user's packet payload which is placed into the 'data' portion of the packet will from this point be encrypted using the public-key received from the other user.

5.1.2 After Initialization

The TCP protocol, at this point will begin computing an additional checksum after sending each packet, by encrypting the whole data portion of the packet and multiplying each portion of the ciphertext by each other and using that as the checksum (leaving the original checksum portion of the packet, as only a checksum over the non-payload data, allowing the recipient to distinguish between errors in the packet overhead and the packet payload). This multiplicative checksum will be placed at the front of the 'data' portion of the packet. The homomorphic property comes in handy at this point as it allows for the checksum to be calculated by the original senders and intermediate carriers directly on the encrypted data, making it simpler to be handled in embedded hardware settings and allowing the network to find checksum errors in packets as they are traveling to their destination and preemptively dropping them before they waste further bandwidth.

6 Conclusion

The ElGamal Cryptosystem's properties make it an interesting object of analysis. Due to some less desirable properties, such as the relative difficulty of computing the cryptographic primitives as compared to those used in other asymmetric public-private key cryptosystems and the fact that the ciphertext generated is twice as long as the plaintext, it is not used often in practical applications. Many applications choose RSA which coincidentally shares the same homomorphic property as the ElGamal cryptosystem [Tes].

Sources

[Elg85] Tamer Elgamal. "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms". In: *IEEE Transactions on Information Theory* IT-31.4 (1985). DOI: http://caislab.kaist.ac.kr/lecture/2010/spring/cs548/basic/B02.pdf.

[Aca] Crypto Academy. *The ElGamal cryptosystem*. URL: https://cryptographyacademy.com/elgamal/. (accessed: 06.10.2018).

[Ine] Inetdaemon. *TCP 3-Way Handshake (SYN,SYN-ACK,ACK)*. URL: http://www.inetdaemon.com/tutorials/internet/tcp/3-way_handshake.shtml. (accessed: 06.10.2018).

[Leo] Jeffrey Leon. *The ElGamal Public Key Encryption Algorithm*. URL: http://homepages.math.uic.edu/~leon/mcs425-s08/handouts/el-gamal.pdf. (accessed: 05.08.2018).

[Mei] Andreas V. Meier. *The ElGamal Cryptosystem*. URL: http://wwwmayr.in.tum.de/konferenzen/Jass05/courses/1/presentations/Meier%20Andreas%20The%20ElGamal%20Cryptosystem.pdf. (accessed: 05.09.2018).

[Mul] Michael Mullins. *Exploring the anatomy of a data packet*. URL: https://www.techrepublic.com/article/exploring-the-anatomy-of-a-data-packet/. (accessed: 05.09.2018).

[Roe] Tom Roeder. *Asymmetric-Key Cryptography*. URL: https://www.cs.cornell.edu/courses/cs5430/2013sp/TL04.asymmetric.html. (accessed: 06.10.2018).

[Tes] Edlyn Teske-Wilson. *Homomorphic Cryptosystems*. URL: https://www.fields.utoronto.ca/programs/scientific/10-11/numtheoryconf/Edlyn_Teske_Wilson.pdf. (accessed: 06.9.2018).