Encryption Algorithm: ElGamal

Ari Torczon, Fabricio Junqueira, Andres Vargas, Bao Tran, Aidan Kumar, Austin Pan

24 March 2025

CS 4600.01

Professor Mingyan Xiao

Report

In a time period where nearly every aspect of modern life occurs via digital communications, it is becoming increasingly important to guard data against unauthorized access. One of the many encryption calculations designed to address this problem is the ElGamal encryption algorithm, which is notable as a revolutionary version of a public-key cryptosystem. An algorithm established in the mid-1980s by cryptographer Taher ElGamal, it presented a working instance of asymmetric encryption that was based on the intractability of discrete logarithms. The algorithm design allows secure communication to convey data without the need to establish shared secrets, showing innovation at the time. Nowadays, ElGamal is relied upon by many systems and protocols, such as, digital signature schemes, (secure) chat protocols, etc. This paper seeks to provide an overview of how the algorithm works, the existing applications in the real world, and the limitations that developers and security experts must consider in its implementation.

Key generation in ElGamal relies on the idea of the Diffie-Hellman key exchange process. This process allows the secure generation of a symmetric cryptographic key, where two parties that do not have prior knowledge of each other can jointly establish a shared key. To start key generation, one party (the receiver) will be responsible for generating the public key, by choosing a very large prime number, q. This value will be used to generate a cyclic group G, which contains the set of integers from 1…q-1. A generator value, g, will then be found and calculated within G, so that a relationship can be established through modular arithmetic. Then, the receiver will randomly choose a value in G, called x. This will be the private key for the exchange, with the public key, h, being calculated as $h=g^x$. Then the values (G, q, g, h) will be passed as the public key for any sender to use. Then, a sender may use these values to establish their own private key, and encrypt a message which can only be decoded by the receiver's private key. This process is possible due to the discrete logarithm problem, where no efficient and reliable method has been discovered to reverse engineer the values passed through the exponential equation found in key generation.

ElGamal's encryption process follows the principles of asymmetric cryptography, utilizing key pairs to ensure secure communication. The Encryption process begins with the sender generating an ephemeral session key, K, which is used to generate a unique ciphertext that changes with every new encryption session. Using the receiver's public key, h, the sender then calculates the first ciphertext component, C1, as $C1 = g^K \bmod q$. The second component of the tuple, C2, is then obtained by computing $C2 = M * h^K \bmod q$, where M is the original plaintext message, which is generally encoded using ASCII or other padding techniques. The resulting ciphertext tuple (C1, C2) is then transmitted to the receiver. Since K is randomly generated for each encryption, the same message encrypted twice will yield different ciphertexts, further enhancing security. The reliance on the discrete logarithm problem ensures that, even if the

public key is known, recovering the private key is computationally implausible via brute force attacks, becoming basically computationally impossible if the chosen prime, q, is large enough. This further ensures that the confidentiality of the encrypted message is preserved, as only the receiver with the corresponding public and private keys can decrypt and recover the plaintext.

The decryption phase of the ElGamal encryption scheme is the final step where the recipient, possessing the private key, recovers the original plaintext message from the received ciphertext. Upon receiving the ciphertext, which consists of the pair (C1, C2), the recipient employs their private key, detonated as X, to reverse the encryption process. Calculating an intermediate value, S, defined as $s = C1^x \bmod q$, is the first step in the decryption process. This step successfully reverses the impact of the random ephemeral key K that was introduced during encryption by using the private key X. Next, the modular inverse of s modulo q, denoted as s^{-1}, is computed. This inverse is a crucial element that allows the recipient to decode the original message. Lastly, the original plaintext message M is recovered by multiplying "C2" with the modular inverse of S modulo q, represented as $M = C2 * s^{-1} \bmod q$. The original message M remains after this process effectively eliminates the h^k that was added during encryption. The secret key X must be in possession for the decryption to be successful. Without it, the ciphertext cannot be deciphered because it is computationally impossible to derive S. Therefore, the private key's confidentiality ensures that only the intended receiver can access the original message.

Here is an example process including calculations for key generation, encryption and decryption. Bob wants to send a message to Alice, which just contains the number 5. First, Alice must generate a public key, so she chooses a random prime number 7, which she uses to generate a cyclic group G, where she finds the generator g to be g = 3. Then she randomly chooses x = 4 from set G, where she calculates the public key $h = g^x \bmod q \Rightarrow 3^4 \bmod 7 = 4$. This is the public key, (7, 3, 4). Now for encryption, Bob, the sender, generates a session key K = 3. C1 must be calculated using $g^k \bmod q \Rightarrow 3^3 \bmod 7 = 6$. Next, C2 must also be calculated using $M * h^k \bmod q \Rightarrow 5 * 4^3 \bmod 7 = 1$. Bob sends the ciphertext (C1,C2) = (6,5). This is decrypted using $s = C1^x \bmod q \Rightarrow 6^4 \bmod 7 = 1$. Find the inverse of s which is $s^{-1} = 1$. Then find the original message M with $M = C2 * s^{-1} \bmod q \Rightarrow 5 * 1 \bmod 7 = 5$, the original message.

El Gamal has two major applications, encryption and digital signatures. In settings where public key cryptography is required, El Gamal can be considered as a viable option due to its reliance on the Diffie-Hellman key exchange. In addition to this, the Diffie-Hellman key exchange also allows for the creation of digital signatures, with the private key being used to create a digital signature for a message and the public key being used to verify a signature. There are two major advantages that come with this, security, which is based on the discrete logarithm problem and is hard to solve, and asymmetric key distribution, which makes key distribution and communication between multiple parties more secure since different encryption and decryption keys are used. Despite its applications and advantages, there are some trade offs. First, El Gamal

is slower compared to other encryption algorithms, especially with long keys which can make it impractical for certain applications that require fast processing speeds. Second, El Gamal requires larger key sizes to achieve the same level of security as other algorithms which can be more difficult to use in some applications. Lastly, El Gamal is vulnerable to attacks based on the discrete logarithm problem such as the index calculus algorithm which can reduce the security of the algorithm in certain situations.

ElGamal encryption represents a key development in the history of public-key cryptography. Built on the hardness of discrete logarithms, it offers a solid approach to performing secure communication in an environment that cannot be trusted. While it has issues with randomness and an expansion of ciphertext and can have difficulties implementing these systems, most are thought to be easy to overcome with careful design and using current technology. ElGamal is used as part of a variety of cryptographic applications, including secure email services and blockchain technologies. Those who design or evaluate any cryptographic system need to understand how ElGamal works and its weaknesses. As the digital world enlarges, nothing will replace algorithms like ElGamal in providing information integrity and privacy.

References

- Tutorials Point. "Public Key Encryption." Tutorials Point, https://www.tutorialspoint.com/cryptography/public_key_encryption.htm. Accessed 11 Mar. 2025.

- "ElGamal Encryption Algorithm." GeeksforGeeks, https://www.geeksforgeeks.org/elgamal-encryption-algorithm/. Accessed 11 Mar. 2025.

- "Public Key Encryption." GeeksforGeeks, https://www.geeksforgeeks.org/public-key-encryption/. Accessed 11 Mar. 2025.