El Gamal Cryptosystem

The El Gamal Cryptosystem uses the same two global elements that the Diffie-Hellman Key Exchange uses, a large prime number which we'll call $\boldsymbol{q}$, and a primitive root, which we'll call $\boldsymbol{\alpha}$.

The user Alice (A), to whom messages can be sent does the following:

1. Generates a random integer X_A such that $1 < X_A < q - 1$. (This is her private key.)

2. Computes $Y_A = \alpha^{X_A} \bmod q$. (This is her posted public key others use to send messages to her.)

Here is how Bob (or anyone for that matter), can send an encrypted message to Alice:

1. M, the plaintext message must be an integer in between 0 and q-1, inclusive.

2. Bob chooses a random integer, k, $1 \leq k \leq q - 1$. (Note: Each choice of k he selects results in a different ciphertext. Thus, for this cipher, a single plaintext maps to many possible different ciphertexts, and each of those ciphertexts maps back to the same plaintext. The value of k that Bob selects determines WHICH of the possible ciphertexts for M will get created.)

3. Compute $K = Y_A^{\,k} \bmod q$. This is a piece used to create a portion of the ciphertext.

4. The ciphertext is the ordered pair (C_1, C_2), computed as follows:

$$C_1 = \alpha^k \bmod q \qquad C_2 = KM \bmod q$$

Here is how Alice recovers the plaintext:

1. She needs to figure out Bob's value of K and can do so as follows:

$$K = C_1^{\,X_A} \bmod q.$$ (If you look carefully, this is precisely Diffie-Hellman!)

2. She computes $K^{-1} \bmod q$ via the Extended Euclidean Algorithm.
3. She recovers the message by calculating $(C_2 K^{-1}) \bmod q$.

First, let's prove she recovers the message from Bob:

Recall that $C_1 = \alpha^k \bmod q$ and $K = Y_A^{\,k} \equiv (\alpha^{X_a})^k \equiv \alpha^{kX_a} \bmod q$.

Now, she calculates $C_1^{\,X_A} \equiv (\alpha^k)^{X_A} \equiv \alpha^{kX_a} \bmod q = K$, as shown above.

Once she knows K, as previously discussed, she can obtain K^{-1}, and it should be fairly clear that

$C_2 K^{-1} \equiv (KM)K^{-1} \equiv (KK^{-1})M \equiv M \bmod q$, thus, Alice properly recovers the plaintext.

The key drawback to this method is that the size of the ciphertext is twice the size of the plaintext, and the time used in the process is slightly more (a couple steps added to a modular exponentiation for both encryption and decryption compared to RSA).

Let's see an example with small values, so we can see the effect of picking different values of k for encryption.

Let's choose q = 11 and α = 2. Let Alice choose $X_A = 4$, so $Y_A = 2^4 \bmod 11 = 5$.

Here is a chart of 2 raised to each power mod 11:

Exp	0	1	2	3	4	5	6	7	8	9	10
Value	1	2	4	8	5	10	9	7	3	6	1

Here is a chart of all possible ways to encrypt M = 7, selecting values of k from 1 to 10:

M	k	K	C_1	C_2	$C_1^{X_A}$	K^{-1}	$K^{-1}C_2$
7	1	5	2	2	5	9	7
7	2	3	4	10	3	4	7
7	3	4	8	6	4	3	7
7	4	9	5	8	9	5	7
7	5	1	10	7	1	1	7
7	6	5	9	2	5	9	7
7	7	3	7	10	3	4	7
7	8	4	3	6	4	3	7
7	9	9	6	8	9	5	7
7	10	1	1	7	1	1	7

Thus, each of the ten ordered pairs (2, 2), (4, 10), (8, 6), (5, 8), (10, 7), (9, 2), (7, 10), (3, 6), (6, 8) and (1, 7) are all possible encryptions of the plaintext M = 7.