ElGamal cryptosystem

For ElGamal cryptosystem the encryption and the decryption function are of the form

\[
e_{k_1} : \mathbb{F}_p^* \rightarrow \mathbb{F}_p^* \times \mathbb{F}_p^* \quad k_1 = g^{k_2} \in \mathbb{F}_p^* \quad \text{public encryption key}
\]

\[
d_{k_2} : \mathbb{F}_p^* \times \mathbb{F}_p^* \rightarrow \mathbb{F}_p^* \quad k_2 \in \mathbb{F}_p^* \quad \text{private decryption key}
\]

where

\[
e_{k_1}(m) = (g^r, mk_1^r) \quad \text{and} \quad d_{k_2}(c_1, c_2) = c_2c_1^{-k_2}
\]

with \(g \in \mathbb{F}_p^*\) of large order (ideally a generator) and \(r\) an arbitrary (random) number. Implicitly a plaintext message is an element \(m \in \mathbb{F}_p^*\) and a ciphertext is a pair \((c_1, c_2) \in \mathbb{F}_p^* \times \mathbb{F}_p^*\).

In Sage the encryption function is

```python
# ElGamal
# pub_key = public key
# g = generator or high order element
# p = prime
# message = number < p
def elgamal_encrypt(pub_key, g, p, message):
    k = floor(1+(p-2)*random())
    return (Mod(g,p)^k, message*Mod(pub_key^k,p))
```

and the decryption function is

```python
# ElGamal
# pri_key = private key
# g = generator or high order element
# p = prime
# (m1,m2) = couple of numbers < p
def elgamal_decrypt(pri_key, g, p, (m1,m2)):
    return Mod(m2,p)*Mod(m1,p)^(pri_key)
```

If we keep \(r\) as a random number the value of \(e_{k_1}(m)\) may be different each time that we use the function.

To compare results, let us put \(k = 333\). Then for instance for the public key 210904 the message 12345 is encrypted with

\[
elgamal_encrypt(210904, 3, 2^{19} - 1, 12345 )
\]

resulting (29073, 277350).

To decrypt we need to know the private key corresponding to 210904. It is 1000 because \(3^{1000} \equiv 210904 \pmod{2^{19} - 1}\) (check it!).

Now

\[
elgamal_decrypt(1000, 3, 2^{19} - 1, (29073, 277350) )
\]
gives the right answer 12345.

Check that allowing \( k \) to be random the decryption function still works.

Breaking the ElGamal cryptosystem getting the private key \( k_2 \) from the public key \( k_1 \) requires to solve the DLP and this is considered very hard when \( p \) has hundreds of digits.

---

**Quiz:**

Take \( p = 2^{31} - 1 \) and \( g = 7 \). If the public key is 833 287 206 and the ciphertext is (1 457 850 878, 2 110 264 777). What is the plaintext message?

---

**Quiz:**

Take \( p = 2^{31} - 1 \) and \( g = 7 \). If the public key is 1659750829 and the ciphertext is (297629860, 1094924871). What is the plaintext message?

---

**Solutions:**

```
sage: log( Mod( 833287206,2^{31}-1) , Mod(7,2^{31}-1))
2011
sage: elgamal_decrypt(2011,3,2^{31}-1 , (1457850878 ,2110264777 ) )
23571113
```

```
sage: log( Mod( 1659750829 ,2^{31}-1) , Mod(7,2^{31}-1))
1001
sage: elgamal_decrypt(1001,3,2^{31}-1 , (297629860 ,1094924871) )
20110310
```

If we have a long text it is unrealistic to assume that we can encode the message with a single number \( m \in \mathbb{F}_p^* \). It leads to some consideration respect the encoding schemes.