Consider the following variant of the ElGamal encryption scheme.

The public key consists of a prime \( p \), a prime \( q | (p-1), \) an element \( \gamma \in \mathbb{Z}_p^* \) of order \( q \), and an element \( \alpha \in \langle \gamma \rangle \). The secret key consists \( p, q, \gamma, \alpha \), along with with \( x = \log_\gamma \alpha \). We assume, of course, that during key generation, \( x \) is chosen at random (modulo \( q \)). Of course, we assume that \( q \) is large, so in particular, \( q \neq 2 \).

In addition to the above group description, the scheme makes use of two cryptographic hash functions, \( H \) and \( H' \). Assume that \( H \) takes two inputs, an integer \( i \geq 0 \), and an arbitrary bit string \( s \), and outputs a bit string of a fixed length \( \ell \). Assume that \( H' \) that maps pairs \((s, t)\) of arbitrary bit strings to \( \ell \)-bit strings.

The message space of the encryption scheme consists of sequences \( m_1 m_2 \cdots m_k \) of \( \ell \)-bit blocks; i.e., each \( m_i \) is an \( \ell \)-bit string, and \( k \geq 0 \) is arbitrary, and may vary from message to message.

The encryption algorithm runs as follows. To encrypt \( m_1 m_2 \cdots m_k \), the algorithm chooses \( r \in \mathbb{Z}_q \) at random, and computes \( \tilde{\gamma} = \gamma^r \) and \( \tilde{\alpha} = \alpha^r \). It then computes \( c_i = H(i, \tilde{\alpha}) \) for \( 0 \leq i \leq k \), and then \( c'_i = m_i \oplus c_i \) for \( 1 \leq i \leq k \), and \( t = H'(c_0, c'_1 c'_2 \cdots c'_k) \). The ciphertext is \((\tilde{\gamma}, c'_1 c'_2 \cdots c'_k, t)\).

To decrypt a ciphertext \((\tilde{\gamma}, c'_1 c'_2 \cdots c'_k, t)\), the decryption algorithm does the following. First, it checks that \( \tilde{\gamma} \) properly encodes an element of \( \mathbb{Z}_p^* \) and that \( \tilde{\gamma}^q = 1 \); if not, it outputs "error" and halts. It then computes \( \tilde{\alpha} = \tilde{\gamma}^x \). It then computes \( c_i = H(i, \tilde{\alpha}) \) for \( 0 \leq i \leq k \), and then \( m_i = c'_i \oplus c_i \) for \( 1 \leq i \leq k \), and \( t' = H'(c_0, c'_1 c'_2 \cdots c'_k) \). If \( t = t' \), it outputs \( m_1 \cdots m_k \); otherwise, it outputs "error."

1. Show that if we drop the requirement that \( q \) is prime, and if \( q \) is in fact divisible by a small prime, then the scheme is not secure against adaptive chosen ciphertext attack.

2. Show that if the decryption algorithm omits the test that \( \tilde{\gamma}^q = 1 \), then the scheme is not secure against adaptive chosen ciphertext attack.

In addition, show that if \( p - 1 \) is divisible by a number of small primes \( p_1, \ldots, p_w \), such that \( q \mid P \) and \( q/P \) is small, where \( P = \prod_{i=1}^w p_i \), then using a chosen ciphertext attack, an attacker may efficiently recover the secret key \( x \)

3. Suppose we modify the encryption algorithm so that we compute \( c'_i = m_i \oplus c_{i-1} \) for \( 1 \leq i \leq k \), and \( t = H'(c_k, c'_1 c'_2 \cdots c'_k) \), and make a corresponding modification to the decryption algorithm. Show that the resulting encryption scheme is not secure against adaptive chosen ciphertext attack.

4. Show that if we model \( H \) and \( H' \) as (independent) random oracles, an if \( \ell \) is sufficiently large, then the scheme is secure against adaptive chosen ciphertext under the following assumption: for random \( u, v \in \mathbb{Z}_q \), it is hard to compute \( \gamma^u \) given \( \gamma^u \) and \( \gamma^v \), and given access to a special oracle. The oracle takes as input \( \gamma^a, \gamma^b, \gamma^c \), for \( a, b, c \in \mathbb{Z}_q \), and returns 1 if \( c = ab \), and 0 if \( c \neq ab \).