

0368.3049.01 Introduction to Modern Cryptography TAU, Fall 2001
Assignment #4, due before class on Feb. 6, 2002

This assignment contains one “dry” problems and one “semi-dry” one. The answers to the “wet” parts of the latter should be given as a combination of written text and the output of an XMAPLE session. The best 3 out of 4 homework assignments will be taken into account when we compute your grade.

Problem 1: Weak Randomness and ElGamal (This problem is taken verbatim from Ron Rivest Crypto course at MIT.)

Recall that the ElGamal signature scheme is randomized. Obviously the key generation involves randomization. But unlike the signature generation process in RSA, the ElGamal signature generation process relies on a source of good randomness too. When this source of randomness is predictable, the security of ElGamal can break down.

We repeat for your convenience the ElGamal signature scheme (discussed in Lecture 10 of our course).

Key generation:

1. Generate a large prime p and a generator α of the multiplicative group Z_p^*
2. Select a random integer a such that $1 \leq a \leq p - 2$.
3. Compute $y = \alpha^a \pmod{p}$
4. The public key is (p, α, y) and the private key is a .

To sign a message m :

1. Select a *random secret* integer k such that $1 \leq k \leq p - 2$ and $\gcd(k, p - 1) = 1$.
2. Compute $r = \alpha^k \pmod{p}$.
3. Compute $k^{-1} \pmod{p - 1}$.
4. Compute $s = k^{-1}\{h(m) - ar\} \pmod{p - 1}$.
5. Output the pair (r, s) as the signature of m .

To verify a signature (r, s) on m :

1. Obtain the public key (p, α, y) .
2. Verify that $1 \leq r \leq p - 1$. Otherwise reject the signature.
3. Compute $v_1 = y^r r^s \pmod{p}$.
4. Compute $h(m)$ and $v_2 = \alpha^{h(m)} \pmod{p}$.
5. Accept the signature iff $v_1 = v_2$.

(a) Reusing k

Explain how to recover the secret key a when the signer reuses the same k for two different messages. Write an equation for a . That is, extract the private key from the public key and two (message, signature) pairs. State your assumptions.

(b) Generating k with a linear congruential number generator

Louis Reasoner was unable to afford the advanced Lava Lamp random number generator – nor would it fit in his wallet-sized smartcard. Instead, he seeds a linear congruential number generator¹ with a truly random number, k_0 . During the signature of the i th message in Louis' scheme, $k =$

1. If $i = 0$, return a truly random number k_0 where $3 \leq k_0 \leq p - 2$ and $\gcd(k_0, p - 1) = 1$.
2. Otherwise let $k = k_{i-1}$.
3. Do $k \leftarrow k + z \pmod{p - 1}$ until $\gcd(k, p - 1) = 1$.
4. Remember $k_i = k$ and return k .

z is a secret, truly random even number that is the same for every signature generation. This LCNG remembers its output for its next execution. Assume that p is a “safe prime” in that $p = 2q + 1$ where q is prime.

Explain how an adversary can break the system after seeing many sequential (message, signature) pairs. Write an equation for a . Note that your attack need not work 100% of the time, but it should work most of the time assuming that certain values are invertible $\pmod{p - 1}$. State your assumptions about which values must be invertible and any other conditions for your attack to work. Discuss how many (message, signature) pairs you will need to break the scheme.

Problem 2: Shamir Secret Sharing and ElGamal Encryption

In this problem we will examine how to construct a t -out-of- n threshold ElGammal encryption scheme. We want to combine Shamir t -out-of- n secret sharing scheme and deal shares d_1, \dots, d_n to n participants. Given the encryption (y_1, y_2) of a message x , any subset of t participants should be able to find x *without* explicitly recovering the secret decryption key d , but any subset of $t - 1$ participants cannot efficiently recover the message x . We will use a variant of ElGamal encryption so that operations are done in a subgroup of Z_p^* of large prime order.

We begin by explaining the variant of ElGamal encryption scheme (the original scheme was discussed in Lecture 9 of the course).

Key generation:

1. Generate a large prime $p = 2q + 1$ where q is prime, and a generator g of the multiplicative group Z_p^* . Let $\alpha = g^2 \pmod{p}$. It is easy to see that α is a generator of QR , the group of quadratic residues in Z_p^* . The group QR is of order q .
2. Select a random integer d such that $1 < d < q$.
3. Compute $z = \alpha^d \pmod{p}$
4. The public key is (p, α, z) and the private key is d .

¹Thousands of Web sites use LCNGs to generate cookie-based Session IDs on the Web. Is this a good idea?

To encrypt a message $x \in QR$:

1. Select a *random secret* integer k such that $1 < k \leq q - 1$.
2. Compute $y_1 = \alpha^k \pmod{p}$.
3. Compute $y_2 = x \cdot z^k \pmod{p}$.
4. Output the pair (y_1, y_2) as the encryption of x .

To decrypt (y_1, y_2) :

1. Compute $v_1 = y_1^d \pmod{p}$.
2. Recover $x = v_1^{-1} y_2 \pmod{p}$.

(a) Threshold ElGamal

Explain how to how to construct a t -out-of- n threshold ElGamal encryption scheme, using Shamir's t -out-of- n secret sharing as a building block.

(b) Briefly explain why the modification is needed.

(c) Implementing ElGamal Encryption

Implement the variant of ElGamal encryption, using a prime p which should be of the form $p = 2q + 1$ and in the range $12 < p < 28$ (you can use the `length` command in Maple to demonstrate p 's length). Type the secret and public keys. For two different $x_1, x_2 \in QR$ show the process of encrypting and decrypting (use different random k in the two cases).

(d) Implementing Shamir Secret Sharing

Implement an instance of Shamir 3-out-of-8 Secret Sharing, with secrets in the range Z_q for q from part (c). Type the 8 shares of two different secrets and how they were derived. Reconstruct the first secret with the shares of parties 1,5,7 and of the second secret with shares of parties 2,3,8. You can certainly use Maple command for polynomial interpolation.

(e) Implementing 3-out-of-8 ElGamal Encryption

Combine (c) and (d) to a working El-Gammal threshold scheme. Show what shares of the secret key d each of the 8 participants gets, and how they are produced. Encrypt the same x_1 from part (c) and show how parties 1,5,7 decrypt it. Do the same for x_2 with parties 2,3,8.