Crypto Engineering Midterm Exam

April, 2023

Student Name :

Student ID :

Department :

For the following Questions, please write down your answers with explanations

# Question 1 :

Let us see what goes wrong when a stream cipher key is used more than once. Below are eleven hex-encoded ciphertexts that are the result of encrypting eleven plaintexts with a stream cipher, all with the same stream cipher key. Your goal is to decrypt the last ciphertext, and submit the secret message within it as solution.

Hint: XOR the ciphertexts together, and consider what happens when a space is XORed with a character in [a-zA-Z].

## Ciphertext #1 :

315c4eeaa8b5f8aaf9174145bf43e1784b8fa00dc71d885a804e5ee9fa40b16349c146fb778cdf2d3aff021dfff5b403b510d0d0455468aeb98622b137dae857553ccd8883a7bc37520e06e515d22c954eba5025b8cc57ee59418ce7dc6bc41556bdb36bbca3e8774301fbcaa3b83b220809560987815f65286764703de0f3d524400a19b159610b11ef3e

## Ciphertext #2 :

234c02ecbbfbafa3ed18510abd11fa724fcda2018a1a8342cf064bbde548b12b07df44ba7191d9606ef4081ffde5ad46a5069d9f7f543bedb9c861bf29c7e205132eda9382b0bc2c5c4b45f919cf3a9f1cb74151f6d551f4480c82b2cb24cc5b028aa76eb7b4ab24171ab3cdadb8356f

## Ciphertext #3 :

32510ba9a7b2bba9b8005d43a304b5714cc0bb0c8a34884dd91304b8ad40b62b07df44ba6e9d8a2368e51d04e0e7b207b70b9b8261112bacb6c866a232dfe257527dc29398f5f3251a0d47e503c66e935de81230b59b7afb5f41afa8d661cb

## Ciphertext #4 :

32510ba9aab2a8a4fd06414fb517b5605cc0aa0dc91a8908c2064ba8ad5ea06a029056f47a8ad3306ef5021eafe1ac01a81197847a5c68a1b78769a37bc8f4575432c198ccb4ef63590256e305cd3a9544ee4160ead45aef520489e7da7d835402bca670bda8eb775200b8dabbba246b130f040d8ec6447e2c767f3d30ed81ea2e4c1404e1315a1010e7229be6636aaa

## Ciphertext #5 :

3f561ba9adb4b6ebec54424ba317b564418fac0dd35f8c08d31a1fe9e24fe56808c213f17c81d9607cee021dafe1e001b21ade877a5e68bea88d61b93ac5ee0d562e8e9582f5ef375f0a4ae20ed86e935de81230b59b73fb4302cd95d770c65b40aaa065f2a5e33a5a0bb5dcaba43722130f042f8ec85b7c2070

## Ciphertext #6 :

32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd2061bbde24eb76a19d84aba34d8de287be84d07e7e9a30ee714979c7e1123a8bd9822a33ecaf512472e8e8f8db3f9635c1949e640c621854eba0d79eccf52ff111284b4cc61d11902aebc66f2b2e436434eacc0aba938220b084800c2ca4e693522643573b2c4ce35050b0cf774201f0fe52ac9f26d71b6cf61a711cc229f77ace7aa88a2f19983122b11be87a59c355d25f8e4

## Ciphertext #7 :

32510bfbacfbb9befd54415da243e1695ecabd58c519cd4bd90f1fa6ea5ba47b01c909ba7696cf606ef40c04afe1ac0aa8148dd066592ded9f8774b529c7ea125d298e8883f5e9305f4b44f915cb2bd05af51373fd9b4af511039fa2d96f83414aaaf261bda2e97b170fb5cce2a53e675c154c0d9681596934777e2275b381ce2e40582afe67650b13e72287ff2270abcf73bb028932836fbdecfecee0a3b894473c1bbeb6b4913a536ce4f9b13f1efff71ea313c8661dd9a4ce

## Ciphertext #8 :

315c4eeaa8b5f8bffd11155ea506b56041c6a00c8a08854dd21a4bbde54ce56801d943ba708b8a3574f40c00fff9e00fa1439fd0654327a3bfc860b92f89ee04132ecb9298f5fd2d5e4b45e40ecc3b9d59e9417df7c95bba410e9aa2ca24c5474da2f276baa3ac325918b2daada43d6712150441c2e04f6565517f317da9d3

## Ciphertext #9 :

271946f9bbb2aeadec111841a81abc300ecaa01bd8069d5cc91005e9fe4aad6e04d513e96d99de2569bc5e50eeeca709b50a8a987f4264edb6896fb537d0a716132ddc938fb0f836480e06ed0fcd6e9759f40462f9cf57f4564186a2c1778f1543efa270bda5e933421cbe88a4a52222190f471e9bd15f652b653b7071aec59a2705081ffe72651d08f822c9ed6d76e48b63ab15d0208573a7eef027

## Ciphertext #10 :

466d06ece998b7a2fb1d464fed2ced7641ddaa3cc31c9941cf110abbf409ed39598005b3399ccfafb61d0315fca0a314be138a9f32503bedac8067f03adbf3575c3b8edc9ba7f537530541ab0f9f3cd04ff50d66f1d559ba520e89a2cb2a83

## Target Ciphertext (decrypt this one) :

32510ba9babebbbefd001547a810e67149caee11d945cd7fc81a05e9f85aac650e9052ba6a8cd8257bf14d13e6f0a803b54fde9e77472dbff89d71b57bddef121336cb85ccb8f3315f4b52e301d16e9f52f904

For completeness, here is the python2 script used to generate the ciphertexts. (it doesn't matter if you can't read this)

import sys

MSGS = ( --- 11 secret messages --- )

def strxor(a, b): # xor two strings (trims the longer input)

return "".join([chr(ord(x) ^ ord(y)) for (x, y) in zip(a, b)])

def random(size=16):

return open("/dev/urandom").read(size)

def encrypt(key, msg):

c = strxor(key, msg)

print

print c.encode('hex')

return c

def main():

key = random(1024)

ciphertexts = [encrypt(key, msg) for msg in MSGS]

# Question 2 :

Suppose you are told that the one time pad encryption of the message “attack at dawn” is “09e1c5f70a65ac519458e7e53f36” (the plaintext letters are encoded as 8-bit ASCII and the given ciphertext is written in hex). What would be the one time pad encryption of the message “attack at dusk” under the same OTP key?

# Question 3 :

The movie industry wants to protect digital content distributed on DVD’s. We develop a variant of a method used to protect Blu-ray disks called AACS.

Suppose there are at most a total of n DVD players in the world (e.g. n = ). We view these n players as the leaves of a binary tree of height n. Each node in this binary tree contains an AES key . These keys are kept secret from consumers and are fixed for all time. At manufacturing time each DVD player assigned a serial number i [0, n - 1]. Consider the set of nodes along the path from the root to leaf number i in the binary tree. The manufacturer of the DVD player embeds in player number i the keys associated with the nodes in the set . A DVD movie m is encrypted as where k is a random AES key called a content-key and is the key associated with the root of the tree. Since all DVD players have the key all players can decrypt the movie m. We refer to as the header and as the body. In what follows the DVD header may contain multiple ciphertexts which each ciphertext is the encryption of the content-key k under some key in the binary tree.

Suppose the keys embedded in DVD player number r are exposed by hackers and published on the Internet. In this problem we show that when the movie industry distributes a new DVD movie, they can encrypt the contents of the DVD using a slightly larger header (containing about keys) so that all DVD players, except for player number r, can decrypt the movie. In effect, the movie industry disables player number r without affecting other players.

As shown below, consider a tree with n = 16 leaves. Suppose the leaf node labeled 25 corresponds to an exposed DVD player key. Check the set of keys below under which to encrypt the key k so that every player other than player 25 can decrypt the DVD. Only four keys are needed.

一張含有 圖表 的圖片

自動產生的描述

(A) 14 (B) 28 (C) 11 (D) 4 (E) 6 (F) 2 (G) 26 (H) 1

# Question 4 :

Continuing with the previous question, if there are n DVD players, what is the number of keys under which the content key k must be encrypted if exactly one DVD player’s key needs to be revoked?

(A) 2 (B) n-1 (C) n (D) n/2 (E)

# Question 5 :

In the following let p be a prime. The set is a group with respect to addition modulo (i.e. every element in has an inverse such that . The set is a group with respect to multiplication modulo (i.e. every element in has an inverse such that .

**Another cipher with perfect secrecy**. Consider the following cipher.

Let be the message space, the key space and the ciphertext space.

Alice and Bob share a key uniformly chosen at random. To send a message to Bob, Alice computes the ciphertext .

1. Prove that this cipher provides **Perfect Secrecy** using the criterium we proved in class.

2. Why one-time pad are **Perfect Secrecy** and also **Semantic Secu**re?

3. Is the use of one-time pads susceptible to statistical analysis (especially if it is known that the plaintext is in American English)?

4. Did public-key encryption scheme provide Perfect Secrecy? We assume there is a public-key encryption scheme (KeyGen, Enc, Dec) with perfect correctness (i.e., for all messages M and valid key-pairs (PK, SK), we have ((M)) = M).

# Question 6 :

**Predicting generators.** Consider the following *congruential generator*. It uses constants . The seed is a value . The value generated is computed as .

The sequence output by the generator is  Assume that an attacker knows and witness the sequence.

1. Prove that after a short prefix (i.e. a few of the values ’s) the attacker is able to predict the rest of the sequence (i.e. the rest of the ’s).

2. What does this say about the security of using the congruential generator as the keystream generator for a stream cipher?

3. If an attacker knows constants and . How many output bits did the attacker to know to rest sequences.

4. However, If an attacker knows but know nothing about constants . In this case, How many output bits did the attacker need to know to recover the rest sequence?

# Question 7 :

In standard RSA the modulus N is a product of two distinct primes. Suppose we choose the modulus so that it is a product of three distinct primes, namely N = pqr. Given an exponent e relatively prime to we can derive the secret key as . The public key (N, e) and secret key (N, d) work as before. What is when N is a product of three distinct primes?

(A)

(B)

(C)

(D)

# Question 8 :

Suppose we choose the modulus so that it is a product of three distinct primes, namely N = 105. Given an encryption key is 13 which is co-prime to . Please find the secret key as using Extended Euclidean Algorithm.

# Question 9 :

An attacker intercepts the following ciphertext (hex encoded) :

20814804c1767293b99f1d9cab3bc3e7ac1e37bfb15599e5f40eef805488281d

He knows that the plaintext is the ASCII encoding of the message “Pay Bob 100$” (excluding the quotes). He also knows that the cipher used is CBC encryption with a random IV using AES as the underlying block cipher. Show that the attacker can change the ciphertext so that it will decrypt to “Pay Bob 500$”. What is the resulting ciphertext (hex encoded)? This shows that CBC provides no integrity.

# Question 10 :

Let G be a finite cyclic group (e.g. ) with generator g. Suppose the Diffie-Hellman function is difficult to compute in G. Which of the following functions is also difficult to compute :

As usual, identify the f below for which the contra-positive holds :

if is easy to compute then so is .

If you can show that then it will follow that if is hard to compute in G then so must be f.

(A)

(B)

(C)

(D)