Tham Jit

Qn1: j2tham

#xor functions

def xor(\*args:str):

    x=0

    for i in args:

        x += int(i)

    return str(x%2)

def xors(a,b):

    out = ''

    for i,j in zip(a,b):

        out += xor(i,j)

    return out

# anaylsis function

def analysep(guess1,guess2,cipherls,plainls):

    sboxmap = [14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7]

    #import files, convert to list

    sum = 0

    for cipher,plain in zip(cipherls,plainls):

        v4\_select1 = xors(cipher[4:8],guess1)

        v4\_select2 = xors(cipher[12:],guess2)

        u4select1 = '{0:04b}'.format(sboxmap.index(int(v4\_select1,2)))

        u4select2 = '{0:04b}'.format(sboxmap.index(int(v4\_select2,2)))

        sum += int(xor(u4select1[1],u4select1[3],u4select2[1],u4select2[3],plain[4],plain[6],plain[7]))

    return sum/len(cipherls) - 1/2

with open('a2q1ciphertexts.txt') as cipheraw:

        cipherls = cipheraw.read().splitlines()

        cipheraw.close()

with open('a2q1plaintexts.txt') as plainraw:

        plainls = plainraw.read().splitlines()

        plainraw.close()

#qn 1ai

print("1ai")

carolguess1 = '0111'

carolguess2 = '0110'

print(analysep(carolguess1,carolguess2,cipherls,plainls))

#qn 1b

print('1b')

best = ''

bestbias = 0

for i in range(256):

    guess12 = f'{i:08b}'

    guess1=guess12[:4]

    guess2=guess12[4:]

    bias = analysep(guess1,guess2,cipherls,plainls)

    if abs(bias)>abs(bestbias):

        best = guess12

        bestbias = bias

print(bestbias)

print(best)

1ai) bias = -0.00340 (3.s.f)

1b) bias = -0.0344 (3.s.f), keys = xxxx 0101 xxxx 1110

1c) biases:

Linear approx.:

Theoretical bias

1d)

Qn2: j2tham

#xor functions

def xor(\*args:str):

    x=0

    for i in args:

        x += int(i)

    return str(x%2)

def xors(a,b):

    out = ''

    for i,j in zip(a,b):

        out += xor(i,j)

    return out

# anaylsis function

def analysep(guess1,guess2,cipherls,plainls):

    sboxmap = [14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7]

    #import files, convert to list

    sum = 0

    for cipher,plain in zip(cipherls,plainls):

        v4\_select1 = xors(cipher[4:8],guess1)

        v4\_select2 = xors(cipher[12:],guess2)

        u4select1 = '{0:04b}'.format(sboxmap.index(int(v4\_select1,2)))

        u4select2 = '{0:04b}'.format(sboxmap.index(int(v4\_select2,2)))

        sum += int(xor(u4select1[1],u4select1[3],u4select2[1],u4select2[3],plain[4],plain[6],plain[7]))

    return sum/len(cipherls) - 1/2

with open('a2q1ciphertexts.txt') as cipheraw:

        cipherls = cipheraw.read().splitlines()

        cipheraw.close()

with open('a2q1plaintexts.txt') as plainraw:

        plainls = plainraw.read().splitlines()

        plainraw.close()

print('2a')

with open('a2q2ciphertexts.txt') as raw:

    ls = raw.read().splitlines()

    raw.close()

cipherls2 = []

plainls2 = []

for i in ls:

    x = i.split(',')

    plainls2.append(x[0])

    plainls2.append(x[1])

    cipherls2.append(x[1])

    cipherls2.append(x[2])

for i in range(256):

    guess12 = f'{i:08b}'

    guess1=guess12[:4]

    guess2=guess12[4:]

    bias = analysep(guess1,guess2,cipherls2,plainls2)

    if abs(bias)>abs(bestbias):

        best = guess12

        bestbias = bias

print(bestbias)

print(best)

2a) bias = -0.0344 (3.s.f), keys = xxxx 0101 xxxx 1110

2b)

3a) for ,

Therefore, will produce

3b)

Assuming IV is , encrypt . If result ,

3c)

1. Query encryption of to get value of first bit of m encrypted

4a)

Suppose H is not preimage resistant. compute for some .

Compute a preimage whereby . Since H is uniform, there is a high probability that. Hence, given an input , there is another where . This means is not second preimage resistant.

Therefore, is preimage resistant for messages of length , where

4b)

Let G: be a second preimage resistant hash function,

Let as:

Since G is second preimage resistant, then H is also second preimage resistant for length n and below. In the case of if , H is also second preimage resistant as the hash value has a one-to-one mapping with the value.

However, given a message of length , it would be computationally feasible to find an input by removing the most significant bit

5a) the operator has the same properties as . This can be seen from the truth table of :

|  |  |  |
| --- | --- | --- |
| x | y | result |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

Which is identical to

Let

This would mean and

Since is commutative, ⊙ is commutative.

x’ ⊙ x’’ = x’’⊙ x’

f(x’ ⊙ x’’) = f(x’’⊙ x’)

h(x1) = h(x2), where x1 =/= x2

Hence, h is not second preimage resistant

5b)

and , .

Since is preimage resistant, cannot be found where .

Therefore, the preimage of h function cannot be determined as well, since this preimage is derived from z. Thus, h is preimage resistant for messages of length 2m.

Acknowlegments:

Discussed with Louis, Sean, Min Htet