SRI TEJA KALE
G01501801
CS 587

ASSIGNMENT-1

Exercise 1. Shift Cipher [10 points] Consider the shift cipher as we discussed it in class.

Key space $\mathcal{K} = \{0, 1, \ldots, 25\}$.

Suppose that we are given the following message space distribution, M:
$\Pr[M = \text{'bye'}] = 0.1$
$\Pr[M = \text{'yes'}] = 0.5$
$\Pr[M = \text{'now'}] = 0.4$

Answer the following questions and make sure that you show your work.

(a) **(5 Points)** Compute the probability:
$\Pr[M = \text{'now'} | C = \text{'baa'}]$

(b) **(5 Points)** Compute the probability:
$\Pr[M = \text{'yes'} | C = \text{'zft'}]$

A **shift cipher (Caesar cipher)** encrypts a message by shifting each letter in the plaintext by a fixed key k, modulo 26:

Ci=(Pi+k) mod26

K=(c-p) mod26

(a) Compute Pr[M='now'|C='baa']

Step 1: Check if 'now' can encrypt to 'baa'

We calculate the key k for each letter:

n→b : k=('b'-'n')mod 26 = (1−13)mod 26=(−12)mod 26=14

o→a : k=('a'-'o')mod 26 =(0−14)mod 26=(−14)mod 26=12

w→a : k=('a'-'w')mod 26 =(0−22)mod 26=(−22)mod 26=4

we obtained various values of k for various letters in message, but in shift cipher identical k is to be utilized so there is no k for message 'now' to shift to cipher 'baa'.
Thus, Pr[M = 'now'IC = 'baa']=0, as there is no key that is valid and encrypts the message 'now' to 'baa'.

(b) Compute: Pr[M = 'yes'|C = 'zft']

Step 1: Check if 'yes' can encrypt to 'zft'

We determine the key k for each letter:

y→z : k=('z'-'y')mod 26= (25−24)mod 26=1

e→f : k=('f'-'e')mod 26= (5−4)mod 26=1

s→t : k=('t'-'s')mod 26= (19−18)mod 26=1

Since the same k=1works for all letters, the shift is valid and encrypts the message 'yes' to 'zft'.

Using bayes theorem:

Pr[M='yes'|C= 'zft']=Pr[C= 'zft'|M= 'yes'] * Pr[M='yes'] /Pr[C= 'zft']

Since there is only one possible value of k that encrypts the message 'yes' to cipher 'zft'

Pr[C= ‘zft’|M= ‘yes’] = 1/26

Given, pr[M=’yes’]=0.5

Step 2: Compute Pr[C=’zft’]

We check if any other message could also encrypt to 'zft':

Checking 'bye' → 'zft'

b→z : k=(‘z’-‘b’)mod 26= (25−1)mod 26=24

y→f : k=(‘f’-‘y’)mod 26= (5−24)mod 26=7

Since k is inconsistent, 'bye' cannot encrypt to 'zft'

Pr[C='zft'|M='bye']=0

Checking 'now' → 'zft'

n→z : k=(‘z’-‘n’)mod 26= (25−13)mod 26=12

o→f : k=(‘f’-‘o’)mod 26= (5−14)mod 26=17

Since k is inconsistent, 'now' cannot encrypt to 'zft'

Pr[C='zft'|M='now'] =0

Thus, the total probability of receiving 'zft' as ciphertext is:

Pr[C= ‘zft’|M= ‘bye’] * Pr[M= ‘bye’] + Pr[C= ‘zft’|M= 'yes'] * Pr[M=’yes’] + Pr[C= ‘zft’|M= ‘now’] * Pr[M= 'now']

Pr[C='zft']=(0×0.1)+(1/26×0.5)+(0×0.4)=0.5/26

Step 3: Compute Pr[M=’yes’|C=’zft’]

Using Bayes’ Theorem:

Pr[M= 'yes'IC= 'zft'] = Pr[C= ‘zft’|M= 'yes'] * Pr[M= 'yes'] /Pr[C=’zft’]

Pr[M='yes'|C='zft']=(1/26×0.5)/(0.5/26)=1

Therefore, Pr[M='yes'|C='zft']=1

Hence, final results are as follows :

Pr[M='now'|C='baa']=0

Pr[M='yes'|C='zft']=1

Exercise 2. OTP [25 points] Recall the One Time Pad encryption system:

Message space $\mathcal{M}$: $\{0,1\}^\ell$
Keyspace $\mathcal{K}$: $\{0,1\}^\ell$

Ciphertext space $\mathcal{C}$: $\{0,1\}^\ell$

Gen : $k = k_1 \dots k_\ell \leftarrow \mathcal{K}$
Enc(k, m) : $c_i = m_i \oplus k_i$
Dec(k, c) : $m_i = c_i \oplus k_i$

Assume that an adversary knows that $\Pr[m = 010] = 0.5$ and $\Pr[m = 011] = 0.5$. The adversary then observes a ciphertext $c = 010$. Compute the following probabilities:

(a) Compute Pr[m=010|c=010]

Step 1: Understanding OTP Decryption

The One-Time Pad (OTP) encryption follows the rule:

$c=m \oplus k$

Decryption is simply:

$m=c \oplus k$

Given the observed ciphertext c=010, we analyse how likely each message is.

Step 2: Finding Possible Keys

For each possible message m, there is exactly one corresponding key:

Given, m=010 and c=010

Now, to find the possible k values: $k= c \oplus m$

- If m=010 then $k=010 \oplus 010=000$

Total number of possible values of key=2^no.of bits=2^3=8

There is only one possible key that can encrypt the message 010 to produce the ciphertext 010.

S0, pr[c=010|m=010] =1/8

Step 3: Applying Bayes' Theorem

Using Bayes' theorem:

Pr[m=010|c=010]=Pr[c=010|m=010]*Pr[m=010]/Pr[c=010]

We calculate:

Pr[c=010]=Pr[c=010|m=010]*Pr[m=010]+Pr[c=010|m=011]*Pr[m=011]

=(1/8×0.5)+(1/8×0.5)=1/16+1/16=1/8

Thus:

Pr[m=010|c=010]=((1/8)×0.5)/(1/8)=0.5

Hence,final result:

Pr[m=010|c=010]=0.5

(b) Compute Pr[m=011|c=010]

Step 1: Understanding OTP Decryption

The One-Time Pad (OTP) encryption follows the rule:

$c=m \oplus k$

we can derive, $k= c \oplus m$

Step 2: Finding Possible Keys

For each possible message m, there is exactly one corresponding key:

Given, m=011 and c=010

Now, to find the possible k values: $k= c \oplus m$

- If m=011 then $k=010 \oplus 011=001$

Total number of possible values of key=2^no.of bits=2^3=8

S0, pr[c=010|m=011] =1/8

Step 3: Applying Bayes' Theorem

Using Bayes' theorem:

Pr[m=011|c=010]=Pr[c=010|m=011]*Pr[m=011]/Pr[c=010]

We calculate:

Pr[c=010]=Pr[c=010|m=010]*Pr[m=010]+Pr[c=010|m=011]*Pr[m=011]

=(1/8×0.5)+(1/8×0.5)=1/16+1/16=1/8

Thus:

Pr[m=011|c=010]=((1/8)×0.5)/(1/8)=0.5

Hence,final result:

Pr[m=011|c=010]=0.5

Alice is using one-time pad and notices that when her key is the all-zeroes string $k = 0^\lambda$, then $Enc(k, m) = m$ and her message is sent in the clear! To avoid this problem, she decides to modify Gen to exclude the all-zeroes key, i.e., choose a key uniformly from $\{0,1\}^\ell \backslash \{0\}^\ell$. In this way, she guarantees that her plaintext is never sent in the clear.

(c)Is the modified cryptosystem still perfectly secret?

Perfect secrecy requires that for any two messages m1 and m2, and any ciphertext c:

Pr[c|m1] =Pr[c|m2]

By removing the all-zero key, the remaining keys are chosen from {0,1}^ℓ\{0^ℓ}, which means that the distribution of ciphertexts is no longer perfectly uniform. This introduces a small bias in the probability distribution of ciphertexts.

Since the probability distributions of ciphertexts differ for different plaintexts, the system is no longer perfectly secret.

The modified OTP is not perfectly secret because the probability distributions are slightly altered by excluding the all-zero key.

Bob decides to modify the OTP so that the key gen algorithm picks two keys $k_1, k_2 \in \{0,1\}^\ell$ and then the encryption algorithm computes $c = k_1\ XOR(k_2\ XOR\ m)$.

(d) How many keys are there in the modified OTP system?

Bob's modification involves selecting two independent keys k1,k2 from {0,1}^l, so the total keyspace size is:
|K|=2^ℓ×2^ℓ=2^2ℓ

Final Answer:

Total keys=2^2l

The strength of the modified cryptosystem remains equivalent to 2^ℓ because of the way encryption is structured. The ciphertext is computed as:

c=k1⊕(k2⊕m)

Rewriting this, we get:

c=(k1⊕k2)⊕m

If we define k=k1⊕k2, then the encryption simplifies to:

c=k⊕m

This is identical to the standard one-time pad (OTP), where a single random key is XORed with the plaintext. Therefore, from a security perspective, the system behaves just like OTP, maintaining its strength at 2^ℓ

However, while the number of possible keys in the key space is 2^2l due to the use of two independent keys, the effective security remains at 2^ℓ, as the system still follows the fundamental OTP structure.

(e)Prove that the resulting cryptosystem is still perfectly secret.

Proving Perfect Secrecy of the Encryption Scheme:

We define an encryption scheme Π=(Gen,Enc,Dec)with a message space M. According to the definition of perfect secrecy, the scheme must satisfy:

Pr[M=m|C=c]=Pr[M=m],∀m∈M,c∈C, where Pr[C=c]>0.

This means that observing the ciphertext c should not reveal any additional information about the original message m.

Step 1: Understanding the Encryption Scheme

The given encryption scheme encrypts a message m as:

c=k1⊕(k2⊕m)

Rewriting this, we define:

c=(k1⊕k2)⊕m

Let k=k1⊕k2, then this simplifies to:

c=k⊕m

which is structurally identical to the one-time pad (OTP).

Step 2: Calculating Probabilities

Since the keys k1and k2 are chosen uniformly and independently from {0,1}^, the number of possible key pairs (k1,k2)is: 2^2l

For any fixed message m, there are exactly 2^ℓ key pairs that can produce a given ciphertext c.

Thus, the probability of obtaining c from m is:

Pr[C=c|M=m]=Number of valid key pairs / Total possible key pairs=2^ℓ / 2^2ℓ=1/2^l

Using the Law of Total Probability, we find Pr[C=c]:

Pr[C=c]=∑mPr[C=c|M=m]*Pr[M=m]

Since messages are chosen uniformly, we assume Pr[M=m]=1/2^ℓ,leading to :

Pr[C=c]=∑m(1/2^ℓ*1/2^ℓ)=1/2^ℓ·∑mPr[M=m]

Since the sum of all message probabilities equals 1, we get:

Pr[C=c]=1/2^ℓ

Step 3: Checking Perfect Secrecy Condition

Applying Bayes' Theorem:

Pr[M=m|C=c]=Pr[C=c|M=m]*Pr[M=m] / Pr[C=c]

Substituting the values:

Pr[M=m|C=c]=(1/2^ℓ)*(1/2^ℓ)1/2^ℓ=1/2^l

Since Pr[M=m]=1/2^ℓ, we see that:

Pr[M=m|C=c]=Pr[M=m]

which confirms that the encryption scheme satisfies the definition of perfect secrecy.

Since the probability distribution of the message remains unchanged even after observing the ciphertext, the encryption scheme ensures perfect secrecy. In simple terms, knowing the ciphertext does not give an attacker any advantage in guessing the original message, making the system as secure as a traditional one-time pad.

To solve this problem, I took assistance from ChatGPT

Exercise 3. A bad reduction [20 points] Consider the following encryption scheme Π:

Let $G() : \{0,1\}^n \to \{0,1\}^{2n}$ be a secure PRG.

$\mathsf{Gen}(1^n)$: Output a key k of size n bits selected uniformly at random from $\{0,1\}^n$
$\mathsf{Enc}(k,m) : c = m_L \oplus G(k) || m_R \oplus G(k)$, (where $|m| = 4n$ bits and $|m_L| = |m_R| = 2n$ bits, and $||$ denotes string concatenation)
$\mathsf{Dec}(k,c) : m = c_L \oplus G(k) || c_R \oplus G(k)$

This scheme *is not* EAV secure. Your task is to **find the mistake** in the following reduction that attempts to prove EAV security of Π assuming that $G()$ is a secure PRG.

Break down your answer as follows:

(a) (10 points) First prove that the scheme is not EAV secure (i.e. show how an adversary break the EAV game and with what probability it wins).

(b) (10 points) Point out the flaw in the reduction and explain why it is wrong.

(a) Proving That the Scheme Is Not EAV Secure

To prove that the encryption scheme **Π** is not EAV secure, we need to show that an adversary can break the **IND-EAV** (Indistinguishability under Eavesdropping Attack) game with probability significantly greater than **1/2**.

Step 1: Understanding the Encryption Scheme

The encryption function is given as:

c=(mL⊕G(k))||(mR⊕G(k))

where the message m is split into two equal halves, mL and mR, both XORed with the same keystream **G(k)**.

The decryption function reverses this process:

m=(cL⊕G(k))||(cR⊕G(k))

where cL and cR are the two halves of the ciphertext.

Step 2: Adversary's Strategy to Break the Scheme

The adversary selects two messages m0 and m1, both of length **4n bits**, ensuring they differ in a specific way:

$m0=m0L||m0R$, $m1=m1L||m1R$

such that:

$m0L \oplus m0R \neq m1L \oplus m1R$

This is key to distinguishing the two encrypted messages.

The challenger picks a random bit $b \in \{0,1\}$ and encrypts mb:

If $b=0$: $c=(m0L \oplus G(k))||(m0R \oplus G(k))$

If $b=1$: $c=(m1L \oplus G(k))||(m1R \oplus G(k))$

The challenger then sends c to the adversary.

The adversary computes:

$cL \oplus cR=(mbL \oplus G(k)) \oplus (mbR \oplus G(k))$

Since the keystream **G(k)** cancels out, this simplifies to:

$cL \oplus cR=mbL \oplus mbR$

This means that the ciphertext directly leaks the XOR of the two message halves.

The adversary checks which message pair satisfies the equation:

If $cL \oplus cR=m0L \oplus m0R$, the adversary guesses b'=0.

If $cL \oplus cR=m1L \oplus m1R$, the adversary guesses b'=1.

The adversary wins the game with probability 1 because the value $cL \oplus cR$ is entirely determined by the chosen plaintext, independent of the key k.

Thus, the adversary correctly identifies b with probability:

$Pr[\text{Adversary wins}]=1$

Since this is **greater than 1/2**, the encryption scheme **is not EAV secure**.

(b) Point out the flaw in the reduction and explain why it is wrong.

The flaw in the reduction is that it incorrectly assumes the encryption scheme's EAV security depends on the security of the PRG G(k). The real issue is that the same PRG output G(k) is used to XOR both halves of the message, which can be exploited by the adversary without breaking the PRG.

An adversary can simply XOR the two halves of the ciphertext together:

$cL \oplus cR=(mL \oplus G(k)) \oplus (mR \oplus G(k))$

Since G(k) is the same for both halves, it cancels out completely:

$cL \oplus cR=mL \oplus mR$

The reduction assumes that to break the encryption scheme, the adversary must break the PRG G(k), but that is not the case. The adversary is exploiting the fact that the same keystream G(k)G(k)G(k) is used twice (once for each half of the message). This structural flaw allows the adversary to learn information about the plaintext (specifically, the XOR of the two halves of the message), without needing to break the PRG at all.

In other words, the attack does not require any cryptographic breakthrough against the PRG. Instead, it is a direct consequence of the poor design of the encryption scheme, where reusing the same keystream for both halves of the message exposes the XOR of the message halves. Thus, the reduction is flawed because it misattributes the vulnerability to the security of the PRG, when in fact the problem lies in the scheme's misuse of the keystream.

Exercise 4. More Fun with PRGs! [25 points]

(a) Let $G : \{0,1\}^{2n} \to \{0,1\}^{4n}$ be a PRG (for every n). Prove that the following construction *is NOT* a secure PRG.

$G_c(s) = G(1^{|s|}||s)$, where $1^{|s|}$ denotes the all 1 string of length equal to the length of s. Here let $s \in \{0,1\}^n$. Assume n is a multiple of 2.

For your proof, you will need to assume that the following fact holds.

Given Fact: if $H : \{0,1\}^n \to \{0,1\}^{4n}$ is a PRG, then $G(s) = H(s_1, \ldots, s_{n/2})$ is also a PRG (where $s_1, \ldots, s_{n/2}$ are the first $n/2$ bits of S).

(b) Assume that $H : \{0,1\}^n \to \{0,1\}^{2n}$ is a secure PRG, then prove that $G(s_L||s_R) = s_L||H(s_R)$ is also a secure PRG (where $s = s_L||s_R$ and $|s_L| = |s_R|$ (you can assume n is always even).

You can break down your answer as follows:

(1) State the contrapositive you will be proving.
(2) Design the reduction (i.e. how the two distinguishers interact with each other).
(3) Analyze the probability of success of your reduction.

(a)We are given that G: {0,1} ^2n→ {0,1}^4n is a secure PRG.

The modified construction is defined as:

Gc(s)=G(1^n||s)

where s∈ {0,1}^n, and 1^n is the all-ones string of length n. We assume n is a multiple of 2.

Step 1: Applying the Given Fact

We are given the fact that if H:{0,1}^2n→{0,1}^4n is a PRG, then defining G(s)=H(s1,...,sn/2), where s1,...,sn/2 are the first n/2 bits of s, still results in a secure PRG.

From this fact, we conclude that G(s) depends only on the first n bits of its input.

Step 2: Analyzing Gc(s)

The input to G in Gc(s) is (1^n || s).

The first half of this input consists entirely of ones: 1^n.

Applying the given fact, G only depends on its first half of input bits.

Therefore, the output of Gc(s) is: Gc(s)=G(1^n||s) =H(1^n) which is a constant value, independent of s.

Step 3: Constructing a Distinguisher

Since Gc(s) always produces the same output H(1^n) for any input s, we can construct a simple distinguisher D as follows:

Given an input y of length 4n, which may be either:

The output of Gc, or

A truly random string.

Compute z= H(1^n).

If y = z, output 1 (guess that y is from Gc).
Otherwise, output 0 (guess that y is random).

Step 4: Analysing the Success of the Distinguisher

If y is from Gc(s), then by our argument, y= H(1^n) always. Thus, the distinguisher always outputs 1 with probability 1.

If y is a truly random string from {0,1}^4n, the probability that y matches the fixed value H(1^n) is at most 1/2^{4n}.

The advantage of the distinguisher is: |1−1/2^4n| which is clearly non-negligible.

Step 5: Conclusion

Since we have a distinguisher that can efficiently distinguish the output of Gc(s) from a truly random string with overwhelming probability, Gc is not a secure PRG.

The fundamental flaw is that Gc(s) always produces a constant output, which violates the requirement that a PRG should generate indistinguishable pseudorandom outputs for different seeds.

(b)Proof that G(sL||sR)=sL||H(sR) is a Secure PRG

Given that H is a Secure PRG

We aim to prove that if H:{0,1}^n→{0,1}^2n is a secure pseudorandom generator (PRG), then the function

G(sL||sR)=sL||H(sR)

is also a secure PRG, where s=sL||sR and |sL|=|sR|=n/2. We assume that n is always even.

(1) Contrapositive Statement

Instead of proving directly that if H is a secure PRG, then G is also a secure PRG, we prove the contrapositive:

Original statement: If H is a secure PRG, then G is also a secure PRG.

Contrapositive: If G is not a secure PRG, then H is not a secure PRG.

This means that if there exists a distinguisher DG that can distinguish the output of G from a truly random string with non-negligible advantage, then we can construct another distinguisher DH that can distinguish the output of H from a truly random string, contradicting the security of H.

(2) Designing the Reduction

We assume there exists a distinguisher DG that can distinguish the output of G(sL||sR)=sL||H(sR) from a truly random string with a non-negligible advantage. We will use DG to construct a distinguisher DH that can break the security of H.

Reduction Construction

The adversary DH receives an input string y of length 2n, which is either:

The output of H(sR) for some random sR, or

A truly random string from {0,1}^2n.

DH picks a uniformly random sL∈{0,1}^n/2.

DH constructs z=sL||y (which is of length 3n/2).

DH forwards z to DG.

DH outputs whatever DG outputs.

Intuition Behind the Reduction

If y was produced by H(sR), then z=sL||H(sR) follows the same distribution as the output of G(sL||sR).

If y was uniformly random, then z=sL||y is also uniformly random because sL is independent and randomly chosen.

Thus, DH perfectly simulates the two cases that DG tries to distinguish.

(3) Probability Analysis

We analyze the success probability of DH.

By assumption, DG has a non-negligible advantage ϵ, meaning: $|Pr[DG(G(sL||sR))=1]-Pr[DG(U_{3n/2})=1]|\geq\epsilon$.

Since DH perfectly maps H(sR) into the same form as G(sL||sR), the probability that DH correctly distinguishes H(sR) from a random string is exactly the same as DG's probability of success.

That is,

$|Pr[DH(H(sR))=1]-Pr[DH(U_{2n})=1]|\geq\epsilon$.

This contradicts the assumption that H is a secure PRG because it implies there exists an efficient distinguisher for H with non-negligible advantage.

Thus, we conclude that no such efficient DG can exist, meaning G(sL||sR)=sL||H(sR) is also a secure PRG.

We have successfully proved by contrapositive that if H is a secure PRG, then G(sL||sR) must also be a secure PRG. The key idea is that any distinguisher for G can be used to distinguish H, which contradicts the security of H. Hence, G remains pseudorandom.

Exercise 5. CPA Encryption [20 points] In this question you will break CPA security of two encryption schemes (the first built with PRGs and the second built with PRFs).

(a) State the CPA security game .

Consider the following encryption scheme:

Let $G() : \{0,1\}^n \to \{0,1\}^{2n}$ be a secure PRG.

$\mathsf{Gen}(1^n)$: Output a key k of size n bits selected uniformly at random from $\{0,1\}^n$
$\mathsf{Enc}(k,m) : c = m \oplus G(k) \oplus 1^{|2n|}$ (where $1^{|n|}$ is a string of all 1's of size n)
$\mathsf{Dec}(k,c) : m = c \oplus G(k) \oplus 1^{|2n|}$

(a) Show that the above construction is **not** CPA-secure.

The **CPA (Chosen-Plaintext Attack) security game** involves an interaction between an adversary A and a challenger:

Key Generation: The challenger runs Gen(1^n) to generate a secret key k, which remains hidden.

Query Phase: The adversary can request encryptions of arbitrary messages m1,m2, ...and receives ciphertexts c1,c2,...

Challenge Phase: The adversary submits two equal-length messages m0 and m1. The challenger selects a random bit $b\in\{0,1\}$, computes $c*=Enc(k,mb)$, and sends c^* to the adversary.

Additional Queries: The adversary can make further encryption queries.

Guess Phase: The adversary outputs a guess b' for b.

Winning Condition: The adversary wins if b' = b with probability significantly greater than 1/2.

A scheme is **CPA-secure** if all polynomial-time adversaries have at most negligible advantage over random guessing.

(b) Describe an adversary that can break CPA security of above construction.

Encryption Scheme Definition

- Key Generation: k∈{0,1}^n is chosen uniformly at random.
- Encryption: Enc(k,m)=m⊕G(k)⊕1^2n
- Decryption: Dec(k,c)=c⊕G(k)⊕1^2n

Adversary's Strategy

1. The adversary chooses two messages:
    - m0=0^2n (all zeros).
    - m1=1^2n (all ones).
2. The challenger encrypts mb and sends the ciphertext:

    C^∗=mb⊕G(k)⊕1^2n
3. The adversary computes:

    C^∗⊕m0=mb⊕G(k)⊕1^2n⊕0^2n=mb⊕G(k)⊕1^2n
4. Distinguishing Between m0 and m1:
    - If b=0(i.e., mb=m0=0^2n), then:

      C^∗=G(k)⊕1^2n
    - If b=1 (i.e., mb=m1=1^2n), then:

      C^∗=G(k)

5.Decision Rule**:**

- The adversary checks if all bits of c^* are complemented compared to another encryption query.
- If c^* has all bits flipped compared to another encryption, then b=0, otherwise b=1.

Since this strategy always reveals bbb correctly, the adversary breaks CPA security.

(c) Explicitly state the probability of success of this adversary

Now consider another construction:
Let F be a fixed-length PRF.
$\mathsf{Gen}(1^n) : k \leftarrow \{0,1\}^n$
$\mathsf{Enc}(k,m) : F_k(0^n) \oplus m$
$\mathsf{Dec}(k,c) : F_k(0^n) \oplus c$

The adversary can determine b **with certainty**, meaning:

Pr[b'=b]=1.

Since this probability is significantly greater than 1/2, the scheme is **not CPA-secure**.

(d) Describe an adversary that can break CPA security of the above construction

Encryption Scheme Definition

- **Key Generation:** k∈{0,1}^n is chosen uniformly at random.
- **Encryption:** Enc(k,m)=Fk(0^n)⊕m
- **Decryption:** Dec(k,c)=Fk(0^n)⊕c

Adversary's Strategy

1. The adversary queries for encryption of m0 and receives:

   c0=Fk(0^n) ⊕m0

2. The adversary queries for encryption of m1 and receives:

   c1=Fk(0^n) ⊕m1

3. The adversary submits (m0, m1) as challenge messages. The challenger picks b and returns: c^∗=Fk(0^n) ⊕mb

4. The adversary computes:

   C^∗⊕c0=(Fk(0^n)⊕mb)⊕(Fk(0^n)⊕m0)=mb⊕m0.

5. If c^∗⊕c0=0^n, then b=0, otherwise b=1.

Since this attack works with certainty, the encryption scheme is **not CPA-secure**.

(e) Explicitly state the probability of success of this adversary.

Since the adversary can always determine b, the probability of success is:

Pr[b'=b]=1.

This probability is significantly greater than 1/2, confirming that the scheme is **not CPA-secure**.

Both encryption schemes fail CPA security due to their deterministic nature. The PRG-based scheme allows bitwise comparisons due to a fixed mask, while the PRF-based scheme reuses the same PRF output, making ciphertexts easily distinguishable. In both cases, an adversary can distinguish encryptions of different messages with certainty, proving they are not CPA-secure.

Exercise 6. [20 points if 587]
Let $G : \{0,1\}^n \to \{0,1\}^{2n}$ be a PRG (for every n), and let $s \in \{0,1\}^n$. Prove that the following construction *is NOT* a secure PRG.

$G_d(s)$: first run $G(s) = x||y$ (where $|x| = |y| = n-$bits). Then run $G(y) = u||v$. Output $x||(y \oplus u)||v$.
Known fact: if $H : \{0,1\}^n \to \{0,1\}^{3n}$ is a PRG, then $G(s_L||s_R) = s_L||H(s_R)$ is also a PRG (where $s = s_L||s_R$ and $|s_L| = |s_R|$ (you can assume n is always even).

Proof: Gd is Not a Secure PRG

We are given that G: {0,1}^n→{0,1}^2n is a **secure Pseudorandom Generator (PRG)**, meaning its output is computationally indistinguishable from a uniformly random string of length 2n. Our task is to show that the modified construction Gd does not necessarily preserve this security.

Step 1: Understanding the Construction Gd

The function Gd(s) is defined as follows:

1. Compute G(s)=x||y, where |x|=|y|=n.
2. Compute G(y)=u||v, where |u|=|v|=n.
3. The final output of Gd(s) is: Gd(s)=x||(y⊕u)||v

To be a secure PRG, Gd(s) should be computationally indistinguishable from a truly random string of length 3n. We will now construct an efficient distinguisher that can tell apart the output of Gd from a uniformly random string.

Step 2: Identifying a Structural Weakness

A crucial observation lies in the middle part of the output, y⊕u. Given that u is derived from G(y), there is a potential dependency between y and u, leading to a non-random structure.

To make this concrete, consider a secure PRG H:{0,1}^n/2→{0,1}^3n/2 and define:

G(s)=sL||H(sR),

where:

- sL and sR are the left and right halves of s (each of length n/2).
- H(sR) produces 3n/2 bits.

From this, when we compute G(s), we obtain:

- x=sL||(first n/2 bits of H(sR)),
- y=(remaining n bits of H(sR)).

Similarly, computing G(y)=u||v involves splitting y into yL||yR, each of length n/2, and setting:

- u=yL||(first n/2 bits of H(yR)),
- v= (remaining n bits of H(yR)).

Now, look at the middle part of Gd(s):

y⊕u=(yL||yR) ⊕(yL||first n/2 bits of H(yR)).

Since the first n/2 bits of y match the first n/2 bits of u, XORing them results in all zeros in those positions:

y⊕u= (0^n/2||yR⊕(first n/2 bits of H(yR))).

This introduces a predictable pattern in the output of Gd, making it distinguishable from a truly random string.

Step 3: Constructing a Distinguisher

We define an efficient distinguisher D that takes a string z∈{0,1}^3n and determines whether it comes from Gd(s) or is uniformly random.

1. Split z into three equal parts: z=a||b||c, where |a|=|b|=|c|=n.
2. Check if the first n/2 bits of b are all zeros.
3. If they are all zeros, output 1 ("Generated by Gd").
4. Otherwise, output 0 ("Random").

Step 4: Analyzing the Distinguishing Probability

- If z is generated by Gd(s), then the first n/2 bits of b=y⊕u are always zeros. Thus, D outputs 1 with probability 1.
- If z is a truly random string, the probability that the first n/2 bits of bb are all zeros is: 1/2^(n/2). This is exponentially small in n.

Thus, the distinguishing advantage of D is:

|1−1/2^(n/2) |≈1,

which is non-negligible.

Since we have constructed a distinguisher D with non-negligible advantage, Gd fails the security requirement of a PRG. This proves that Gd is not necessarily a secure PRG, completing the proof.

References:
1. ChatGPT Assistance

2. Class Materials

Lecture slides and notes from CS 587 covering topics on shift ciphers, probability, Bayes' Theorem, and perfect secrecy.

Course textbook or reference materials recommended by the professor.

3.Cryptography References from Online Sources

Katz, J. & Lindell, Y. (2020). Introduction to Modern Cryptography (3rd ed.). CRC Press.

Menezes, A. J., van Oorschot, P. C., & Vanstone, S. A. (1996). Handbook of Applied Cryptography. CRC Press.

Boneh, D. & Shoup, V. (2017). A Graduate Course in Applied Cryptography. (Online version is available at https://crypto.stanford.edu/~dabo/cryptobook/)

4.Web Resources

Cryptography Stack Exchange : Discussion and explanations on several concepts of cryptography (https://crypto.stackexchange.com)

NIST Cryptographic Standards : Guidelines and recommendations on encryption schemes (https://csrc.nist.gov/)

Wikipedia-One-Time Pad : General overview of the One-Time Pad and perfect secrecy (https://en.wikipedia.org/wiki/One-time_pad)

Khan Academy-Probability & Bayes' theorem : Basic concepts surrounding probability (https://www.khanacademy.org/math/statistics-probability)