① (a) $1 - \mathbb{P}(X_1 \neq 6 \cap X_2 \neq 6 \cap X_3 \neq 6 \cap X_4 \neq 6) = 1 - \mathbb{P}(X_1 \neq 6)^4 = 1 - \left(\frac{5}{6}\right)^4$

(b) Bet 1: win := guessing first 6 coin tosses

$\Pr(\text{win}) = \left(\frac{1}{2}\right)^6 \Rightarrow \mathbb{E}[\text{gain}] = \Pr(\text{win}) \cdot 50 - \Pr(\text{loss}) \cdot 1$

$= \frac{1}{26} \cdot 50 - \left(1 - \frac{1}{26}\right) \cdot 1 = -\frac{13}{64}$

Bet 2: win := guessing 7 coin tosses as a subsqce of length 10 tosses

Let A_1 := 7 coin toss guess appears in the first 7 positions

A_2 := " 2-8 "

A_3 := " 3-9 "

A_4 := " 4-10 "

$\Pr(\text{win}) = \Pr(A_1 \cup A_2 \cup A_3 \cup A_4) = \sum_{i=1}^{4} \Pr(A_i) - \sum_{i<j} \Pr(A_i \cap A_j) + \sum_{i<j<k} \Pr(A_i \cap A_j \cap A_k) - \Pr(A_1 \cap A_2 \cap A_3 \cap A_4)$

↓

This prob. is maximized when the events A_i's have no intersection, so $\Pr(\cdot \cap \cdot) = 0$.

BEST: So, if you are going for Bet 2, winning prob. is maximized with a guess that can appear exactly once as a subsqce of length 10 sqce.

Ex: HHTTHHT

$\Pr(\text{win}) = \sum_{i=1}^{4} \Pr(A_i) = 4 \cdot \frac{1}{2^7} = \frac{1}{32} \Rightarrow \mathbb{E}[\text{gain}] = \Pr(\text{win}) \cdot 50 - \Pr(\text{loss}) \cdot 1$

$= \frac{1}{32} \cdot 50 - \left(1 - \frac{1}{32}\right) \cdot 1 = \frac{19}{32}$.

② X_i, Y_i are indep. $\Leftrightarrow \Pr(X_i = x, Y_i = y) = \Pr(X_i = x) \Pr(Y_i = y) \quad \forall x, y \in \{0,1\}$

$P_1(X_1, Y_1)$: $P_1(0,0) = \frac{1}{2}$ $P_1(0,1) = 0$ $P_1(1,0) = 0$ $P_1(1,1) = \frac{1}{2}$

$\Rightarrow$ not indep.: $\Pr(X_i = 0, Y_i = 0) = \frac{1}{2} \neq \Pr(X_i = 0) \Pr(Y_i = 0) = \frac{1}{2} \cdot \frac{1}{2}$

$P_2(X_2, Y_2)$: $P_2(0,0) = \frac{1}{3}$ $P_2(0,1) = \frac{1}{3}$ $P_2(1,0) = 0$ $P_2(1,1) = \frac{1}{3}$

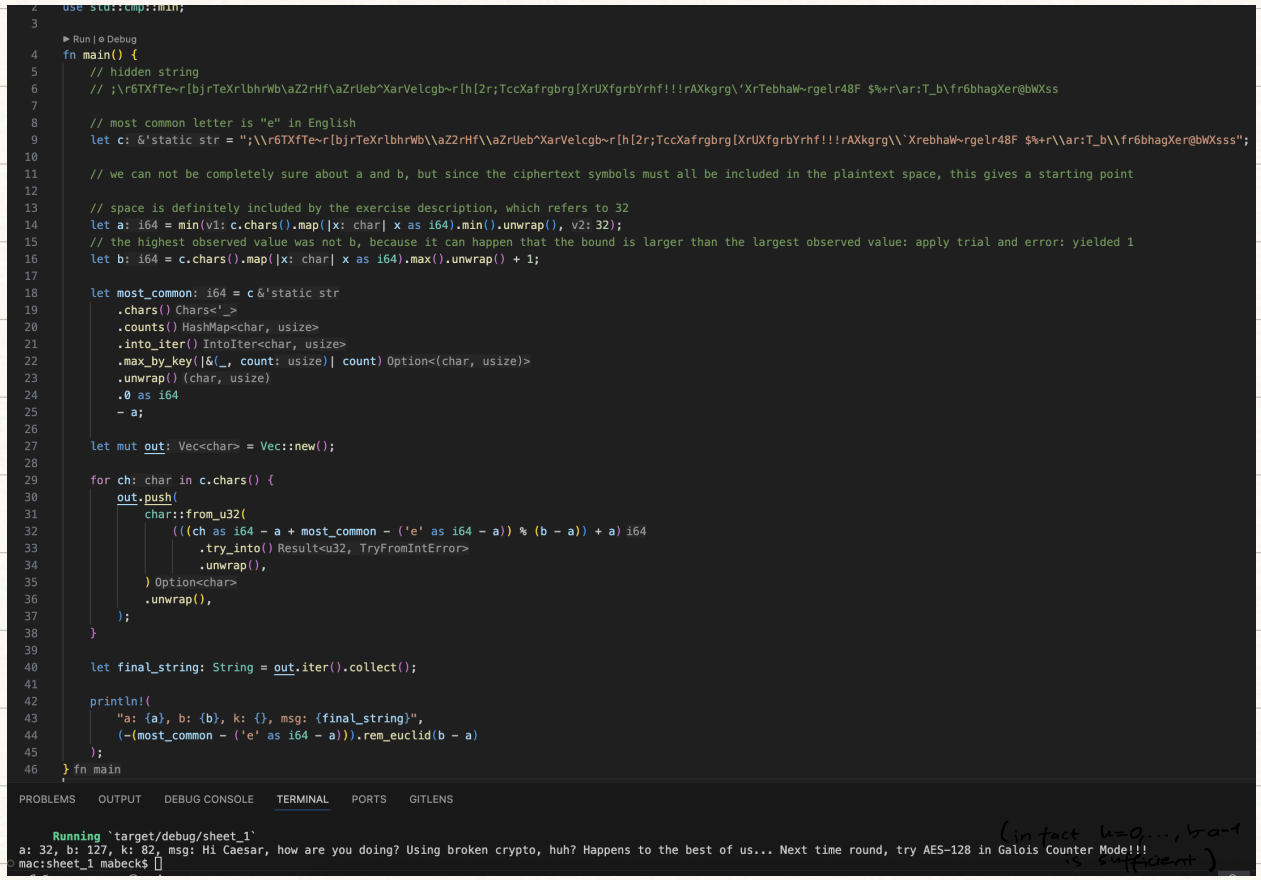
$\Rightarrow$ not indep.: $\Pr(X_i = 0, Y_i = 0) = \frac{1}{3} \neq \Pr(X_i = 0) \Pr(Y_i = 0) = \frac{2}{3} \cdot \frac{1}{3}$

$P_3(X_3, Y_3)$: $P_3(0,0) = \frac{1}{4}$ $P_3(0,1) = \frac{1}{4}$ $P_3(1,0) = \frac{1}{4}$ $P_3(1,1) = \frac{1}{4}$

$\Rightarrow X_3$ and Y_3 are independent.

Show that desired equality holds for all $(X_3, Y_3) \in \{0,1\}^2$.

3. In summary, fix some candidates for a and b, then find key k by brute force or frequency analysis or some other strategy.



```rust
use std::cmp::min;

▶ Run | ⚙ Debug
fn main() {
    // hidden string
    // ;\r6TXfTe~r[bjrTeXrlbhrWb\aZ2rHf\aZrUeb^XarVelcgb~r[h[2r;TccXafrgbrg[XrUXfgrbYrhf!!!rAXkgrg\'XrTebhaW~rgelr48F $%+r\ar:T_b\fr6bhagXer@bWXss

    // most common letter is "e" in English
    let c: &'static str = ";\\r6TXfTe~r[bjrTeXrlbhrWb\\aZ2rHf\\aZrUeb^XarVelcgb~r[h[2r;TccXafrgbrg[XrUXfgrbYrhf!!!rAXkgrg\\`XrebhaW~rgelr48F $%+r\\ar:T_b\\fr6bhagXer@bWXsss";

    // we can not be completely sure about a and b, but since the ciphertext symbols must all be included in the plaintext space, this gives a starting point

    // space is definitely included by the exercise description, which refers to 32
    let a: i64 = min(v1: c.chars().map(|x: char| x as i64).min().unwrap(), v2: 32);
    // the highest observed value was not b, because it can happen that the bound is larger than the largest observed value: apply trial and error: yielded 1
    let b: i64 = c.chars().map(|x: char| x as i64).max().unwrap() + 1;

    let most_common: i64 = c &'static str
        .chars() Chars<'_>
        .counts() HashMap<char, usize>
        .into_iter() IntoIter<char, usize>
        .max_by_key(|&(_, count: usize)| count) Option<(char, usize)>
        .unwrap() (char, usize)
        .0 as i64
        - a;

    let mut out: Vec<char> = Vec::new();

    for ch: char in c.chars() {
        out.push(
            char::from_u32(
                (((ch as i64 - a + most_common - ('e' as i64 - a)) % (b - a)) + a) i64
                    .try_into() Result<u32, TryFromIntError>
                    .unwrap(),
            ) Option<char>
            .unwrap(),
        );
    }

    let final_string: String = out.iter().collect();

    println!(
        "a: {a}, b: {b}, k: {}, msg: {final_string}",
        (-(most_common - ('e' as i64 - a))).rem_euclid(b - a)
    );
} fn main
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

    Running `target/debug/sheet_1`
a: 32, b: 127, k: 82, msg: Hi Caesar, how are you doing? Using broken crypto, huh? Happens to the best of us... Next time round, try AES-128 in Galois Counter Mode!!!
mac:sheet_1 mabeck$
```

(in fact k=0..., b-a sufficient)

(4) Physical security systems vary in how well they follow Kerckhoff's principle. When they rely too much on obscurity (like hidden camera replacement or weak locks mechanisms), they become unreliable.

① SKE is correct $\Leftrightarrow \forall k \in K, \forall m \in M: \Pr[Dec(k, Enc(k,m)) = m] = 1$.

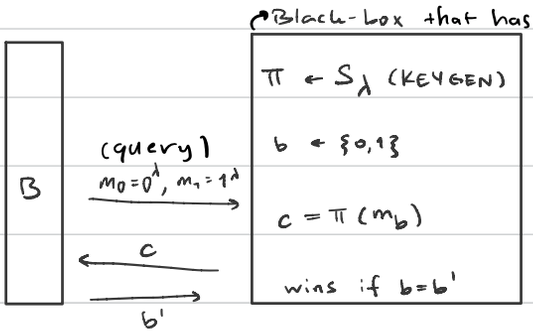
Set $\lambda = 2$, $k = 01$, $m = 10$. Then $Dec(k, Enc(k,m)) = 00$ but $m = 10$.

So, there exists a key and message such that $Dec(k, Enc(k,m)) \neq m \Rightarrow \Pr[Dec(k, Enc(k,m)) = m] \neq 1$.

② Permutation π is applied to the indices of the given message $\Rightarrow$ message is shuffled.

(1) $Dec(k, Enc(k,m)) = Dec(k, \pi(m)) = \pi^{-1}(\pi(m)) = (\pi^{-1} \circ \pi)(m) = Id(m) = m$.

(2)



where $b' = \begin{cases} 0 \text{ if } c = 0^\lambda \\ 1 \text{ if } c = 1^\lambda \end{cases}$ as part of the strategy of B.

$$\Pr[b = b'] = \Pr[b = b' \wedge L_{OTS-0}] + \Pr[b = b' \wedge L_{OTS-1}]$$

(i.e. $b = 0$ / i.e. $b = 1$)

$$= \underbrace{\Pr[0 = b' \mid L_{OTS-0}]}_{1} \underbrace{\Pr[L_{OTS-0}]}_{\frac{1}{2}} + \underbrace{\Pr[1 = b' \mid L_{OTS-1}]}_{1} \underbrace{\Pr[L_{OTS-1}]}_{\frac{1}{2}} = 1$$

Recall: For any event A and B, $\Pr(A \cap B) = \Pr(A|B)\Pr(B) = \Pr(B|A)\Pr(A)$.

③ (1) $Dec(k, Enc(k,m)) = Dec(k, k \oplus m) = k \oplus (k \oplus m) = m$

(2) $m = 111$ & $k[i] \leftarrow B_{0.75} \Rightarrow c[i] = 1$ with prob. 0.75 and $c[i] = 0$ w. prob. 0.25

- $\Pr[k = 000] = (0.75)^3 \Rightarrow \Pr[c = 111] = (0.75)^3$
- $\Pr[k = 100] = \Pr[010] = \Pr[001] = (0.75)^2 (0.25)$

  $\Rightarrow \Pr[c = 011] = \Pr[101] = \Pr[110] = (0.75)^2 (0.25)$
- $\Pr[k = 110] = \Pr[101] = \Pr[011] = (0.75)(0.25)^2$

  $\Rightarrow \Pr[c = 001] = \Pr[010] = \Pr[100] = (0.75)(0.25)^2$
- $\Pr[k = 111] = (0.25)^3 \Rightarrow \Pr[c = 000] = (0.25)^3$

(3) $\Pr[B \circ L_{OTS-Real} = 1] = \Pr[Enc(k,m) = m] = \Pr[k \oplus m = m] = \Pr[k = 000] = (0.75)^3$

$\Pr[B \circ L_{OTS-Rand} = 1] = \Pr[c = m] = 1/8$

(4) $L_{OTS-Real} \equiv L_{OTS-Rand}$ if for every $m \in M, c \in C$ $\Pr[c = Enc(k,m) \mid k \leftarrow keygen()] = 1/|c|$.

By part (3), we know that for $m = 111$, $c = 111$, this is not the case.

(5) Optimal strategy is that B outputting 1 when $c = 111, 110, 101$, or 011 because all this ciphertexts occur with higher probability in the real world compared to random one.

① 1. 3DES.DEC(k,c):

1. Check that $k = (k_1, k_2, k_3)$.

2. Output DES.DEC(k_1, DES.ENC(k_2, DES.DEC(k_3, c)))

Correctness:

We assume DES is correct, i.e., $\forall k \in K, m \in M$ DES.DEC(k, DES.ENC(k,m)) = m.

Moreover, since DES is a Feistel construction, ENC and DEC algorithms are inverses of each other, we know DES.ENC(k, DES.DEC(k,m)) = m. Then,

3DES.DEC(k, 3DES.ENC(k,m))

= DES.DEC(k_1, DES.ENC(k_2, DES DEC(k_3, DES.ENC(k_3, DES.DEC(k_2, DES.ENC(k_1,m))))))

= DES.DEC(k_1, DES.ENC(k_2, DES.DEC(k_2, DES.ENC(k_1,m)))))

= DES.DEC(k_1, DES.ENC(k_1,m)) = m

2. For every key k, you can check if $m = DES.DEC(k,c)$ or $c = DES.ENC(k,m)$.

(a) $|K| = 2^{56} \geq 2^{54.8}$ ns ≈ a year (key space for DES)

(b) $1024 = 2^{10} \Rightarrow 2^{56}/2^{10} = 2^{46} = 2^4 \cdot 2^{42}$ ns ≈ 16 hours

(c) K': key space for 3DES ⇒ $|K'| = 2^{3\times56} = 2^{168} \geq 2^{86.5}$ ns ≈ age of universe

② 1. Take r as a global variable.

$G_{PRF\text{-}real}$

$k \leftarrow K$

Lookup $(x \in \{0,1\}^{in})$:

1. $y \leftarrow G(k,x)$
2. output y

$G_{PRF\text{-}rand}$

$T = [\,]$

Lookup$(x \in \{0,1\}^{in})$:

1. If T[x] is undefined then $T(x) \leftarrow \{0,1\}^{out}$
2. Output T[x]

2. $L^{G}_{PRF\text{-}real} \equiv \overline{L}^{G}_{PRF\text{-}real} \circ L^{F}_{PRF\text{-}real}$

$k \leftarrow K$

1. $y \leftarrow Lookup(x)$
2. $y' \leftarrow y \oplus r$
3. output y'

3. $L^{G}_{PRF\text{-}real} \equiv \overline{L}^{G}_{PRF\text{-}real} \circ L^{F}_{PRF\text{-}real} \equiv \overline{L}^{G}_{PRF\text{-}real} \circ L^{F}_{PRF\text{-}rand} := L$

4. $Lookup(x) \oplus r$, output of L, can be replaced with the call to the random library since XORing with r will produce something random. (Remember OTP.)

Hence, $L \equiv L^{G}_{PRF\text{-}rand}$, and consequently $L^{G}_{PRF\text{-}real} \equiv L^{G}_{PRF\text{-}rand}$.

③ 1.

```
def Birthdayproblem(q,N):
    p = 1
    for i in range(q-1):
        p = p * (1-(i/N))
    return 1-p

def Upperbound(q,N):
    return (q*(q-1))/2*N
```

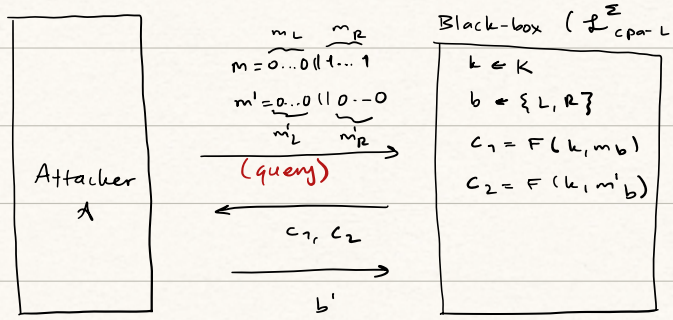
2. Assume #people = 22 = q and #days = 365 = N, then Birthdayproblem(22, 365) $\approx \frac{1}{2}$

3. The only way to distinguish F from G is if we find two inputs to the library such that $Lookup(x) = Lookup(x')$. So, prob. of distinguishing F from G is upper bounded by (success of finding two people with the same birthday) $\frac{q(q-1)}{2N}$ where $N = 2^{32}$. (length of outputs)

Compute for $q = 2, 2^8, 2^{12}, 2^{16}$.

① 1. There are two messages $m = \underbrace{0\ldots0}_{B}\underbrace{1\ldots1}_{B}$ and $m' = \underbrace{0\ldots0}_{B}\underbrace{0\ldots0}_{B}$, each consisting of two blocks. In ECB, each block is encrypted with the same key. So, encryption of m' will be duplicate of the same block, whereas, encryption of m will be concatenation of distinct blocks. (Note that block ciphers / PRPs are invertible.)

2.



Recall: $\mathcal{L}^{\Sigma}_{cpa\text{-}L}$ encrypts the left block of the messages, similarly $\mathcal{L}^{\Sigma}_{cpa\text{-}R}$ does the right

Strategy: $b' = \begin{cases} L & \text{if } c_1 = c_2 \\ R & \text{if } c_1 \neq c_2 \end{cases}$

$$Pr[\mathcal{A} \text{ wins}] = Pr[b=b'] = Pr[b=b' \cap b=L] + Pr[b=b' \cap b=R]$$

$$= Pr[b=b' \mid b=L]\, Pr[b=L] + Pr[b=b' \mid b=R]\, Pr[b=R]$$

$$= \underbrace{1}_{1} \cdot \underbrace{\frac{1}{2}}_{\frac{1}{2}} + \underbrace{1}_{1} \cdot \underbrace{\frac{1}{2}}_{\frac{1}{2}} = 1$$

② 1. Assume we encrypt without adding whole block of length B for padding. When we decrypt the encrypted message and see 01 at the end, we will think 1 represents the no. of bytes added to the original message m due to the designated padding. Then we will ignore 01 and treat the rest as the original message, which is not the case.

2. Notice that number of bytes to be added, namely b, is encoded as a single byte. And an upper bound for b is B, i.e., $b \leq B$. So, B should fit into a single byte. One byte is 8 bits and greatest number that can be written with 8 bits is 255. So, $B \leq 255$.

3. Let $m = m_1 \| m_2 \| \dots \| m_t$ where $|m_i| = B$ for $i = 1, \dots, t-1$ and $0 < |m_t| < B$, i.e., m is not a multiple of the block length.

$r \leftarrow \{0,1\}^B$
$c_0 := r$
for $i = 1$ to $t-1$:
  $c_i := F(k, r) \oplus m_i$
  $r := r + 1 \bmod 2^B$
$c_t := F(k, r)[0 : (|m_t| - 1)] \oplus m_t$
return $c_0 \| c_1 \| \dots \| c_t$

} exact steps of CTR

} here we chop $F(k,r)$ to do bitwise XOR with m_t
} as a result c_t has the same size with m_t and without padding we can encode the info. about size of the message m in the ciphertext

(3) 1. Assume adversary $\mathcal{A}$ calls Sample($R \subset \{0,1\}^\lambda$) and gets output r. If $r \in R$, adv. $\mathcal{A}$ knows that he talked to $L_{GenSample}$ library. If $r \in R$, adv. $\mathcal{A}$ cannot conclude something better than a random guess. More precisely, algo. $\mathcal{A}$:

$r \leftarrow$ Sample($R \subset \{0,1\}^\lambda$)
if $r \in R$:
  output 1 (succeeding in distinguishing)
else:
  output 0

2. Observe that when $\mathcal{A}$ talks to $L_{GenSample|ignore}$, $\mathcal{A}$ never outputs 1, i.e., $Pr[\mathcal{A} \circ L_{GenSample|ignore} \Rightarrow 1] = 0$. On the other hand, when $\mathcal{A}$ talks to $L_{GenSample}$, $\mathcal{A}$ outputs 1 if and only if $r \in R$. So, outputs of $L_{GenSample}$ for which $\mathcal{A}$ will output 1 with the same probability as when $\mathcal{A}$ talks to $L_{GenSample|ignore}$ are $r \notin R$.

And $Pr[\mathcal{A} \circ L_{GenSample} \Rightarrow 1] = Pr[L_{GenSample} \text{ outputs } r \in R] = \frac{|R|}{2^\lambda}$.

$Adv(\mathcal{A}) = |Pr[\mathcal{A} \circ L_{GenSample|ignore} \Rightarrow 1] - Pr[\mathcal{A} \circ L_{GenSample} \Rightarrow 1]| = \frac{|R|}{2^\lambda}$.

3. Let $r_1 \leftarrow$ Sample($R_1 \subset \{0,1\}^\lambda$) is the output of the 1st call and, similarly, $r_2 \leftarrow$ Sample($R_2 \subset \{0,1\}^\lambda$) is the output of the 2nd call. By part (2), we know that $\mathcal{A}$ can only distinguish if it talks to $L_{GenSample}$ and gets $r_1 \in R_1$ or $r_2 \in R_2$. So, distinguishing prob. is maximal when $\mathcal{A}$ talks to $L_{GenSample}$, and distinguishing prob. is $Pr[r_1 \in R_1 \text{ or } r_2 \in R_2]$

$= 1 - Pr[r_1 \notin R_1 \text{ and } r_2 \notin R_2] = 1 - Pr[r_1 \notin R_1] \, Pr[r_2 \notin R_2] = 1 - \left(1 - \frac{|R_1|}{2^\lambda}\right)\left(1 - \frac{|R_2|}{2^\lambda}\right)$.

(Recall: A and B are independent $\Leftrightarrow Pr(A) \, Pr(B) = Pr(A \cap B)$)

4. Similar to (3), maximal prob. of distinguishing/success with q calls is $1 - \prod_{i=1}^{q} \left(1 - \frac{|R_i|}{2^d}\right)$.

5. A has polynomial runtime, therefore can make poly-many calls, i.e., $q \leq \text{poly}(d)$. Also, in each call A generates a set R where generating each item takes 1 unit of time, therefore size of the set R in each call is $\text{poly}(d)$. Then prob. success $\leq$ (by part (4)) $\frac{q^2 \cdot |R|}{2^d} = \frac{\text{poly}(d)}{2^d}$ is negligible.

① 1. $t_1 = 0 \cdot MAC(k, 0^\lambda)$ and $t_2 = 0 \cdot MAC(k, 1 \| 0^{\lambda-1})$

$t_1' = 0 \cdot MAC(k, 0 \| 1^{\lambda-1})$ and $t_2' = 0 \cdot MAC(k, 1^\lambda)$

Then for $m = 0^{\lambda-1} \| 1^{\lambda-1}$, $ECB\text{-}MAC'(k, m) = (t_1, t_2')$. (malleable)

2. $\mathcal{A} \circ L$ where L is either $L_{MAC\text{-}real}$ or $L_{MAC\text{-}fake}$ for ECB-MAC'.

$(t_1, t_2) \leftarrow Gettag(0^{2\lambda-2})$

$(t_1', t_2') \leftarrow Gettag(1^{2\lambda-2})$

if $Checktag(0^{\lambda-1} \| 1^{\lambda-1}, (t_1, t_2')) == True$:

output "real"

else:

output "fake"

$Pr[\mathcal{A} \text{ succeeds}] = Pr[\mathcal{A} \circ L \Rightarrow \text{"real"} \wedge L = L_{MAC\text{-}real}] + Pr[\mathcal{A} \circ L \Rightarrow \text{"fake"} \wedge L = L_{MAC\text{-}fake}]$

$= Pr[\mathcal{A} \circ L \Rightarrow \text{"real"} \mid L = L_{MAC\text{-}real}] \, Pr[L = L_{MAC\text{-}real}]$

$+ Pr[\mathcal{A} \circ L \Rightarrow \text{"fake"} \mid L = L_{MAC\text{-}fake}] \, Pr[L = L_{MAC\text{-}fake}]$

$= 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = 1$

② 1. By assumption, we have $t = 0 \cdot MAC(k, 0 \cdot MAC(k, m_1) \oplus m_2)$. The MAC for $m_1 \| m_2 \| (m_1 \oplus t) \| m_2$ is

$0 \cdot MAC(k, 0 \cdot MAC(k, \underbrace{0 \cdot MAC(k, 0 \cdot MAC(k, m_1) \oplus m_2)}_{t} \oplus (m_1 \oplus t)) \oplus m_2) = t$

$\underbrace{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}_{m_1 \oplus t \oplus t}$

2. $\mathcal{A} \circ L$ where L is either $L_{MAC\text{-}real}$ or $L_{MAC\text{-}fake}$ for CBC-MAC

$t \leftarrow Gettag(m_1 \| m_2)$

if $Checktag(m_1 \| m_2 \| (m_1 \oplus t) \| m_2, t) == True$:

output "real"

else:

output "fake"

Similar to exercise ①, $Pr[\mathcal{A} \text{ succeeds}] = 1$.

③ 1. $\Sigma.\mathrm{Dec}\left((k_E, k_M), \Sigma.\mathrm{Enc}\left[(k_E, k_M), m\right]\right) = \Sigma.\mathrm{Dec}\left((k_E, k_M), (\Omega.\mathrm{Enc}(k_E, m), \theta.\mathrm{MAC}(k_M, c))\right)$

$= \Omega.\mathrm{Dec}(k_E, \Omega.\mathrm{Enc}(k_E, m)) = m$

↓ by correctness of Ω

2. $L^{\Sigma}_{CCA\text{-}0}$

$k \leftarrow \Sigma.\mathrm{KeyGen}()$
$S = \emptyset$
CTXT$(m_0, m_1 \in \Sigma.M)$:
1. If $|m_0| \neq |m_1|$ output $\perp$
2. $c \leftarrow \Sigma.\mathrm{Enc}(k, m_0)$
3. $S = S \cup \{c\}$
4. Output c

Decrypt$(c \in \Sigma.C)$:
1. If $c \in S$ output $\perp$
2. Output $\Sigma.\mathrm{Dec}(k, c)$

Replace → every call to Σ with their definitions

$L_{Step\text{-}1}$

$k_E \leftarrow \Omega.\mathrm{KeyGen}()$
$k_M \leftarrow \theta.\mathrm{KeyGen}()$
$k = (k_E, k_M)$
$S = \emptyset$
CTXT$(m_0, m_1 \in \Omega.M)$:
1. If $|m_0| \neq |m_1|$ output $\perp$
2. $c' \leftarrow \Omega.\mathrm{Enc}(k_E, m_0)$
3. $t' \leftarrow \theta.\mathrm{MAC}(k_M, c')$
4. $S = S \cup \{(c', t')\}$
5. output $c = (c', t')$

Decrypt$(c = (c', t') \in \Omega.C \times \theta.T)$
1. If $c \in S$ output $\perp$
2. If $t' \neq \theta.\mathrm{MAC}(k_M, c')$ output $\perp$
3. output $\Omega.\mathrm{Dec}(k_E, c')$

3. Now, we make a call to $L^{\theta}_{MAC\text{-}real}$ every time we need to use $\theta.\mathrm{MAC}$.

composition of libraries (meaning functions called are defined as in $L^{\theta}_{MAC\text{-}real}$)

$L_{Step\text{-}1} \circ L^{\theta}_{MAC\text{-}real}$

$k_E \leftarrow \Omega.\mathrm{KeyGen}()$
$k_M \leftarrow \theta.\mathrm{KeyGen}()$
$k = (k_E, k_M)$
$S = \emptyset$
CTXT$(m_0, m_1 \in \Omega.M)$:
1. If $|m_0| \neq |m_1|$ output $\perp$
2. $c' \leftarrow \Omega.\mathrm{Enc}(k_E, m_0)$
3. $t' \leftarrow \mathrm{Gettag}(c')$
4. $S = S \cup \{(c', t')\}$
5. output $c = (c', t')$

Decrypt$(c = (c', t') \in \Omega.C \times \theta.T)$
1. If $c \in S$ output $\perp$
2. If $\mathrm{Checktag}(c', t') = \mathrm{false}$ output $\perp$
3. output $\Omega.\mathrm{Dec}(k_E, c')$

$L^{\theta}_{MAC\text{-}real}$

$k_M \leftarrow \theta.\mathrm{KeyGen}()$
Gettag$(m, \theta.M)$:
Return $\theta.\mathrm{MAC}(k_M, m)$
Checktag$(m \in \theta.M, t \in \theta.T)$:
Return $\theta.\mathrm{MAC}(k_M, m) == t$

Since the library $L^{\theta}_{MAC\text{-}real}$ uses the actual functions of θ we get a perfect simulation of $L^{\Sigma}_{CCA\text{-}0}$ by $L_{Step\text{-}1} \circ L^{\theta}_{MAC\text{-}real}$, then $L^{\Sigma}_{CCA\text{-}0} \equiv L_{Step\text{-}1} \circ L^{\theta}_{MAC\text{-}real}$

4. By assumption θ is a secure MAC scheme, i.e., $L^{\theta}_{\text{MAC-real}} \approx L^{\theta}_{\text{MAC-fake}}$.

And we know that if $L_1 \approx L_2$ and L runs in poly. time, then $L \circ L_1 \approx L \circ L_2$.

Therefore, $L_{\text{Step-1}} \circ L^{\theta}_{\text{MAC-real}} \approx L_{\text{Step-1}} \circ L^{\theta}_{\text{MAC-fake}}$.

5. We write $L_{\text{Step-1}} \circ L^{\theta}_{\text{MAC-fake}}$ into a new lib. that doesn't call $L^{\theta}_{\text{MAC-fake}}$

$L^{\theta}_{\text{MAC-fake}}$

$k_M \leftarrow \theta.\text{KeyGen}()$

$S_\theta = \emptyset$

GetTag$(m \in \theta.M)$:

1. $t = \theta.\text{MAC}(k_M, m)$
2. $S_\theta = S_\theta \cup \{(m,t)\}$
3. output t

CheckTag$(m \in \theta.M, t \in \theta.T)$

return $(m,t) \in S_\theta$

$L_{\text{Step-2}}$

$k_E \leftarrow \Omega.\text{KeyGen}()$

$k_M \leftarrow \theta.\text{KeyGen}()$

$k = (k_E, k_M)$

$S = \emptyset$

CTXT$(m_0, m_1 \in \Omega.M)$:

1. If $|m_0| \neq |m_1|$ output $\perp$
2. $c' \leftarrow \Omega.\text{Enc}(k_E, m_0)$
3. $t' \leftarrow \theta.\text{MAC}(k_M, c')$
4. $S = S \cup \{(c', t')\}$
5. output $c = (c', t')$

Decrypt$(c = (c', t') \in \Omega.C \times \theta.T)$

1. If $c \in S$ output $\perp$
2. If $c \notin S$ output $\perp$
3. output $\Omega.\text{Dec}(k_E, c')$

Notice that we didn't introduce the set S_θ in $L_{\text{Step-2}}$. There is no need of to do this, since S and S_θ contain the same elements. We just rewrite the lib., their functions are the same. So,

$L_{\text{Step-2}} \equiv L_{\text{Step-1}} \circ L^{\theta}_{\text{MAC-fake}}$.

6. Under which conditions will $\Omega.\text{Dec}(K_E, c')$ be reached? Since $c \in S$ and $c \notin S$ are complementary events, line 3 of "Decrypt" is never reached. Therefore, $L_{\text{Step-3}} \equiv L_{\text{Step-2}}$.

7. Recall: $L^{\Omega}_{\text{CPA-0}}$

$k_E \leftarrow \Omega.\text{KeyGen}()$

CTXT$(m_0, m_1 \in \Omega.M)$:

1. If $|m_0| \neq |m_1|$ output $\perp$
2. $c \leftarrow \Omega.\text{Enc}(k_E, m_0)$
3. output c

$L_{step\text{-}4} \circ L^{\Omega}_{CPA\text{-}0}$

$k_E \leftarrow \Omega.KeyGen()$

$k_M \leftarrow \Theta.KeyGen()$

$k = (k_E, k_M)$

$S = \emptyset$

CTXT($m_0, m_1 \in \Omega.M$):

1. If $|m_0| \neq |m_1|$ output $\perp$
2. $c' \leftarrow$ CTXT(m_0, m_1)
3. $t' \leftarrow \Theta.MAC(k_M, c')$
4. $S = S \cup \{(c', t')\}$
5. output $c = (c', t')$

Decrypt($c = (c', t') \in \Omega.C \times \Theta.T$)

1. output $\perp$

Again, $L_{step\text{-}3} \equiv L_{step\text{-}4} \circ L^{\Omega}_{CPA\text{-}0}$ since they are using the same function to encrypt the message.

8. By assumption Ω is CPA-secure enc. scheme (for the left-right def.), i.e., $L^{\Omega}_{CPA\text{-}0} \approx L^{\Omega}_{CPA\text{-}1}$.

Similar to 4., we have $L_{step\text{-}4} \circ L^{\Omega}_{CPA\text{-}0} \approx L_{step\text{-}4} \circ L^{\Omega}_{CPA\text{-}1}$. By 2. to 7. we showed that $L^{\Sigma}_{CCA\text{-}0} \approx L_{step\text{-}4} \circ L^{\Omega}_{CPA}$. In the same way, we prove $L^{\Sigma}_{CCA\text{-}1} \approx L_{step\text{-}4} \circ L^{\Omega}_{CPA\text{-}1}$.

Combining all together we get $L^{\Sigma}_{CCA\text{-}0} \approx L_{step\text{-}4} \circ L^{\Omega}_{CPA\text{-}0} \approx L_{step\text{-}4} \circ L^{\Omega}_{CPA\text{-}1} \approx L^{\Sigma}_{CCA\text{-}1}$.

① MDMAC (k, m):

1. Let $m_1, \dots, m_\tau = \text{MDPad}_\ell(m)$ where $\ell = |m_i|$
2. Set $y_0 := k$
3. For $i = 1, \dots, \tau$:
   $y_i = h(m_i | y_{i-1})$
4. Output y_τ

Assume we are given $(m, t) = (m, \text{MDMAC}(k, m))$ with $\text{MDPad}_\ell(m) = m_1, \dots, m_\tau$ where $|m_i| = \ell$. Set $m' = m_1 | \dots | m_\tau$. Then $|m'| = \tau \cdot \ell$, and e' is the ℓ-bit string representing $|m'| = \tau \cdot \ell$ in binary form. So, $\text{MDPad}_\ell(m') = m_1, \dots, m_\tau, e'$ and $\text{MDMAC}(k, m') = h(e' | t)$. In conclusion, $(m', t') = (m', h(e' | t))$. ($m'$ is necessarily distinct from m because padding introduces length of m, i.e., $m_\tau = e$ where e is binary representation of $|m|$.)

② Let m be any message with $\text{SMDPad}_\ell(m) = m_1, \dots, m_\tau$ where $|m_1| = d$ and $|m_i| = \ell$ for $i \geq 2$ and $\text{SMDHash}(m) = h$. Set $m' = h(m_2 | m_1) | m_3, \dots, m_\tau$. Then $\text{SMDPad}_\ell(m') = h(m_2 | m_1), m_3, \dots, m_\tau$ and $\text{SMDHash}(m') = h$.

③ Recall:

$L^H_{\text{CR-real}}$

$s \leftarrow \{0,1\}^\lambda$

GetSalt():
Return s

Test(m_1, m_2):
If $H(s, m_1) = H(s, m_2)$ & $m_1 \neq m_2$:
  output 1
Else:
  output 0

$L^H_{\text{CR-fake}}$

$s \leftarrow \{0,1\}^\lambda$

GetSalt():
Return s

Test(m_1, m_2):
Output 0

Observe that H being homomorphic implies that $0^\lambda = H(1^n) \oplus H(1^n) = H(1^n \oplus 1^n) = H(0^n)$ (property of XOR; H is homomorphic; property of XOR)

for any n. Then construct adversary $\mathcal{A}$ as follows: $\mathcal{A}^{\mathcal{OL}}$:
If Test$(0, 0^2) = 1$:
  output "real"
Else:
  output "fake"

$$\Pr[\mathcal{A} \text{ succeeds}] = \Pr[\mathcal{A}\circ L \Rightarrow \text{real} \wedge L = L^{H}_{\text{CR-real}}] + \Pr[\mathcal{A}\circ L \Rightarrow \text{fake} \wedge L = L^{H}_{\text{CR-fake}}]$$

$$= \underbrace{\Pr[\mathcal{A}\circ L \Rightarrow \text{real} \mid L = L^{H}_{\text{CR-real}}]}_{1} \cdot \underbrace{\Pr[L = L^{H}_{\text{CR-real}}]}_{\frac{1}{2}} + \underbrace{\Pr[\mathcal{A}\circ L \Rightarrow \text{fake} \mid L = L^{H}_{\text{CR-fake}}]}_{1} \cdot \underbrace{\Pr[L = L^{H}_{\text{CR-fake}}]}_{\frac{1}{2}}$$

$$= \frac{1}{2}$$

④ Think of $F(x,y)$ as $F_x(y)$ where x is a key and F_x is a permutation from $\{0,1\}^B$ to $\{0,1\}^B$. In particular, F_x is surjective. For a given output value $z \in \{0,1\}^B$ for each key $x \in \{0,1\}^\lambda$, there exists only one value $y \in \{0,1\}^B$ s.t. $F_x(y) = z$.

$\mathcal{A} \circ L$:

$x_1 \leftarrow \{0,1\}^\lambda$ — This defines a perm. F_{x_1}

$y_1 \leftarrow \{0,1\}^B$ — Input for F_{x_1}

$z := F(x_1, y_1)$ — i.e. $F_{x_1}(y_1)$

$x_2 \leftarrow \{0,1\}^\lambda \setminus \{x_1\}$ — Creates a key different than x_1

$y_2 := F_{x_2}^{-1}(z)$ — Finds the input y_2 for F_{x_2} s.t. $F_{x_2}(y_2) = F_{x_1}(y_1) = z$

If $\text{Test}(x_1 | y_1, x_2 | y_2) = 1$: — We know $H(x_1|y_1) = H(x_2|y_2)$

output "real"

Else:

output "fake"

$\Pr[\mathcal{A} \text{ succeeds}] = 1$ as in Ex. 3.

① Coding exercise.

② 1. $x = \underset{a_1}{3} \bmod \underset{b_1}{7}$ and $x = \underset{a_2}{5} \bmod \underset{b_2}{9}$

- Compute T, i.e., $7^{-1} \bmod 9$:

$9 = 7 \cdot 1 + 2$

$7 = 2 \cdot 3 + 1 \Rightarrow 1 = 7 - 2 \cdot 3$

$= 7 - (9 - 7 \cdot 1)3$

$= 4 \cdot 7 - 3 \cdot 9 \Rightarrow 4 \cdot 7 = 1 \bmod 9 \Rightarrow 7^{-1} \bmod 9 = 4$

Compute $\gcd(7,9)$ using Euclidean algo. | Reverse it to write $\gcd(7,9)=1$ as a linear comb. of 7 and 9 | Reduce it mod 9 so that $-3 \cdot 9$ disappears since $9 \equiv 0 \bmod 9$. | Follows by definition of inverse in modulo

- Compute u, i.e., $(5-3) \cdot 4 \bmod 9 = 8$.

- Compute x, i.e., $3 + 8 \cdot 7 = 59$

2. In the computation of T, number of steps is mainly determined by number of steps in the Euclidean algorithm, and Euclidean algo. terminates in finite no. of steps since in each step remainder is smaller than the one previous remainder and set of remainders is bounded from below by 0. Computation of u and x are clearly finite many steps.

Output is correct, and we can check this by replacing final output x into given equations: $x = a_1 + u \cdot b_1 \bmod b_1 = a_1 \bmod b_1$ since $b_1 \equiv 0 \bmod b_1$

$$x = a_1 + u \cdot b_1 \bmod b_2 = a_1 + (a_2 - a_1) T \cdot b_1 \bmod b_2$$
$$= a_1 + (a_2 - a_1) b_1^{-1} \cdot b_1 \bmod b_2$$
$$= a_1 + (a_2 - a_1) \cdot 1 \bmod b_2 = a_2 \bmod b_2$$

③ 1. $c = m^e = m^{2^{16}+1} = m^{2^{16}} \cdot m \bmod N$ and $m^{2^{16}} = m^{2^{15}} \cdot m^{2^{15}} \bmod N$

1 mult.

$m^{2^{15}} = m^{2^{14}} \cdot m^{2^{14}} \bmod N$

$\vdots$

$m^{2^1} = m \cdot m \bmod N$

16 mult.

2. Let $sk' = (d, p, q) \leftarrow \text{Keygen}()$, then $\text{Dec}(sk, c)$ should compute $c^d \bmod N$. We can do this by computing $c^d \bmod p$ and $c^d \bmod q$, then apply CRT to compute unique solution $c^d \bmod N$.

3. Coding exercise.

Hint: When you have mod N in the base, you can work in mod $\varphi(N)$ in the power. You need to compute $m = c^d \bmod N$, instead compute $m = c_p^{d_p} \bmod p$ and $m = c_q^{d_q} \bmod q$ where $d_p = d \bmod \varphi(p)$, $d_q = d \bmod \varphi(q)$, $c_p = c \bmod p$, $c_q = c \bmod q$. Then apply CRT.

4. Assume $e = 2^{16} + 1$ and $|\varphi(N)| = 2000$ bits.

Recall that $d = e^{-1} \bmod \varphi(N)$, i.e., $e \cdot d = 1 \bmod \varphi(N)$. This means $e \cdot d = 1 + k \cdot \varphi(N)$ for some $k \in \mathbb{Z}$ over integers. Then $d = \frac{1 + k \cdot \varphi(N)}{e}$. Observe that $|1 + k \cdot \varphi(N)| \geq 2000$ bits and $|e| = 17$. So, $|d| \geq 2000 - 17 > 1980$

④ 1. $\varphi(N)$ is the number of integers from 1 to N that are relatively prime with N.

Consider $N = p \cdot q$ for different primes p and q, then any integer in $\{1, \cdots, N\}$ is relatively prime with N as long as it is not a multiple of p or q. So

$\varphi(N) = (p-1)(q-1) = pq - (p+q) + 1$.

Given $\varphi(N)$ and N, $N - \varphi(N) + 1 = pq - (pq - (p+q) + 1) + 1 = p + q$.

2. Roots of a degree two poly. $p(x) = ax^2 + bx + c$ are $\frac{\sqrt{\Delta} - b}{2a}$ and $\frac{-\sqrt{\Delta} - b}{2a}$, $\Delta = b^2 - 4ac$

Roots of $f(x)$ are clearly p and q since $f(p) = f(q) = 0$. Given N and $p+q$ (by 1) we can construct $f(x) = x^2 - (p+q)x + N$ and apply above formula to compute the roots p and q.

① 1. $\Pr\left[e_i \le \frac{p}{4\ell} \quad \forall i = 1, \dots, \ell\right] = \Pr\left[e_1 \le \frac{p}{4\ell} \cap \dots \cap e_\ell \le \frac{p}{4\ell}\right]$ (#samples)

$= \prod_{i=1}^{m} \Pr\left[e_i \le \frac{p}{4\ell}\right]$ (due to independence of e_i's)

$= \prod_{i=1}^{m} \left(1 - \Pr\left[e_i > \frac{p}{4\ell}\right]\right)$

$= \left(1 - \Pr\left[e > \frac{p}{4\ell}\right]\right)^{\ell}$ (due to randomness of e_i's)

$\ge (1-B)^{\ell}$

since $\Pr\left[e > \frac{p}{4\ell}\right] = \Pr\left[e \ge \left\lceil \frac{p}{4\ell} \right\rceil\right] \le \frac{\sigma}{\nu\left(\left\lceil \frac{p}{4\ell} \right\rceil - 1\right)\sqrt{2\pi}} \cdot e^{-\left(\left\lceil \frac{p}{4\ell} \right\rceil - 1\right)^2 / 2\sigma^2}$

- $\frac{p}{4\ell}$ not an integer since p is a prime
- $\left\lceil \frac{p}{4\ell} \right\rceil$ must be an integer

$\le \frac{\sigma}{\left(1 - \frac{1}{\sqrt{2\pi}\sigma} \cdot e^{-1}\right)\left(\left\lceil \frac{p}{4\ell} \right\rceil - 1\right)\sqrt{2\pi}} \cdot e^{-\left(\left\lceil \frac{p}{4\ell} \right\rceil - 1\right)^2 / 2\sigma^2} := B$

due to fact 1

2. $m' = \left\lfloor \frac{2}{p}(c_1 - c_0 \cdot s) \right\rceil = \left\lceil \frac{2}{p}\left(m\left(\frac{p-1}{2}\right) + \sum_{i=1}^{\ell} r_i \cdot b_i - s \cdot \sum_{i=1}^{\ell} r_i \cdot a_i\right)\right\rfloor$ (a_i: i-th row of A, a sample)

(b_i: size n; s: size n)

$= \left\lceil \frac{2}{p}\left(m\left(\frac{p-1}{2}\right) + \sum_{i=1}^{\ell} r_i \cdot (a_i \cdot s + e_i) - s \cdot \sum_{i=1}^{\ell} r_i \cdot a_i\right)\right\rfloor$ (a_i: i-th row of A)

$= \left\lceil \frac{2}{p}\left(m\left(\frac{p-1}{2}\right) + \sum_{i=1}^{\ell} r_i \cdot e_i\right)\right\rfloor$

$= \left\lceil m \cdot \frac{p-1}{p} + \frac{2}{p} \sum_{i=1}^{\ell} r_i \cdot e_i \right\rfloor$

Due to the rounding fct. we can recover m if $\left|\frac{2}{p} \sum_{i=1}^{\ell} r_i \cdot e_i\right| \le \frac{1}{2}$ i.e., $\left|\sum_{i=1}^{\ell} r_i \cdot e_i\right| \le \frac{p}{4}$.

Otherwise decryption fails and its probability is computed as follows:

$\Pr[\text{decryption failure}] = \Pr\left[\left|\sum_{i=1}^{\ell} r_i \cdot e_i\right| > \frac{p}{4}\right] = 1 - \Pr\left[\left|\sum_{i=1}^{\ell} r_i \cdot e_i\right| \le \frac{p}{4}\right]$ and observe that

$\Pr\left[\left|\sum_{i=1}^{\ell} r_i \cdot e_i\right| \le \frac{p}{4}\right] \ge \Pr\left[\sum_{i=1}^{\ell} |r_i \cdot e_i| \le \frac{p}{4}\right] \ge \Pr\left[\sum_{i=1}^{\ell} |e_i| \le \frac{p}{4}\right] = \Pr\left[\ell \cdot e \le \frac{p}{4}\right] = \Pr\left[e \le \frac{p}{4\ell}\right] \ge 1 - B$

- $\left|\sum_{i=1}^{\ell} r_i \cdot e_i\right| \le \sum_{i=1}^{\ell} |r_i \cdot e_i|$
- $|r_i \cdot e_i| \le |e_i|$ since $r_i \in \{0,1\}$
- $\ell \cdot e \le \frac{p}{4}$: random sample

So, $\Pr[\text{decryption failure}] \le B$ (computed in part 1.)

Remark: $x \ge B \iff -x \le -B \iff 1 - x \le 1 - B$

(2) 1. Notice that $b_i = s \cdot a_i + e_i = s \cdot a_i$ when $e_i = 0$ (a_i: vector, s: integer). Then we can write $b = \begin{bmatrix} b_1 \\ \vdots \\ b_\ell \end{bmatrix}$, $A = \begin{bmatrix} - a_1 - \\ \vdots \\ - a_\ell - \end{bmatrix}$ and observe that $b = A \cdot s$.

By assumption $\ell \gg n$, so we can solve this linear system using Gaussian elimination and get s. Once you have the secret key you can recover the message.

2. When $r_i = 1$ for all i, $c_1 = m\left(\frac{p-1}{2}\right) + \sum_{i=1}^{\ell} r_i \cdot b_i = m\left(\frac{p-1}{2}\right) + \sum_{i=1}^{\ell} b_i$. Since b_i's are known, we can compute $c_1 - \sum_{i=1}^{\ell} b_i = m\left(\frac{p-1}{2}\right) = \begin{cases} 0 & \text{if } m = 0, \\ \text{not } 0 & \text{if } m = 1. \end{cases}$

3. "$\Pr_{e \leftarrow D_\sigma}[e \neq 0 \bmod p] = \frac{1}{2\ell}$" can be interpreted as among ℓ-many samples of (a_i, b_i, e_i) only one of them will have a nonzero error e_i. Choosing some subset of these samples of size $> n$ and applying Gaussian elimination as in part 1 will give secret key with high probability. More precisely, as long as triples (a_i, b_i, e_i) where $e_i \neq 0$ are not included in the subset, Gaussian el. will work and s will be discovered.

(3) (referring to the one in Ex. 3, not the parameter in Regev's Enc.)

1. Note that in Regev's scheme, $\ell = 2$. Similarly, decryption scheme can be written as $\mathrm{Dec}(s, (c_0, c_1)) = \lfloor (c_1 - s \cdot c_0) \frac{\ell}{p} \rceil = \lfloor \left(e + \lfloor m \frac{(p-1)}{\ell} \rceil\right) \frac{\ell}{p} \rceil$.

Decryption will be correct if $|e \cdot \frac{\ell}{p}| < \frac{1}{2}$, i.e., $|e| < \frac{p}{2\ell} =: \sigma$.

2. $\mathrm{Dec}(s, c'') = \mathrm{Dec}(s, (c_0'', c_1'')) = \mathrm{Dec}(s, (c_0 + c_0', c_1 + c_1')) = \lfloor ((c_1 + c_1') - s(c_0 + c_0')) \frac{\ell}{p} \rceil$.

$c_1 + c_1' = (a + a')s + (e + e') + \lfloor m \frac{(p-1)}{\ell} \rceil + \lfloor m' \frac{(p-1)}{\ell} \rceil$ and $c_0 + c_0' = a + a'$ implies

$$\lfloor ((c_1 + c_1') - s(c_0 + c_0')) \frac{\ell}{p} \rceil = \lfloor \left((e + e') + \lfloor m \frac{(p-1)}{\ell} \rceil + \lfloor m' \frac{(p-1)}{\ell} \rceil \right) \frac{\ell}{p} \rceil.$$

When $|(e + e') \frac{\ell}{p}| < \frac{1}{2}$, the decryption will give $m + m'$.

① 1. Given a hash fct. $H: \{0,1\}^* \to \{0,1\}^n$ and a secure (EUF-CMA) digital sign. scheme $\Sigma = (\Sigma.\text{KeyGen}, \Sigma.\text{Sign}, \Sigma.\text{Ver})$ for messages of length n bits, we build a DSS for arbitrary-length messages. We denote the scheme by $\Pi = (\text{KeyGen}, \text{Sign}, \text{Ver})$ s.t. KeyGen = Σ.KeyGen outputs a pair (sk, vk),

$\text{Sign}(sk, m) = \Sigma.\text{Sign}(sk, H(m))$ where $m \in \{0,1\}^*$,

$\text{Ver}(vk, m, \sigma) = \Sigma.\text{Ver}(vk, H(m), \sigma)$ where σ is a signature.

2. Now, we want to show that Π is EUF-CMA secure assuming that Σ is secure and H is collision resistant. Let $\mathcal{A}$ be an adv. that aims to break Π, i.e., $\mathcal{A}$ wins if it outputs a pair (m, σ) s.t. $\text{Ver}(vk, m, \sigma) = 1$ for $(sk, vk) \in \text{KeyGen}$ and $m \notin \mathcal{L}_\Pi$, where $\mathcal{L}_\Pi$ is the list of messages that $\mathcal{A}$ queries to the oracle ($\mathcal{L}_\Pi = \{(m_i, \sigma_i)\}_{i \in I}$). To each message m_i in $\mathcal{L}_\Pi$ we can associate an hash $H(m_i)$ by doing so we create a second list $\mathcal{L}_\Sigma$. We notice that if (m_i, σ_i) is a valid signature for Π, then $(H(m_i), \sigma_i)$ is a valid signature for Σ. When $(m, \sigma) \notin \mathcal{L}_\Pi$ and it is a valid signature, we have two possibility:

1. $H(m) \notin \mathcal{L}_\Sigma$. In this case, $(H(m), \sigma)$ is a forgery for Σ.

2. $H(m) \in \mathcal{L}_\Sigma$. In this case, there exists $m_i \in \mathcal{L}_\Pi$ s.t. $m \neq m_i$ and $H(m) = H(m_i)$. Hence, we have a collision for H.

With this observation, given an EUF-CMA adv. $\mathcal{A}$ against Π, we get

$$\Pr[\text{EUF-CMA}_{\Pi,\mathcal{A}} = 1] = \Pr[\text{EUF-CMA}_{\Sigma,\mathcal{A}'} = 1 \text{ or } \text{Collision}_{H,\mathcal{A}''} = 1]$$

↳ (i.e. $\mathcal{A}$ wins)

$$\leq \underbrace{\Pr[\text{EUF-CMA}_{\Sigma,\mathcal{A}'} = 1]}_{\text{negl}(n)} + \underbrace{\Pr[\text{Collision}_{H,\mathcal{A}''} = 1]}_{\text{negl}(n)} \leq \text{negl}(n)$$

where $\mathcal{A}'$ is an EUF-CMA adv. against Σ that runs $\mathcal{A}$ as subroutine,
$\mathcal{A}''$ is a collision adv. against H that runs $\mathcal{A}$ as subroutine.

(2) 1. Let $\mathcal{A}'$ be an algo. that aims to distinguish $L_{sign\text{-}real}$ and $L_{sig\text{-}fake}$ s.t.

$\mathcal{A}' \circ L$ (where L is either $L_{sig\text{-}real}$ or $L_{sig\text{-}fake}$)

$(vk, sk) \leftarrow KeyGen$

$(m, \sigma) \leftarrow \mathcal{A}$ s.t. $Versig(m, \sigma) = 1$ (but σ is not generated by $Getsig(m)$ since no one has the sk)

If $Versig(m, \sigma) == 1$:

output "real"

Else:

output "fake"

$$Pr[\mathcal{A}' \text{ wins}] = Pr[\mathcal{A}' \circ L \Rightarrow \text{real} \mid L = L_{sig\text{-}real}] \, Pr[L = L_{sig\text{-}real}]$$
$$+ Pr[\mathcal{A}' \circ L \Rightarrow \text{fake} \mid L = L_{sig\text{-}fake}] \, Pr[L = L_{sig\text{-}fake}]$$
$$= 1 \cdot \frac{1}{2} + 1 \cdot \frac{1}{2} = 1.$$

2. In MAC, we use the same key for Gettag and Checktag ($\sim$ Getsig and Versig). So, the key needs to be shared to use Checktag and we cannot make sure who signed it. In RSA-FDH signing and verification keys are different, so only the owner of sk is able to sign a message.

(3) $\sigma : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$, $m \mapsto m^d \bmod N$ where $m = m_k \| m_{k-1} \| \dots \| m_1 \| m_0$ in byte representation ($m_k \neq 0$, $m_0 \neq 0$)

Let $\mathcal{A}$ be an EUF-CMA adversary s.t. gets $(m, \sigma) \leftarrow Getsig(m)$, $\sigma = m^d \bmod N$, then generates $M' = \{(m^i)^e : i = 1, 2, \dots, 2^{16}\}$. If there is a valid message $m' = (m^e)^j$ in M' for some j, then $\mathcal{A}$ wins by outputting (m', σ') where $\sigma' = \sigma^{e \cdot j}$.

First, note that $Versig(m', \sigma') : (\sigma')^e = (\sigma^{e \cdot j})^e = ((m^d)^{e \cdot j})^e = (m^{ed})^{j \cdot e} = m^{j \cdot e} = m' \bmod N$.

So, $Pr[\mathcal{A} \text{ wins}] \leq Pr[\exists m' \text{ in } M' \text{ that is valid}]$.

Next, observe that m' is valid iff $m' \in \mathbb{Z}_N^*$ and first two bytes of m' are zero.

- $m' \in \mathbb{Z}_N^*$ since $m' = m^e \bmod N$ and $m \in \mathbb{Z}_N^*$.

- Notice that, by assumption, e-th power is a perfect permutation and the ratio of "messages whose first two bytes are 0" to "all messages" is $1/2^{16}$. Therefore, it is expected that among 2^{16} randomly selected messages in $\mathbb{Z}_N^*$, one of them is valid.

More precisely, $\Pr[\exists m' \text{ in } M' \text{ that is valid}] = \Pr\left[\begin{array}{c}\exists m' \text{ whose first two bytes are 00 among} \\ 2^{16} \text{ uni. randomly selected elt. in } \mathbb{Z}_N^*\end{array}\right]$

$$\approx \frac{\binom{\lfloor \phi(N)/2^{16} \rfloor}{1}\binom{\phi(N)-1}{2^{16}-1}}{\binom{\phi(N)}{2^{16}}}$$

where $\phi(N) = |\mathbb{Z}_N^*|$ and $\lfloor \phi(N)/2^{16} \rfloor \leq$ # elements in $\mathbb{Z}_N^*$ starting with 00.

(4) $\hat{H}: \{0,1\}^* \to \mathbb{Z}_N$ (randomly)

Number of outputs of $\hat{H}$ that lie outside of $\mathbb{Z}_N^* = |\mathbb{Z}_N| - |\mathbb{Z}_N^*| = N - \phi(N)$

$= pq - \phi(pq) = pq - (p-1)(q-1) = p+q-1 \approx 2^{1001} - 1$

BONUS: Assume you know an output t of $\hat{H}$ lying outside of $\mathbb{Z}_N^*$, then t is not relatively prime with N. So, $\gcd(N,t)$ is p or q, and $\frac{N}{\gcd(N,t)}$ is p or q as well.

g is 12

① Recall: Order of $g \in \mathbb{Z}_n^*$ is a $\Leftrightarrow$ $g^a = 1$ in $\mathbb{Z}_n^*$. Also, $a \mid |\mathbb{Z}_n^*| = \phi(N)$ (divides; Euler-phi function)

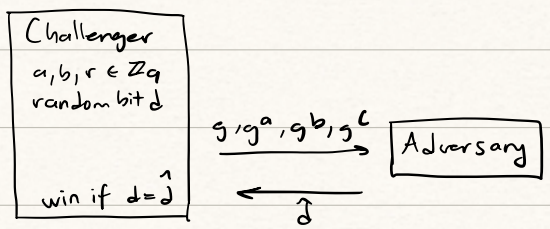
1. g has order 42. So, g generates $\mathbb{Z}_{43}^*$, i.e., $\mathbb{Z}_{43}^* = \{ g^i : i \in \{1, 2, \dots, 42\} \}$.
Since $h \in \mathbb{Z}_{43}^*$, $\exists j \in \{1, 2 \dots, 42\}$ s.t. $g^j = h$, i.e., $11^j = 5$. In the worst case, one needs to try 41 multiplications.

2. g has order 42. So smallest power of g giving 1 in $\mathbb{Z}_{43}^*$ is 42.
In particular, $x = 21$, $y = 14$, $z = 6$. Then smallest power of $g^x = g^{21}$ giving 1 in $\mathbb{Z}_{43}^*$ is 2, i.e., order of g^x is 2. Similarly, order of $g^y = g^{14}$ is and order of $g^z = g^6$ is 7.

3. Discrete logarithm (DL) of h in $\mathbb{Z}_p$ for the base g is a, i.e., $g^a = h \bmod p$.
$\Rightarrow (g^a)^x = h^x \bmod p \Rightarrow (g^x)^a = h^x \bmod p \Rightarrow a \bmod 2$ (order of g^x) is the DL of h^x for base g^x.
Similarly, $a \bmod 3$ is the DL of h^y for the base g^y and $a \bmod 7$ is the DL of h^z for the base g^z. After finding $a = a_1 \bmod 2$, $a = a_2 \bmod 3$, $a = a_3 \bmod 7$, use Chinese Remainder Theorem to recover a. This is more efficient because a_1 can be found with at most 1 multiplication, a_2 with 2 mult., a_3 with 6 mult. In total, we need at most 9 multiplications.

4. Similar to 3, if $p = \prod_{i=1}^{\ell} p_i$ with $p_i < B$, we need at most $\sum_{i=1}^{\ell} (p_i - 1) < \ell (B-1)$ mult

② Recall: DDH: p, q primes s.t. $q \mid p-1$. Fix $g \in \mathbb{Z}_p^*$ s.t. order of g is q.



$$c = \begin{cases} r & \text{if } d = 0 \\ a \cdot b & \text{if } d = 1 \end{cases}$$

1. Observe that g is a generator, so q is $p-1$. We need 2 observations:

1. a, b odd $\Rightarrow a \cdot b$ odd

2. $p-1$ even $\Rightarrow \frac{p-1}{2}$ is integer.

If a is even, then $a = 2 \cdot a'$ and $(g^a)^{\frac{p-1}{2}} = (g^{p-1})^{a} = 1 \mod p$.

If a is odd, then $a = 2a'+1$ and $(g^a)^{\frac{p-1}{2}} = (g \cdot g^{2a'})^{\frac{p-1}{2}} = g^{\frac{p-1}{2}} \cdot \underbrace{(g^{p-1})^{a'}}_{1} \neq 1 \mod p$.

Similarly, we can check parity of b and c.

Notice that $a \cdot b$ is odd iff both a and b is odd, but c is is odd with prob. $\frac{1}{2}$. Given (g, g^a, g^b, g^c), adversary $\mathcal{A}$ computes $(g^{\frac{p-1}{2}}, (g^a)^{\frac{p-1}{2}}, (g^b)^{\frac{p-1}{2}}, (g^c)^{\frac{p-1}{2}})$.

It checks if parities are consistent. If they are, it outputs $c = ab$, i.e., $L_{ddh\text{-}real}$.

$$\Pr[\mathcal{A} \text{ wins}] = \Pr[\mathcal{A} \Rightarrow L = L_{ddh\text{-}real} \mid L_{ddh\text{-}real}] \Pr[L_{ddh\text{-}real}] + \Pr[\mathcal{A} \Rightarrow L = L_{ddh\text{-}ideal} \mid L_{ddh\text{-}ideal}] \Pr[L_{ddh\text{-}ideal}]$$

$$= 1 \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{3}{4}$$

$$Adv(\mathcal{A}) = |\Pr[\mathcal{A} \text{ wins}] - \frac{1}{2}| = \frac{1}{4}.$$

③ 1. Given c_1, c_2, Bob computes $c_1^b = (g^a)^b = (g^b)^a = B^a \mod p$.

If c_1^b is equal to c_2, then $m=0$, otherwise $m=1$.

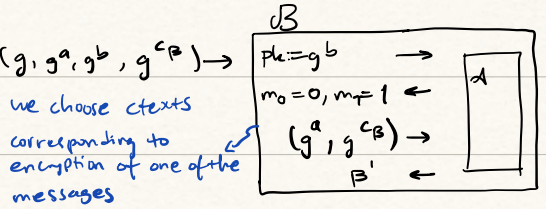
2. Let $\mathcal{A}$ be an adversary that wins the IND-CPA security game with prob. $P > \frac{1}{2}$.

Construct an adv. $\mathcal{B}$ that runs $\mathcal{A}$ as a subroutine to break DDH.

$c_0 := a \cdot b$

$c_1 \sim \mathbb{Z}_p^*$

$\beta \sim \{0,1\}$



- If $c_B = a \cdot b$, $\text{Dec}(sk, (g^a, g^{c_B})) = 0$ indeed $g^{c_B} = g^{ab} = (g^b)^a = B^a$ with prob. p.
- If $c_B \sim \mathbb{Z}_p^*$, $\text{Dec}(sk, (g^a, g^{c_B})) = 1$, with prob. $p\left(1 - \underbrace{\frac{1}{p-1}}_{\text{prob. of random number being } ab}\right) \approx p$.

④ Recall: A group $(\overbrace{G}^{\text{set}}, \overbrace{\cdot}^{\text{operation}})$ is suitable for DH key exchange protocol if it is hard to compute $a \in \{1, \dots, \text{ord}(G)\}$ given G, g, g^b where g is the generator.

1. If g is a generator, any elt. in $\mathbb{Z}_p$ can be written as $k \cdot g \bmod p$ for some $k \in \{0, \dots, p-1\}$.
   Given g, bg for some b, $b \cdot g \cdot g^{-1} = b \bmod p$. (→ modular inverse is easy.)
2. Given $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}$ and $\begin{pmatrix} 1 & a \\ 0 & 1 \end{pmatrix}^b = \begin{pmatrix} 1 & ab \\ 0 & 1 \end{pmatrix}$ for some b, one can recover b by $b \cdot g \cdot g^{-1} = b \bmod p$.
3. Given g and g^b, observe that $g^i(1)$ is distinct for each $i \in \{1, \dots, n\}$. Apply brute-force on the power on the power of g and evaluate it at 1 (WLOG). When $g^i(1) = g^b(1)$, it means $i = b$. This has polynomial time complexity, namely $\underbrace{(n-1)}_{\leq \#g^i(1)\text{'s}} \underbrace{\log n}_{\leq \text{bits to represent } g^i(1)}$.

① 1. Observe that $(x_1, y_1) = (6,2)$ and $(x_2, y_2) = (7,1)$ are distinct points with distinct x-coordinates.

Then $\lambda = \frac{1-2}{7-6} = -1$, $x_3 = \lambda^2 - x_1 - x_2 = 1 - 6 - 7 = 10 \bmod 11$, $y_3 = (6-10)\cdot -1 - 2 = 2 \bmod 11$.

Hence, $P = (x_3, y_3) = (10,2)$

$R = P + Q = (6,2) + (7,1) - (7,1) = (6,2) + O = (6,2)$

2. We want to show $\underbrace{(x_1,y_1) + (x_2,y_2)}_{P \;+\; Q} = \underbrace{(x_2,y_2)}_{Q} + \underbrace{(x_1,y_1)}_{+P}$.

Case 1: $x_1 \neq x_2$

Then $\lambda_{P+Q} = \frac{y_2 - y_1}{x_2 - x_1} = \frac{-(y_1 - y_2)}{-(x_1 - x_2)} = \frac{y_1 - y_2}{x_1 - x_2} = \lambda_{Q+P}$, call this value λ.

Consequently, $x_{3\,P+Q} = \lambda^2 - x_1 - x_2 = \lambda^2 - x_2 - x_1 = x_{3\,Q+P}$, call this x_3.

Now, we only need to show $y_{3\,P+Q} = (x_1 - x_3)\lambda - y_1$ and $y_{3\,Q+P} = (x_2 - x_3)\lambda - y_2$ are equal.

Observe that $y_{3\,P+Q} = y_{3\,Q+P} \iff (x_1 - x_3)\lambda - y_1 = (x_2 - x_3)\lambda - y_2$

$\iff \lambda x_1 - y_1 = \lambda x_2 - y_2$

$\iff y_2 - y_1 = \lambda(x_2 - x_1)$

$\iff \frac{y_2 - y_1}{x_2 - x_1} = \lambda$.

Since $\lambda = \frac{y_2 - y_1}{x_2 - x_1}$, we conclude $y_{3\,P+Q} = y_{3\,Q+P}$.

Case 2: $x_1 = x_2$. Similarly…

② IK: Identity keys, EK: Ephemeral Keys, MK: Medium-term "back stop" keys

$k_1 = DH(IK_A, MK_B)$ → authenticates Alice to Bob

$k_2 = DH(EK_A, IK_B)$ → authenticates Bob to Alice

$k_3 = DH(EK_A, MK_B)$ → provides forward security and post-compromise security

③ 1. Verify (pk, m, σ) where $\sigma = (A, z)$ and $pk = g^s \bmod p$

$e = H(A, m)$

$$A \cdot S^e = g^a \cdot (g^s)^e = g^{a+s\cdot e} = g^z \bmod p$$

$A = g^a$ since σ is correctly generated & $S = g^s$ since pk is correctly generated

since $g \in \mathbb{Z}_p$ is of order q we can reduce the power mod q while the base is mod p & $z = a + s\cdot e$ since σ is correctly generated

Hence, Verify(pk, m, σ) outputs 1.

2. $\mathcal{B}$ picks $g \in \mathbb{Z}_p$ and computes $A = g^a \bmod p$. $\mathcal{D}$ runs $\mathcal{A}$ as a subroutine; inputs $g, A\,(=g^a)$, gets a and inputs $g, S\,(=g^s)$, gets s. Then $\mathcal{B}$ can generate σ for any message using H.

④ 1. In $L_{ddhp\text{-}ideal}$, 2nd to last entries are independent but in $L_{ddhp\text{-}real}$, 5th entry is uniquely determined by 2nd and 3rd, 6th " 2nd and 4th.

2.

Algorithm $\mathcal{B}$:

1. Sample $\beta' \leftarrow \{0,1\}$
2. Sample $c, \iota \leftarrow \mathbb{Z}_q$
3. Take in (g, g^a, g^b, h) // $h = \begin{cases} g^{ab} & // \beta = 0 \\ g^r & // \beta = 1 \end{cases}$
4. if $\beta' = 0$: input $= (g, g^a, g^b, g^c, h, g^{ac})$
   else: input $= (g, g^a, g^b, g^c, h, g^c)$
5. Run $\mathcal{A}$ with input and forward its output $\hat{\beta}$.

if $\beta=0 \,\&\, \beta'=0 \Rightarrow$ input $= (g, \hat{g}, g^b, \hat{g}^c, g^{ab}, \hat{g}^{ac})$

if $\beta=1 \,\&\, \beta'=1 \Rightarrow$ input $= (g, \hat{g}, g^b, \hat{g}^c, \hat{g}^{ab}, g^{ac})$

$$\Pr[B \text{ wins}] = \Pr[\hat{\beta}=\beta] = \sum_{i,j \in \{0,1\}} \underbrace{\Pr[\hat{\beta}=\beta \mid \beta=i \wedge \beta'=j]}_{A_{i,j}} \cdot \Pr[\beta=i \wedge \beta'=j]$$

Observe: $\Pr[A_{00}] = \Pr[A \text{ wins} \mid \mathcal{L}_{\text{ddhp-real}}]$

$\Pr[A_{11}] = \Pr[A \text{ wins} \mid \mathcal{L}_{\text{ddhp-ideal}}]$

$$\begin{aligned}
\Pr[B \text{ wins}] &= \frac{1}{4}\left(\Pr[A \text{ wins} \mid \mathcal{L}_{\text{ddhp-real}}] + \Pr[A \text{ wins} \mid \mathcal{L}_{\text{ddhp-ideal}}]\right) \\
&\quad + \frac{1}{4}\left(\Pr[A_{01}] + \Pr[A_{10}]\right) \\
&\geq \frac{1}{4}\left[\Pr[A \text{ wins} \mid \mathcal{L}_{\text{ddhp-real}}] + \Pr[A \text{ wins} \mid \mathcal{L}_{\text{ddhp-ideal}}]\right) \\
&= \frac{1}{2} \Pr[A \text{ wins dist game } \mathcal{L}_{\text{ddhp-real}} \text{ vs } \mathcal{L}_{\text{ddhp-ideal}}]
\end{aligned}$$

$\Rightarrow$ a factor of $\frac{1}{2}$

3. We are given $\Pr(A \text{ win}) = \alpha$. We know $\Pr(A \text{ win}) = \frac{Adv(A)}{2} + \frac{1}{2} = \alpha$

where $Adv(A) = \left|\Pr[A \Rightarrow \text{real} \mid L = L_{ddhp\text{-}real}] - \Pr[A \Rightarrow \text{real} \mid L = L_{ddhp\text{-}ideal}]\right|$.

Observe that one of these terms is at least $Adv(A)$

Hence, $\Pr(B \text{ win}) \geq Adv(A) \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2}\left(Adv(A) + \frac{1}{2}\right) \geq \frac{\alpha}{2}$.

(5) An adversary can choose $m_0=0$ and $m_1=1$. By getting $c=g^{m_i} \bmod p$ the adv. can easily determine the message used. Indeed $g^{m_0}=g^0=1$ and $g^{m_1}=g^1=g$ and $g\neq 1$.

1. Assume we have an algo. $\mathcal{A}$ that, on input p,q,g,h generates values m,m',c,r,r' s.t. $g^m h^r = g^{m'} \cdot h^{r'} \bmod p$ where $m\neq m'$. In particular, we know that $g^{m-m'} = h^{r'-r} \bmod p$. We know that $\exists a \in \mathbb{Z}_q \setminus \{0\}$ s.t. $g^a = h \bmod p \Rightarrow g^{m-m'} = (g^a)^{r'-r} \bmod p$. Since g is an elt. of order q, we have $m-m' = a(r-r') \bmod q$. We know that $m,m' \in \mathbb{Z}_q$ and $m\neq m' \Rightarrow m-m' \neq 0 \bmod q \Rightarrow$ also $a(r'-r)\neq 0 \bmod q \Rightarrow (r'-r)\neq 0 \bmod q$. Since q is a prime $(r'-r)$ is invertible mod $q \Rightarrow$ we can define $a = \frac{m-m'}{r'-r} \bmod q$ as disc. log. of h for the base g

2. We want to show that $\forall (m,r) \in \mathbb{Z}_q \times \mathbb{Z}_q \quad \exists (m',r') \in \mathbb{Z}_q \times \mathbb{Z}_q$ s.t. $m\neq m', r\neq r'$ and $g^m h^r = g^{m'} \cdot h^{r'} \bmod p$

We fix $m' \in \mathbb{Z}_q / \{m\}$. We want to find r' s.t. $g^m \cdot h^r = g^{m'} \cdot h^{r'} \bmod p$. By using $h=g^a \bmod p$, we write $g^{m+ar} = g^{m'+ar'} \bmod p \Leftrightarrow m+ar = m'+ar' \bmod q \Leftrightarrow m-m'+ar = ar' \bmod q$. Since both g and h have order q, a must be invertible mod q. Then $m-m'+ar = ar' \bmod q$ $\Leftrightarrow (m-m'+ar)a^{-1} = r' \bmod q$. We have just shown that $\forall m' \in \mathbb{Z}_q \quad \exists r' \in \mathbb{Z}_q$ s.t. $r\neq r'$ and $g^m h^r = g^{m'} h^{r'} \bmod p$.