Pontificia Universidad Católica de Chile
Departamento de ciencias de la computacion
IIC3253 – Criptografía y Seguridad Computacional
Primer Semestre del 2025

Ayudantía 01

Aritmetica modular - OTP - Perfect secrecy

Ayudante: Manuel Cifuentes – mecifuentes@uc.cl

Repaso

Problema 1 (Aritmetica modular)

Calcula los siguientes módulos:

- $53 \mod 7 = 4$
- $255 \mod 16 = 15$
- $-14 \mod 6 = 4$

Determina si las siguientes igualdades son verdaderas o falsas:

- $6 \equiv 3 \mod 4$ Falso
- $8 \equiv 77 \mod 3$ Verdadero
- $14 \equiv 29 \mod 5$ Verdadero
- $-4 \equiv 45 \mod 6$ Falso
- $3879 \equiv 8391274129 \mod 10$ Verdadero

Problema 2 (Propiedades)

Explique las siguientes propiedades del módulo y utilícelas en algún problema anterior:

- $a \equiv b \mod n$ si y solo si n divide a $b - a$.

  Esto mas que una propiedad es la definicion de $a \equiv b \mod n$. Se puede emplear sobre $6 \equiv 3$ mód 4 y nos queda "Se cumple si 4 divide a 3-6", lo cual es falso ya que 4 no divide a -3 (Aunque si lo haria si fuese -4).

- Si $b = a \mod n$, entonces $a \equiv b \mod n$.

  Esta propiedad habla basicamente sobre la equivalencia entre las dos formas de trabajar con modulo, la primera se usa mas cuando uno quiere preguntar el resultado de aplicar el modulo y la segunda se utiliza para cuando ya conoces el resultado y quieres dejarlo expresado. Se puede aplicar en $255 \mod 16 = 15$, transpasandolo nos queda como $255 \equiv 15 \mod 16$ y podemos ver que se cumple ya que 16 si divide a (15-255), ya que es 240 y lo divide -15 veces.

- Si $a \equiv b \mod n$ y $c \equiv d \mod n$, entonces $(a + c) \equiv (b + d) \mod n$.

  Propuesto

- $a \equiv b \mod n$ si y solo si $a \mod n = b \mod n$.

  Esta propiedad se puede pensar como, para que se cumpla $a \equiv b \mod n$, se tiene que cumplir que $(b - a) \mod n = 0$. Esto solo se puede cumplir si es que a y b dejan el mismo residuo al ser divididas por n, ya que necesitamos que al final sea 0. Se puede utilizar en $3879 \equiv 8391274129 \mod 10$, remplazando nos queda $3879 \mod 10 = 8391274129 \mod 10$, lo cual es verdadero ya ambos dan modulo 9.

- $(a + b) \mod n = [(a \mod n) + (b \mod n)] \mod n$.

  Similar a la idea de arriba, esta vez con suma, se recomienda imaginar esta propiedad utilizando n=10, asi queda mas simple de ver que si utilizas las unidades para sumarse y luego les dejas el modulo. Es lo mismo que si hubieses sumado los numeros completos y luego sacado el modulo, ya que no importa que numero hayan luego de la unidad para el caso de n=10.

Problema 3 (OTP)

Demuestre, utilizando propiedades del módulo, que la siguiente igualdad se cumple:

$$\mathbf{Dec_k(Enc_k(m)) = m}$$

Solución

Usando la definicion vista en clases tenemos que $Enc_k(m) = (m + k) \mod N$ y $Dec_k(c) = (c - k) \mod N$, con esto podemos rescribir la igualdad como

$$Dec_k(Enc_k(m)) = ((m + k) \mod N - k) \mod N$$

Sabiendo que siempre se cumple que $k < N$ se cumple que $k = k \mod n$, remplazando

$$Dec_k(Enc_k(m)) = ((m + k) \mod N - k \mod N) \mod N$$

Luego utilizando la ultima propiedad del problema 2 (que es analoga entre resta y suma), se tiene que

$$\begin{aligned}((m + k) \mod N - k \mod N) \mod N &= ((m + k) - k) \mod N \\ &= (m + k - k) \mod N \\ &= m \mod N \\ &= m\end{aligned}$$

Problema 4 (Implementacion OTP)

Complete el siguiente código para implementar el cifrado OTP, tanto para clave completa como para clave incompleta:

```
def encrypt(msg,key):
  msg = msg.upper()
  key = key.upper()
  final = ""
  if len(key)>=len(msg):
    for i in range(len(msg)):
      val = ord(msg[i]) - 65
      key_val = ord(key[i]) - 65
      final += chr((val + key_val) % 26 + 65)
  else:
    for i in range(len(msg)):
      val = ord(msg[i]) - 65
      key_val = ord(key[i%len(key)]) - 65
      final += chr((val + key_val) % 26 + 65)

  return final

def decrypt(msg,key):
  msg = msg.upper()
  key = key.upper()
  final = ""
  if len(key)>=len(msg):
    for i in range(len(msg)):
      val = ord(msg[i]) -65
      key_val = ord(key[i]) - 65
      final += chr((val - key_val) % 26 + 65)
  else:
    for i in range(len(msg)):
      val = ord(msg[i]) - 65
      key_val = ord(key[i%len(key)]) - 65
      final += chr((val - key_val) % 26 + 65)
  return final
```

Con el código generado, realice lo siguiente:

- Encripte "ElAsesinoEsShmebulock" con la clave "noconfieseneltriangulo".

  El mensaje encriptado queda como RZCGRXQRGIFWSFVJUYUWV.

- Desencripte "PZSJXJSYGRTSUGAGOUMSSQ" con "claveseguraasisi".

  El mensaje desencriptado queda como NOSOTROSMATAMOSEDUROAM

Problema 5 (Perfect secrecy)

Desafío: Tienes que desencriptar este mensaje:

"OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ"

Sabes que fue encriptado con la técnica OTP. Como ayuda, sabes que este tipo de mensajes siempre comienza con la palabra "orden".

Para los siguientes dos casos, discute y demuestra:

- Sabes que el largo de la llave es igual al tamaño del mensaje. ¿Se puede desencriptar? Si la respuesta es sí, inténtalo?.

  Si el largo de la llave es igual al largo del texto, OTP es teóricamente inquebrantable sin la llave, porque el texto cifrado no contiene suficiente información para reducir el número de mensajes posibles. Aqui se cumplen los principios de perfect secrecy del OTP: sin la clave exacta, la probabilidad de que cualquier mensaje sea el correcto es la misma, lo que hace imposible el descifrado aun teniendo esta informacion adicional.

- Sabes que el largo de la llave es menor al tamaño del mensaje. ¿Se puede desencriptar? Si la respuesta es sí, pruébalo.

  Como tenemos la informacion que es parcial, se puede ingeniar un poco mas una posible respuesta. Con la informacion de que el mensaje comienza con "orden" se puede hardcodear el inicio de la llave, llegando al siguiente resultado:

```
decrypt("OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ","ABZCY")

'ORDENGWIRDLPRYBZBCOTMQSXFFPHYJGNBQBKOQSSLAIZQUNEPGZ'
```

  Se puede ver que al utilizar la llave "ABZCY" se logra generar la primera parte que conociamos. Desde aqui usaremos una de las estrategias mas comunes para romper otp cuando sabemos que la llave es parcial y el mensaje es pequeño. Vamos agregando una letra "A" a la key hasta ver un mensaje que pareciera tener sentido. Lo que nos genera los siguientes resultados:

```
decrypt("OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ","ABZCYA")

'ORDENGXGUZNQQZAXEBQQNPTZDERECHGNBQBKPOVONBHAPSQDRDA'
```

  Pareciera ser que se nos genera "DERECH" en el centro, pero no hay un mensaje claro en el resto.

```
decrypt("OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ","ABZCYAA")

'ORDENGXHSCJSQAZYDZSRMRQABHQGAGHMCSZKOQSSLBHAPSQDREY'
```

Tenemos "REY" al final, pero nada en el resto del mensaje.

```
decrypt("OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ","ABZCYAAA")

'ORDENGXHTAMOSAZZCARPORRZDERECHGOARAIRPUQLAIZQUODRDA'
```

De nuevo, obtenemos la palabra "DERECH" en el centro y parecieramos tener mayores bloques con texto legible, probaremos con este numero. Para el caso de 3 letras, tenemos un total de 17,576 combinaciones posibles. Hoy en dia esto es un numero sencillamente computable, cuando ya tenemos un mensaje parcial como este se pueden hacer 2 cosas, fuerza bruta o probar las mejores combinaciones. Fuerza bruta con GPT es posible, pero para efectos de la ayudantia intentaremos el otro acercamiento.

Podemos probar las dos opciones que tenemos al medio, generar "DERECHA" y "DERECHO".

```
print(decrypt("OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ","ABZCYAGA"))
print(decrypt("OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ","ABZCYASA"))

ORDENGRHTAMOSATZCARPORLZDERECHAOARAIRPOQLAIZQUIDRDA
ORDENGFHTAMOSAHZCARPORZZDERECHOOARAIRPCQLAIZQUWDRDA
```

Fijandonos en el final, la mejor opcion es cuando generamos "DERECHA", "DERECHO" genera una palabra invalida "IZQUWDRDA", que a simple vista pareceria ser izquierda, si intentamos una llave que la genere.

```
decrypt("OSCGLGXHTBLQQAZZCBQRMRRZDFQGAHGOASZKPPUQLBHBOUODREZ","ABZCYAGZ")

'ORDENGRITAMOSATACARPORLADERECHAPARAIRPORLAIZQUIERDA'
```

Con esto logramos obtener un mensaje y podemos verificar como OTP con llave parcial no te asegura perfect secrecy, ya que si tenemos informacion adicional del mensaje original, hay mayores posibilidades de encontrar el mensaje