MA-53G Intr. a la Criptografía	**Semestre Otoño, 1998**

Tarea No. 1 : 13 de Abril

Profesor: Marcos Kiwi — *Fecha de entrega: 27 de Abril*

De haber clarificaciones se publicarán en el newsgroup del curso.

PROBLEMA 1: Sea $C = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ un criptosistema tal que $I_C \subseteq \mathbb{N}$, $\mathcal{K} = \bigcup_{m \in I_C} \mathcal{K}_m$, y todas las funciones de encripción asociadas a claves en $\mathcal{K}_m$ tienen el mismo dominio, $\mathcal{P}_m$, y el mismo recorrido $\mathcal{C}_m$. Cualquiera sea $m \in I_C$, sean $\mathcal{F}_{\mathcal{P}_m}$ y $\mathcal{F}'_{\mathcal{P}_m}$ dos distribuciones sobre $\mathcal{P}_m$, y sea $\mathcal{G}_{\mathcal{K}_m}$ una distribución sobre $\mathcal{K}_m$. Pruebe que si C es un criptosistema incondicionalmente seguro cuando el texto puro y las claves se eligen de acuerdo a las distribuciones $\mathcal{F}_{\mathcal{P}_m}$ y $\mathcal{G}_{\mathcal{K}_m}$ respectivamente, entonces C también es un criptosistema incondicionalmente seguro cuando el texto puro y las claves se eligen de acuerdo a las distribuciones $\mathcal{F}_{\mathcal{P}'_m}$ y $\mathcal{G}_{\mathcal{K}_m}$ respectivamente.

PROBLEMA 2: Sea k una clave secreta compartida por Alicia y Roberto. Roberto desea una garantía que los mensajes que recibe de Alicia realmente provienen de ella. Para lograr esto Alicia envía sus mensajes junto con un MAC que depende de la clave k.

Se dice que un MAC es seguro con respecto a ataques de texto elegido si un adversario, Eva, una vez obtenido los MAC, $MAC_k(m_i)$, para n mensajes distintos de su elección $m_1, \ldots, m_n$, no podrá producir un *nuevo* mensaje $m \notin \{m_1, \ldots, m_n\}$ y su respectivo MAC, $MAC_k(m)$.

Sea f_k: $\{0,1\}^l \to \{0,1\}^l$ una función de encripción con clave k (por ejemplo si $l = 64$ la función f podría ser DES). Sea además

$$MAC_k(m) \quad = \quad f_k(f_k(\ldots f_k(f_k(m_1) \oplus m_2) \oplus \ldots m_{b-1}) \oplus m_b)\,,$$

donde $m = m_1 \ldots m_b$ es un mensaje de bl bits, $m_i \in \{0,1\}^l$ para todo $i \in \{1, \ldots, b\}$, y $\oplus$ denota el XOR de dos secuencias de bits.

1. Muestre que si los mensajes m tienen largo variable (i.e., b es un entero positivo cualquiera que no esta fijo), entonces se puede diseñar un ataque de texto plano elegido contra el MAC antes definido.

   Indicación: Elija dos textos planos de l bits y a partir del MAC de los mismos construya el MAC de un texto plano de $2l$ bits.

2. Suponga que el largo de los mensajes, m, cuyo MAC se desea calcular siempre tienen largo menor que 2^l. Una forma de evitar el ataque del punto anterior sería aumentar m en l bits de forma que los últimos l bits representen el largo de m expresado en binario y luego calcular el MAC del mensaje así obtenido. Pruebe que todavía es posible montar un ataque de texto plano elegido contra esta modificación del MAC antes definido.

PROBLEMA 3:

3.1.- Sea n un entero positivo. Demostrar que $\sum_{d\,|\,n} \phi(d) = n$.

Indicación: Considere la factorización $n = \prod_i p_i^{e_i}$ donde los p_i son primos y los e_i enteros positivos.

3.2.- Pruebe que no existe un polinomio en dos variables $f(x, y)$ tal que $f(m, n) = \mathrm{mcd}(m, n)$ para todo m y n enteros positivos.

Indicación: Considere la función $g(m, n) = \mathrm{mcd}(m, n) - 1$ y observe que se anula para m y n primos.

PROBLEMA 4: Sean a y b dos enteros positivos tales que $a > b$. Se desea determinar si existen enteros positivos i y j tales que $a^i = b^j$.

Considere el siguiente algoritmo para este problema:

1. Se verifica si $b \,|\, a$. Si no, se retorna *no*.
2. De otra manera, se reemplaza (a, b) con $(a/b, b)$ si $a \geq b^2$ y con $(b, a/b)$ si $a < b^2$.
3. Si eventualmente el par $(a', 1)$ es alcanzado, se retorna *si*.
4. De lo contrario, se itera nuevamente.

Muestre que este algoritmo usa $O\left(\lg^2(a)\right)$ operaciones de bits.

PROBLEMA 5: El objetivo de este problema es diseñar y escribir un programa en C, C++ o Java que implemente el algoritmo rápido de multiplicación de **Karatsuba**.

El programa deberá poder multiplicar números de hasta 8192 bytes (65536 bits) con un resultado máximo de hasta 16384 bytes. Cada uno de dichos números deberán almacenarse en una estructura de nombre `bigNumber`[1].

El programa deberá incluir una función de nombre `multiplicar`, que reciba como argumentos dos variables de tipo `bigNumber` y retorne un valor de tipo `bigNumber`. Asimismo, deberá poseer un ejemplo de uso en el programa principal donde deberá estar claramente indicada la manera de asignar y mostrar en pantalla un número de tipo `bigNumber`[2].

En la programación del algoritmo no debe usar ninguna biblioteca preexistente que provea funciones explícitas de exponenciación o de manejo números *grandes*. Utilice sólo *shifts*, adiciones y multiplicación de enteros usual (4 u 8 bytes).

Modo de entrega: El código fuente del programa y una breve explicación de su funcionamiento deberá ser enviada por e–mail al profesor auxiliar (ahevia@dcc.uchile.cl). Sólo se corregirán las tareas recibidas hasta las 23:59 hrs. del día de entrega.

Nota: Los autores de la implementación más rápida recibirán como premio \$5000.

[1] Por eficiencia, se recomienda utilizar una estructura de tipo arreglo de números enteros para este efecto.

[2] Para la corrección se modificará dicho ejemplo testeándola con números grandes de producto conocido.