IIC3253: Criptografía y Seguridad Computacional - Tarea #4

Raimundo Herrera - rjherrera@uc.cl
Colaborador(a):

22 de junio de 2018

Problema 1

1. **Muestre que dado un texto cifrado ElGamal $\langle c_1, c_2 \rangle$ para un mensaje m es posible construir texto cifrado $\langle c_1', c_2' \rangle$ para el mensaje $\alpha \cdot m$ para cualquier α en $\mathbb{G}$.**

   En ElGamal el algoritmo de encriptación, considerando un grupo de orden q, un mensaje m, y una llave secreta x, retorna $\langle c_1, c_2 \rangle$ construidos de la siguiente forma:

   - $r \xleftarrow{\$} \mathbb{Z}_q$
   - $c_1 = m \cdot g^{xr}$
   - $c_2 = g^r$

   Así, para el mensaje $\alpha \cdot m$, basta con construir dicho par de la siguiente forma:

   - $r \xleftarrow{\$} \mathbb{Z}_q$
   - $c_1' = \alpha \cdot c_1 = \alpha \cdot m \cdot g^{xr}$
   - $c_2' = c_2 = g^r$

   Con esto la decriptación es inmediata, pues dado el par $\langle c_1', c_2' \rangle$ se tiene:

   $$\begin{aligned} m^* &= c_1' \cdot (c_2'^x)^{-1} \\ &= \alpha \cdot (c_1 \cdot (c_2^x)^{-1}) \\ &= \alpha \cdot m \end{aligned}$$

   Teniendo con esto que para el mensaje $\alpha \cdot m$, el par $\langle c_1, c_2 \rangle$ es un texto cifrado válido para cualquier α en $\mathbb{G}$.

2. **Muestre que también se puede hacer con la restricción $c'_1 \neq c_1$ y $c'_2 \neq c_2$.**

   Si se agrega esa restricción, lo que se puede hacer es multiplicar ambos componentes por un elemento aleatorio del grupo, y luego considerar los casos en los que aún con dicha consideración, el nuevo elemento c'_1 o c'_2 sean iguales a los primeros. Para esto cambiamos el algoritmo para construir el par de la siguiente forma:

   - $r \xleftarrow{\$} \mathbb{Z}_q$
   - $r_1 \xleftarrow{\$} \mathbb{Z}_q - \{0\}$ (para evitar el caso en que $r_1 = 0$, ya que $g^0 = 1$)
   - $c'_1 = \alpha \cdot c_1 \cdot (g^x)^{r_1} = \alpha \cdot m \cdot g^{x(r+r_1)}$
   - $c'_2 = c_2 \cdot g^{r_1} = g^{r+r_1}$

   Notar que para el algoritmo anterior puede suceder que $c_1 = c'_1$ ya que tanto α como $g^{x \cdot r_1}$ pertenecen al grupo, por lo tanto podrían anularse entre ellos. Para evitar esto el algoritmo debe considerar generar nuevamente r'_1 en caso de que ocurra.

   Para decriptar se tiene lo siguiente:

$$\begin{aligned} m^* &= c'_1 \cdot (c'^x_2)^{-1} \\ &= \alpha \cdot m \cdot g^{x(r+r_1)} \cdot (g^{x(r+r_1)})^{-1} \\ &= \alpha \cdot m \end{aligned}$$

   Mostrando que, incluso cumpliendo la restricción, se puede hacer lo anterior.

3. **Discuta si la maleabilidad es siempre una propiedad negativa.**

   A priori parece ser una propiedad negativa, porque se podría utilizar por un agente malicioso para modificar un mensaje enviado por otra persona aunque este este encriptado. Notar que el hecho de que pueda modificar el mensaje no implica que lo pueda leer, de hecho no puede, solamente puede modificarlo, y esto podría ser grave por ejemplo si es capaz de aumentar la cantidad en un mensaje correspondiente a una transferencia de dinero.

   Sin embargo, es una propiedad deseable para esquemas de encriptación homomórfica, como Pallier. Una aplicación donde la maleabilidad es una propiedad útil, es en los esquemas donde se requiere ver información agregada de los datos, y no desencriptar cada uno de ellos, como para obtener los resultados de una votación sin comprometer la secretitud de quienes votan, sino que obteniendo la suma de los votos al aplicar una transformación a la encriptación dado el homomorfismo por la maleabilidad.

Problema 2

Un esquema de cifrados de llave pública es *key-private* si el texto cifrado no revela información alguna sobre la llave pública. **Argumente si ElGamal y RSA son *key-private* o no.**

1. **ElGamal**: ElGamal es *key-private* si se considera que tanto el grupo $\mathbb{G}$, como su generador g y el orden q, son conocidos. Esto ocurre debido a que el texto cifrado se compone de dos elementos c_1 y c_2 y ambos corresponden a elementos del grupo. El elemento c_2 es evidente pues corresponde a directamente g^r, el caso de c_1 es también evidente ya que m también es parte del grupo, por lo que $m \cdot (g^x)^r$ también lo es. Así, ambos elementos son g^y para algún y. De este modo, al ser ambos elementos aleatorios en el grupo, y el grupo ser conocido, no se obtiene ninguna información sobre la llave.

   En caso de que dichos elementos no sean conocidos el esquema deja de ser *key-private* ya que revela dos elementos del grupo, cosa que no era conocida.

2. **RSA**: No es *key-private* ya que el mensaje encriptado se construye a partir de un módulo, esto es $c = m^e \mod N$. De este modo, sabemos que hay una cota para dicho texto cifrado, y esta es N. Para comprobar que esto rompe la propiedad, es claro ver con el siguiente ejemplo: si un adversario tiene dos candidatos válidos a llave pública, el par (N_1, e_1) y (N_2, e_2), con $N_1 < N_2$, si el mensaje encriptado c es mayor a N_1, el adversario sabe inmediatamente que la llave es el segundo par, por lo que el texto cifrado sí revela información sobre la llave.

Problema 3

DDH en $\mathbb{Z}_p^*$ con p primo.

1. **Describa un algoritmo de tiempo polinomial para saber si un elemento y pertenece a $\texttt{QR}$.**

   Para describir dicho algoritmo, apoyaré mi procedimiento en el conocido Criterio de Euler[1]. Este enuncia lo siguiente:

   Sea p es un número primo y a un entero positivo, tal que $p \nmid a$, entonces a es un residuo cuadrado de p sí y solo sí $a^{\frac{(p-1)}{2}} \equiv 1 \pmod{p}$

   De este modo, el algoritmo debe computar $y^{\frac{(p-1)}{2}} \mod p$ ya que y al ser parte de $\mathbb{Z}_p^*$, cumple inmediatamente que $p \nmid y$, de modo que si dicho cómputo es igual a 1, pertenece a $\texttt{QR}$, y si es -1, no pertenece.

   El algoritmo corre en tiempo polinomial puesto que se compone de una exponenciación modular, y es sabido que existen algoritmos para la exponenciación modular que se pueden realizar en tiempo polinomial.

2. **Construya un algoritmo PPT que quiebre DDH en $\mathbb{Z}_p^*$. Es decir, dado un generador g para $\mathbb{Z}_p^*$, g^a, g^b y T, construya un algoritmo que decida si $T = g^{ab}$, o T es uniforme en $\mathbb{Z}_p^*$.**

   Considerando el algoritmo descrito anteriormente, es posible saber si un elemento pertenece a los residuos cuadrados de $\mathbb{Z}_p^*$ o no. De este modo, dada la tupla $(\mathbb{Z}_p^*, g^a, g^b, T)$, es posible saber si g^a y g^b pertenecen a $\texttt{QR}$.

   Por otro lado, dada la pertenencia o no a $\texttt{QR}$ de g^a, g^b, es posible saber, si un potencial g^{ab} pertenecería a QR. Existen para eso 3 casos:

   a) $g^a, g^b \in \texttt{QR}$.

   b) $g^a, g^b \notin \texttt{QR}$.

   c) $g^a \in \texttt{QR}, g^b \notin \texttt{QR}$ (el inverso es el mismo caso).

   Para (a) es evidente que $g^{ab} \in \texttt{QR}$ ya que como g^a sí pertenece, entonces existe un x tal que $g^a = x^2 \mod p$. De este modo, si se considera $(g^a)^b$ se llegará a que $g^{ab} \in \texttt{QR}$ ya que con x^b se cumple la condición. Para el caso (c) la demostración es análoga pues no depende del valor de la segunda potencia.

   Para (b) es evidente que $g^{ab} \notin \mathbb{Z}_p^*$ ya que por el Criterio de Euler se tiene que $g^{\frac{a(p-1)}{2}} \equiv -1 \pmod{p}$, de este modo, es trivial que $g^{\frac{ab(p-1)}{2}} \equiv -1 \pmod{p}$, dado que b es impar (sino sería un residuo, y $-1^b = -1$), teniendo entonces que $g^{ab} \notin \texttt{QR}$.

[1] Koshy, T. (2007). *Elementary number theory with applications*. Academic press.

Así, basta con construir un algoritmo PPT que use el algoritmo polinomial descrito en el ejercicio anterior, de modo que cuando observe que se da una combinación inviable, retorne que viene uniformemente de $\mathbb{Z}_p^*$, esto es cuando ocurra que:

a) $g^a, g^b \in \texttt{QR}$ y $g^{ab} \notin \texttt{QR}$

b) $g^a \in \texttt{QR}$ y $g^b, g^{ab} \notin \texttt{QR}$

c) $g^a, g^b \notin \texttt{QR}$ y $g^{ab} \in \texttt{QR}$

Dicho de otro modo, que retorne 0 en esos casos, y que retorne 1, en cualquier otro caso.

Así, considerando $\texttt{inQR}(x)$ como el algoritmo que retorna `true` si un elemento x pertenece a los residuos cuadrados de $\mathbb{Z}_p^*$ y `false` en caso contrario, el algoritmo $\texttt{Distinguisher}(g, g^a, g^b, T)$ sería:

- $x = \texttt{inQR}(g^a)$
- $y = \texttt{inQR}(g^b)$
- $z = \texttt{inQR}(T)$
- if x `OR` y
  if $!z$ return 1
- if $!x$ `AND` $!y$
  if z return 1
- return 0

De este modo solo queda considerar el análisis de probabilidades, sin embargo, es claro ver que los casos en que retorna 1 son la mitad de los casos posibles, de este modo, cuando T es uniforme, la probabilidad $\Pr[D(g, g^a, g^b, T) = 1] = 1/2$

Por otro lado, cuando T viene de g^{ab}, entonces el análisis es más complejo, sin embargo, no es difícil concluir que la probabilidad de distinguir cuando ocurre lo anterior es 1.

De este modo,

$$\Pr_{T \leftarrow \mathbb{Z}_p^*}[D(g, g^a, g^b, T) = 1] - \Pr_{T \leftarrow g^{ab}}[D(g, g^a, g^b, T) = 1] = 1 - 1/2$$
$$= 1/2 > negl(\lambda)$$

Problema 4

Sea Π un protocolo de intercambio de llaves para 2 participantes en donde el elemento acordado a un grupo $\mathbb{G}$, y en donde cada participante envía un solo mensaje (El protocolo Diffie-Hellman visto en clases cumple esta propiedad).
Muestre como transformar Π en un esquema de cifrados de llave pública.

Para esto, consideremos que gracias a tener este protocolo de intercambio de llaves, cada participante tiene 2 algoritmos cada uno, el primero que retorna un estado privado y un mensaje público, y el segundo algoritmo que retorna, a partir del estado privado y el mensaje ajeno, una llave. Dicha llave es igual para ambos participantes. Llamemos al primer algoritmo `Estado` y al segundo `Llave`

Así, para transformar este protocolo en un esquema de cifrados, se considera el `Estado` como el `Gen`, y que su retorno es la llave privada y la llave pública, correspondiendo a el estado privado y el mensaje público respectivamente.

El algoritmo de encriptación `Enc` para nuestro esquema lo que hace es llamar a `Llave`(*privado*, *ajeno*) obteniendo una llave, donde privado es nuestra nueva llave privada, generada anteriormente, y ajeno es la llave pública del destinatario, y retorna entonces el resultado de encriptar nuestro mensaje m con esa llave, utilizando un esquema de encriptación simétrico, ya que sabemos que vamos a compartir la clave con el destinatario, de modo que ha de ser simétrico.

El algoritmo de decriptación `Dec` para este esquema lo que hace es llamar a `Llave` con el secreto propio y el mensaje ajeno, que a estas alturas corresponde a la llave pública, y obtenida dicha llave puede decriptar ya que el cifrado es simétrico.

Así el protocolo de intercambio de mensajes sería de la siguiente forma:

- Alice genera su secreto S_a y llave pública P_a como se describe y envía su llave pública a Bob
- Bob genera su secreto S_b y llave pública P_b como se describe.
- Bob encripta simétricamente con la llave obtenida a partir de S_b y la llave de Alice P_a el mensaje m.
- Bob envia el par (mensaje encriptado, P_b).
- Alice a partir de su secreto S_a y la llave pública de Bob P_b obtiene la llave (igual a la usada para encriptar).
- Alice decripta el mensaje encriptado con dicha llave obteniendo m.

La gracia radica en ser capaz de mapear *estado privado* a *llave privada*, y *mensaje público* a *llave pública*. Después utilizar el algoritmo `Llave` para que a partir de una llave pública ajena y la llave privada propia se pueda obtener el mismo secreto que el otro participante que hará lo mismo, pero con los datos invertidos.

Problema 5

Con $\mathbb{G}$, y dado $\mathbb{G}^{\mathbb{T}}$ y e tal que $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}^{\mathbb{T}}$ con las propiedades descritas:

1. **Demuestre que el supuesto DDH no se cumple en $\mathbb{G}$.**

   Si no se cumple el supuesto entonces se puede distinguir entre T aleatorio y T proveniente de g^{ab} con probabilidad no negligible. Dada la definición de e es fácil ver que $e(g^{ab}, g) = e(g^a, g^b)$, ya que $e(g^a, g^b) = e(g, g)^{ab}$. De este modo se puede definir un algoritmo distinguidor $\texttt{Distinguisher}(g, g^a, g^b, T)$ de la siguiente forma:

   - if $e(T, g) = e(g^a, g^b)$ return 1
   - else return 0

   Es sencillo ver que el análisis de probabilidades se cumplirá puesto que cuando el elemento T provenga de g^{ab} siempre retornará 1, y en caso de que T sea uniforme el algoritmo retornará 1 solo una fracción de las veces, siendo esa fracción exáctamente la probabilidad de que ese elemento uniforme sea exáctamente igual a g^{ab} y eso ocurre 1 sobre el orden del grupo, esto es $1/q$. Así:

   $$\Pr_{T \leftarrow g^{ab}}[D(g, g^a, g^b, T) = 1] - \Pr_{T \xleftarrow{\$} \mathbb{G}}[D(g, g^a, g^b, T) = 1] = 1 - 1/q > negl(\lambda)$$

   Por lo tanto al existir dicho algoritmo que distingue con probabilidad no negligible, el supuesto no se cumple en dicho grupo.

2. **Proponga un protocolo de intercambio de llaves para 3 participantes en donde el output de los participantes es un elemento en $\mathbb{G}^{\mathbb{T}}$ (asumma que a pesar de que DDH no se cumple en $\mathbb{G}$, DLog sigue siendo un supuesto razonable).**

   El protocolo de intercambio a proponer debe basarse en que DLog sigue es razonable. Para esto la intuición dice que deberá seguir existiendo un secreto de cada participante y este secreto utilizarse para elevar lo obtenido de los otros participantes, siendo entonces imposible obtener el secreto a partir de la llave compartida, por DLog.

   El protocolo sería el siguiente: cada participante i con $i \in \{1, 2, 3\}$ genera su llave secreta uniformemente escogida en $\mathbb{Z}_q$, la llamaré s_i. Luego, cada participante envía a los demás g^{s_i}. De este modo para obtener el secreto *shared* compartido por los 3, el procedimiento es el siguiente:

   $$\begin{aligned} shared &= e(g^{s_j}, g^{s_k})^{s_i} \\ &= e(g, g)^{s_j s_k s_i} \end{aligned}$$

Donde j y k son los índices de los interlocutores e i es el índice propio, este procedimiento lo realiza cada uno de los participantes con la configuración de i, j y k que corresponda.

Además, como $e(g, g) = g_t$, entonces

$$shared = g_t^{s_j s_k s_i}$$

Por lo tanto, si DLog es difícil, como lo que se comparte es g^{s_i}, no será posible obtener s_i, por lo que cumple con las propiedades deseadas.

Problema 6

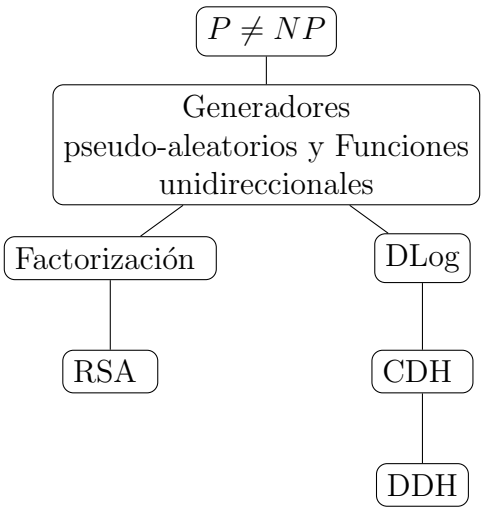
Ordene los siguientes supuestos de acuerdo a que tan fuertes/débiles son:
$P \neq NP$, DDH, RSA, DLog, CDH, existen funciones unidireccionales, existen generadores pseudo-aleatorios, factorización.

En primer lugar notar que un supuesto A es más fuerte que otro B si existe una reducción por la cual si no se cumple B, entonces A tampoco se cumple.

De esta forma, dicho de forma intuitiva el más fuerte de los supuestos es el que se construye supuesto tras supuesto, y el más débil el que no se construye sobre tantas suposiciones.

Para este ejercicio diagramaré un árbol que muestre la fuerza de los supuestos dado que según mi respuesta, hay ciertos supuestos que no son comparables entre ellos, por lo que no es posible ordenarlos en una secuencia.

El árbol es el siguiente:



Sin embargo, hay que notar que la existencia de funciones unidireccionales se da sí y solo sí existen generadores pseudo-aleatorios, por lo que son equivalentes y se podrían considerar un mismo nodo. No obstante, no ocurre lo mismo para los otros supuestos que comparten altura. Ahora, veamos las razones detrás de dicho ordenamiento.

Si se rompe el supuesto $P \neq NP$ entonces los generadores pseudo-aleatorios no podrían existir, puesto que si $P = NP$ entonces existe un algoritmo en tiempo polinomial (el algoritmo exponencial que se deduce de probar todo) que será capaz de distinguir con probabilidad no negligible si un elemento viene de un generador o es verdaderamente aleatorio.

Por otro lado, si $P = NP$, entonces no existen las funciones unidireccionales, esto porque para obtener la inversa de una función, se puede siempre buscar la permutación que lo realiza, sin embargo esto se puede realizar en tiempo no polinomial, si

se rompe el supuesto, entonces se podría computar la función inversa en tiempo polinomial, por lo que no existirían estas funciones unidireccionales. Notar que ambas afirmaciones anteriores son equivalentes, existe el teorema que señala que *FU sí y solo sí PRGs*.

Por la rama derecha del árbol, DLog asume que las funciones unidireccionales existen. Si no existieran se podría argumentar que $f(x) = g^x$ es una función y por ende encontrar la inversa significaría obtener x a partir de g^x que justamente es el contrario al supuesto.

Siguiendo por este lado, se tiene que CDH es más fuerte que DLog, esto porque en CDH se asume que dado g^a y g^b es difícil computar g^{ab}, que es el secreto común, sin embargo, si DLog fuera fácil, a partir de g^a y g^b se podría obtener a y b, teniendo entonces que calcular g^{ab} sería fácil.

Por otro lado, DDH es más fuerte que CDH ya que DDH asume que g^{ab} es pseudoaleatorio, o bien que no se puede distinguir de un elemento aleatorio, esto no sería cierto si CDH no se cumple, ya que si se es capaz de computar fácilmente g^{ab} a partir de g^a y g^b, entonces se puede `distinguir` que cierto elemento vino del cálculo g^{ab} y no de una obtención aleatoria.

Por el lado izquierdo del árbol, se tiene que la factorización es más fuerte que los PRG, y equivalentemente se tiene que la factorización es más fuerte que las funciones unidireccionales. Es claro ver que si no existen las funciones unidireccionales la factorización tampoco es difícil ya que el hecho de que la factorización sea difícil asume que la multiplicación de ciertos factores es una función unidireccional, dicho de otro modo, encontrar x e y dado $x \cdot y$ es difícil. Si se puede encontrar la inversa de cualquier función en tiempo polinomial, entonces la factorización en particular también es fácil bajo ese contexto.

Siguiendo por este lado, RSA asume que factorizar es difícil, ya que el N utilizado por el algoritmo de encriptación corresponde a $P \cdot Q$, ambos factores primos, por lo que si se pudiera obtener dichos factores fácilmente, entonces se podría encontrar el inverso de la llave privada e y por ende se podría decriptar el mensaje.