NOMBRE: Lucas Aguilera Gómez SIGLA: IIC3253

N° ALUMNO: 20643039

Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC3253 — Criptografía y seguridad computacional — 1' 2025

Tarea 2 – Respuesta Pregunta 2

(a) Para resolver esto basta con mostrar un adversario de tiempo polinomial A que puede ganar el juego PreImageModification(1^n) con una probabilidad NO despreciable.

Sabemos que la construcción Merkle-Damgård transforma una función de compresión: $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ en una función de hash: $h_n : \{0,1\}^n \to \{0,1\}^n$.

Consideremos esta función h_n y un valor inicial de IV $\in \{0,1\}^n$. Sabemos que inicialmente, dado un mensaje m, se genera el padding $Pad(m)$ para obtener la secuencia de bloques de tamaño n (longitud del mensaje sea múltiplo de n) y así se pueda dividir en bloques m = $m1||m2||...||mk$, donde cada uno es de tamaño n bits. Luego el hash se calcula de forma iterativa en ciclos de compresión:

- $H_0 = IV$
- $H_i = F(H_{i-1}, M_i)$ para todo i = 1,2,..., k.
- $h_n(m) = H_k$

Antes de proseguir, analicemos la función de padding del enunciado:

Dado un mensaje m con longitud $|m|$ entonces $l = |m| \mod n$, y sea $m_1 \in \{0,1\}^n$ la representación como string binario del numero $|m| \mod 2^n$. Si $l = 0$, entonces Pad(m) = $m||m_1$. Si $l > 0$, entonces Pad(m) = $m||10^{n-l-1}||m_1$.

Ahora, sabemos que en el contexto del juego, el verificador debe escoger un x de tamaño n, por lo que sabemos que: $|x| = n$ y $l = 0$, por tanto $Pad(x) = x||x_1$.

Definamos ahora un adversario A de la siguiente manera:

Dado un mensaje $x \in \{0,1\}^n$, el adversario A simplemente extiende x concatenando de la siguiente forma: $A(x) = x||x_1||0^n$. De esto vemos que x es prefijo de $A(x)$, y obviamente $A(x)$ es estrictamente mas grande.

Entonces la estrategia del adversario sería:

- Envía A al verificador.
- Al recibir $h_n(x)$ hacer lo siguiente: como el adversario sabe que el largo de x = n, y por tanto sabe que $Pad(x) = x||x_1$, el adversario puede calcular $h_n(A(x)) = h_n(x||x_1||0^n)$ asumiendo que que $h_n(x)$ es un estado intermedio de la iteración de compresión, es decir, podemos usar $h_n(x)$ como un paso H_0 donde sabemos que $h_n(x) = H_0$ y por la misma definición: $H_1 = F(h_n(x), M_1)$.
  Notemos que en este nuevo caso, el estado h_n acaba de resolver el ultimo bloque de $x||x_1$, y por tanto H_1 estaría comenzando a resolver los bloques (nombrados desde M_1 para adelante) que pertenecen a $Pad(x||x_1||0^n)$ (particularmente partiendo desde 0^n) el cual es calculable pues tenemos los tamaños y por lo tanto la función puede seguir.

- Acabamos de demostrar entonces que al seguir con la función hash, podemos terminar de calcular el nuevo $h_n(A(x)) = h_n(x||x_1||0^n)$ solo a partir de $h_n(x)$ y el tamaño de x.

Expliquemos ahora la probabilidad de éxito de este algoritmo:

- Caso b = 0: El verificador calcula $y = h_n(A(x))$. Sabemos que el adversario siempre puede resolver $h_n(A(x)) = h_n(x||x_1||0^n)$ por tanto la probabilidad de éxito es 1.
- Caso b = 1: El verificador calcula $y =$ **aleatorio en** $\{0,1\}^n$. El adversario solo perdería si la permutación "le achunta" sin querer a lo esperado, es decir: $1/2^n$. Por tanto ganaría con probabilidad $1 - 1/2^n$.

Esto ultimo nos deja con la siguiente conclusión:

$Pr[exito] = (1/2) + (1/2) \cdot (1 - 1/2^n)$ Esto es claramente no despreciable, es mas, es mayor que $1/2$ y por tanto, demostramos que existe adversario en tiempo polinomial que puede ganar el juego PreImageModification(1^n), entonces esto no es seguro frente a modificaciones de pre-imagen.