Universidad Rey Juan Carlos

Grado en I. de la Ciberseguridad

Criptografía

Resumen. El ROM. La construcción de Bellare y Rogaway.

---

1. El Modelo del Oráculo Aleatorio (ROM)

Debido a la dificultad de hacer demostraciones de seguridad formales, a finales del siglo pasado empezaron a utilizarse modelos *idealizados* de demostración en los que se toma como hipótesis la existencia de una herramienta *perfecta, ideal,* que se usa en la construcción de un esquema criptográfico, para luego hacer la demostración matemática de seguridad de dicho esquema de modo más simple gracias a esta suposición.

Una de las idealizaciones más extendidas es el llamado *modelo del oráculo aleatorio*. Este modelo, propuesto por Mihir Bellare y Philip Rogaway hacia 1993, toma como hipótesis el poder usar una función hash H ideal, imbatible, que cumple sobradamente las propiedades (CR, TCR y PR) vistas en clase porque suponemos que *las imágenes ($H(x)$) se construyen independientemente de las entradas correspondientes (x)*. Así, no hay patrones que el adversario pueda aprovechar.

La idea es como sigue, decimos que se asume que una función hash H (con salida de n bits –n será típicamente 128, 256...—) es un *oráculo aleatorio*, cuando se implementa a través de un algoritmo con estas características:

1. H recibe entradas de cualquier tamaño y da como salida secuencias de n bits. Al inicializarse, H crea una tabla con dos columnas (`entrada / salida`).
2. Cuando "llamamos" al algoritmo de evaluación de H con un valor x, éste busca dicho valor en la columna de `entrada` de la tabla;
   i. Si x no está ahí, escribe un nuevo registro en la tabla añadiendo x en `entrada` y un valor r en `salida` que elige uniformemente al azar entre todas las cadenas posibles de n bits.[1] En esta elección, x no juega ningún papel (r se elige al azar, ¡¡sin hacer nada con los bits de x!!)

[1] que son 2^n, como ya sabéis.

ii. Si x está en la tabla, el algoritmo devuelve el valor r que está escrito como `salida` en la segunda columna de la tabla, junto al valor x.

Cuando se usa el ROM, estamos suponiendo siempre que el oráculo (la función H) es accesible para todo el mundo. Es decir, igual que ocurre con las funciones hash involucradas en las construcciones criptográficas, no hay clave asociada y cualquiera (usuario legítimo o adversario) puede evaluar la función H en entradas x de su elección. Además, por supuesto, H es *determinista*; una vez escrita en la tabla una entrada x la salida asociada será el valor r que se fijó en la primera llamada. Recordad además que r se elige al azar sin usar x para nada, ahí es dónde está la idelización, pues toda función hash "real" calcula la salida a partir de los bits de entrada. Si queréis ver un ejemplo sencillo de construcción real de un hash, podéis lo tenéis en este vídeo sobre la función hash SHA-1.

`entrada`	`salida`
x_1	r_1
x_2	r_2
.	.
.	.
.	.

Tabla del oráculo aleatorio H

2. El cifrado genérico de Bellare and Rogaway

Hemos visto en las transparencias de clase la construcción RSA-OAEP, que es un ejemplo paradigmático de uso de funciones hash para conseguir mejorar esquemas que parten de una buena idea (como el RSA de libro de texto) pero son inseguros. La construcción OAEP es un caso particular de aplicación de una idea genérica debida a Bellare y Rogaway. Explicamos en estos apuntes una versión (¡simplificada!) de esta idea, que está detrás de muchas construcciones seguras para cifrado de mensajes.

Para cada parámetro de seguridad fijado n, vamos a considerar definida una función f de una vía, con dominio e imagen en un conjunto finito X (que nos servirá como conjunto de mensajes en claro). Por comodidad, pensemos que $X \subseteq \{0,1\}^*$, i.e., que es conjunto de cadenas de bits. También, supondremos que f es biyectiva (vamos, que es una permutación que "revuelve" los elementos de X).

Ahora necesitamos dos funciones especiales, que en el mundo real serán funciones hash y en el "ideal" (a la hora de hacer demostraciones), se tratarán como oráculos aleatorios. Una primera función, que llamaremos $\mathcal{G}$, transforma elementos de X en cadenas de n bits. La otra, $\mathcal{H}$ resumirá cadenas de bits arbitrarias en cadenas de k bits (donde k es polinomial en n, es decir, su tamaño está muy relacionado con n y no se *desmadra* demasiado). Tenemos por tanto

$$\mathcal{G} : X \longmapsto \{0,1\}^n \text{ y } \mathcal{H} : \{0,1\}^* \longmapsto \{0,1\}^k.$$

Recordad que estas funciones son públicas y pueden ser evaluadas sin problema por el adversario. Describimos un esquema de cifrado $\mathsf{BR} = (\mathcal{K}, \mathcal{E}, \mathcal{D})$ de la siguiente forma:

- $\mathcal{K}$, *generación de claves:* Con un parámetro de seguridad $n \in N$ como entrada, especifica el conjunto X y la implementación de la función f y su inversa f^{-1}. La clave pública es la información necesaria para evaluar f y la secreta la que se requiere para evaluar f^{-1}. Escribimos por tanto $(p_k, s_k) = (f, f^{-1})$
- $\mathcal{E}$, *cifrado:* Dado $m \in X$, elige un valor uniformemente al azar $r \in X$ y construye

  $$a := f(r), \quad b := m \oplus \mathcal{G}(r) \text{ y } c := \mathcal{H}(m, r).$$

  El texto cifrado de salida es por tanto la cadena de bits $a \parallel b \parallel c$.
- $\mathcal{D}$ *descifrado:* usando la clave secreta, se calcula r como $f^{-1}(a)$ y eso permite obtener m como $b \oplus \mathcal{G}(r)$. Se comprueba que el valor recibido c coincide con $\mathcal{H}(m, r)$ y si todo va bien, damos el texto m como salida. En otro caso, la salida es $\perp$.

Algunas observaciones:

- A menudo (como en el ejercicio que os propongo en la hoja *magistral*) se escribe un producto en lugar del $\oplus$ al construir el cifrado; todo depende de si trabajamos con números en un grupo o cadenas de bits, pero a todas luces los enfoques son equivalentes (sólo hay que ser formal; si escribes en bits, no puedes multiplicar cosas!!)
- Si cogemos, por ejemplo, una función f definida a partir de RSA, la idea sería que, fijando N, e, d f tome una cadena de bits que representa a un mensaje $m \in \mathbb{Z}_N$ y devuelva la cadena binaria correspondiente a m^e (mód N). De manea similar, f^{-1} calcula la cadena de bits que representa a un valor de entrada c elevado al exponente

d , de nuevo módulo N. Escribe, como ejercicio, como funcionarían los algoritmos $\mathcal{E}$ y $\mathcal{D}$ para ver si has entendido los procesos bien.

- Cuando un esquema criptográfico está definido en el ROM, sabemos que la seguridad de la (o las) funciones hashes que aparecen es *crucial*. Si se descubre alguna vulnerabilidad en las mismas, por pequeña que sea, hemos de ser conscientes de que la demostración de seguridad pierde su significado y es urgente modificar la implementación usando otras funciones más confiables.