

Apuntes de Criptografía

Fernando Virdia

mayo 2025

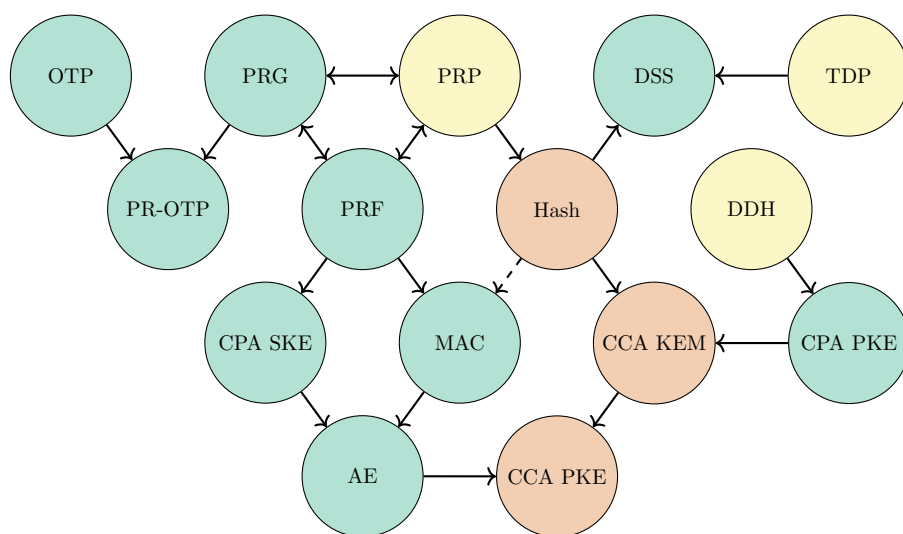


Figure 1: Mapa de las primitivas criptográficas que vamos a ver. En amarillo, primitivas sujetas a postulado. En verde, primitivas que reciben una demostración matemática de seguridad. En naranja, primitivas para las cuales no damos una demostración formal.

Clase 0: introducción

Fernando Virdia, versión: 0.1, mayo 2025

Esta clase se encuentra en forma de diapositivas en <https://apuntes.indcpa.com>.

Clase 1: preliminares matemáticos I

Fernando Virdia, versión: 0.1, mayo 2025

1 Preliminares

1.1 Números y conjuntos

Usamos $\mathbb{Z}_{>0}$ para representar el conjunto de Números enteros positivos $\{1, 2, 3, \dots\}$. Dado $n \in \mathbb{Z}_{>0}$, denotamos con $[n]$ el conjunto $\{1, 2, \dots, n\}$.

Usamos $\emptyset$ para representar el conjunto vacío, $\emptyset = \{ \}$. Usamos $|S|$ para representar la cardinalidad de S .

Dado un valor lógico V definimos $\llbracket V \rrbracket := 1$ si V es verdadero, y $\llbracket V \rrbracket := 0$ si V es falso.

Dados dos bits $b, c \in \{0, 1\}$, escribimos $b \oplus c = b \text{ XOR } c = \llbracket b \neq c \rrbracket = \begin{cases} 0 & \text{si } b = c, \\ 1 & \text{si } b \neq c. \end{cases}$

Usamos $\{0, 1\}^\ell$ para representar el conjunto de cadenas de ℓ bits. Si $\ell = *$, las cadenas pueden ser arbitrariamente largas. Dada una cadena de bits c , denotamos $|c|$ el número de bits en c .

Si $b = b_1 b_2 \dots b_n$ y $c = c_1 c_2 \dots c_n$ son dos cadenas de n bits, escribimos $b \oplus c$ para representar la cadena $d = d_1 d_2 \dots d_n$ donde $d_i = b_i \oplus c_i$.

Lemma 1 (Desigualdad triangular). *Dados $x, y \in \mathbb{R}$, $|x + y| \leq |x| + |y|$.*

1.2 Probabilidad

Dado un conjunto finito S , usamos $U(S)$ para representar la distribución de probabilidad uniforme sobre S . Una variable aleatoria X distribuida según $U(S)$, $X \sim U(S)$, es tal que

$$\Pr[X = x] = \frac{1}{|S|} \quad \forall x \in S.$$

Dado un segundo conjunto S' , y una función $f: S \rightarrow S'$, podemos definir una segunda variable aleatoria $Y = f(X)$. La distribución de Y no es necesariamente la misma de X . A veces escribimos

$$\Pr_{X \sim U(S)}[Y = y]$$

para indicar $\Pr[Y = y] = \Pr[f(X) = y]$ cuando $X \sim U(S)$.

Dadas dos variables aleatorias A y B , la *probabilidad condicional* de $A = a$ dado $B = b$ puede definirse de varias maneras. En este curso consideramos $\Pr[A = a \mid B = b]$ como la probabilidad que $A = a$ asumido que ya se sabe que $B = b$ (por ejemplo, A podría ser el resultado de un experimento, donde B determina parte de la aleatoriedad del mismo). Se mantiene que si $\Pr[B = b] > 0$,

$$\Pr[A = a \mid B = b] = \frac{\Pr[A = a \wedge B = b]}{\Pr[B = b]}.$$

Dados eventos $A = a$ y $B = b$:

- $\Pr[A \neq a] = 1 - \Pr[A = a]$,
- $\Pr[A = a \mid B = b] = 1 - \Pr[A \neq a \mid B = b]$.¹

Teorema 1 (Teorema de la probabilidad total, informal). *Dada una variable aleatoria B con valores en un conjunto finito S , podemos descomponer el evento $A = a$ como*

$$\begin{aligned} \Pr[A = a] &= \sum_{b \in S} \Pr[A = a \wedge B = b] \\ &= \sum_{b \in S} \Pr[A = a \mid B = b] \cdot \Pr[B = b]. \end{aligned}$$

Lemma 2 (Union bound, cota de la unión). *Dados eventos $A_1, \dots, A_n$,*

$$\Pr[A_1 \vee A_2] \leq \Pr[A_1] + \Pr[A_2], \text{ e iterando, } \Pr\left[\bigvee_{i=1}^n A_i\right] \leq \sum_{i=1}^n \Pr[A_i].$$

¹Demostración: $1 = \frac{\Pr[B=b \wedge A=a] + \Pr[B=b \wedge A \neq a]}{\Pr[B=b]} = \Pr[A = a \mid B = b] + \Pr[A \neq a \mid B = b]$.

1.3 Aleatoriedad

Postulado 1. Dado un conjunto finito R , podemos tomar muestras r_i de la distribución uniforme sobre R , $r_i \sim U(R)$.

Comentario 1. Generalmente consideramos $R = \{0, 1\}^n$ con $n \in \mathbb{Z}_n$.

Comentario 2. Generalmente, llamamos una tal muestra $r_i \sim U(R)$, monedas (“coins”). A veces escribimos $r_i \stackrel{\$}{\leftarrow} R$.

Definición 1 (Algoritmo aleatorio). Sea $\mathcal{A}$ un algoritmo con conjunto de inputs I y conjunto de outputs O . Decimos que $\mathcal{A}$ es aleatorio si dado $x \in I$, $\mathcal{A}(x)$ es una variable aleatoria con valor $\mathcal{A}(x) \in O$.

En criptografía, definimos la variante “de-randomizada” $\bar{\mathcal{A}}$ de $\mathcal{A}$ como la función $\bar{\mathcal{A}}: I \times R \rightarrow O$ tal que

$$\forall y \in O, \quad \Pr[\mathcal{A}(x) = y] = \Pr_{r \sim U(R)}[\bar{\mathcal{A}}(x, r) = y].$$

Generalmente abusamos notación y escribimos $\mathcal{A}(x; r)$ en lugar de $\bar{\mathcal{A}}(x, r)$.

En criptografía trabajamos con “adversarios” que interceptan comunicaciones e intentan vulnerarlas de varias maneras. Estos adversarios son modelados como algoritmos aleatorios, que corren en tiempo $\leq t$. En criptografía a menudo argumentamos que dos variables aleatorias tienen una distribución muy “parecida”. A través del siguiente teorema, eso nos permite de prever los logros de un adversario.

Teorema 2 (Informal, “data processing inequality”). Dadas dos variables aleatorias X, Y con distribución “parecidas”, y dada una función o algoritmo aleatorio $\mathcal{A}$, las variables aleatorias $\mathcal{A}(X)$, $\mathcal{A}(Y)$ “se parecen” al menos cuanto X e Y .

Proof. (boceto) Sea $\Delta(X, Y)$ la “distancia estadística” (*statistical distance* o *total variation distance*) entre dos variables aleatorias X e Y con soporte S finito:

$$\Delta(X, Y) := \frac{1}{2} \sum_{\alpha \in S} |\Pr[X = \alpha] - \Pr[Y = \alpha]|$$

Dada una función $f: S \rightarrow S'$, $f(X)$ y $f(Y)$ son variables aleatorias con soporte S' y distancia estadística

$$\Delta(f(X), f(Y)) \leq \Delta(X, Y) \quad (\text{sumando sobre } \alpha \in S').$$

□

Definición 2 (función aleatoria). Sean $\mathcal{D}$ y $\mathcal{R}$ conjuntos finitos. Decimos que $f: \mathcal{D} \rightarrow \mathcal{R}$ es una función aleatoria, si cada salida $f(x)$ fue muestreada aleatoriamente en $\mathcal{R}$.

Generalmente, f puede ser realizada a través de la siguiente máquina de estados.

$f(x)$
1: if $L = \perp$:
2: $L \leftarrow []$ // una tabla hash
3: if $x \notin L$:
4: $L[x] := y \stackrel{\$}{\leftarrow} \mathcal{R}$
5: return $L[x]$

Si $\mathcal{D}$ es un conjunto finito, esta definición coincide con la de una función muestreada uniformemente del conjunto (finito) $\text{Func}(\mathcal{D}, \mathcal{R}) := \{f \mid f: \mathcal{D} \rightarrow \mathcal{R}\}$ de funciones de $\mathcal{D}$ a $\mathcal{R}$.

Comentario 3 (Algoritmos “eficientes”). Durante este curso vamos a ver varios algoritmos usados para cifrar y autenticar datos. Aunque no los tratamos de manera asintótica, vamos a requerir implícitamente que estos algoritmos sean eficientes. Cifrado y autenticado generalmente tardan tiempo proporcional a $|\text{input}|$. Para un input de pocos bits, estas operaciones suelen tardar pocos milisegundos o menos (en Linux, corran `openssl speed` para ver un ejemplo).

Clase 2: cifrado simétrico, secreto perfecto

Fernando Virdia, versión: 0.1, mayo 2025

2 One-Time Pads y secreto perfecto

Introducimos la noción de cifrado simétrico.

Definición 3 (SKES, cifrado simétrico). Sean $\mathcal{K}$, $\mathcal{M}$, $\mathcal{C}$ tres conjuntos. Decimos que $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ forman un cifrado simétrico (SKES, “secret-key encryption scheme”) con espacio de llaves $\mathcal{K}$, espacio de mensajes $\mathcal{M}$, y espacio de cifrados $\mathcal{C}$, si

- Gen es un algoritmo aleatorio, $\text{Gen}: \emptyset \rightarrow \mathcal{K}$;
- Enc es un algoritmo (posiblemente aleatorio), $\text{Enc}: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{C}$;
- Dec es un algoritmo determinista $\text{Dec}: \mathcal{C} \rightarrow \mathcal{M}$;

tales que

$$\Pr_{\substack{k \sim \text{Gen}(), \\ \$ \text{ en Enc}}} [\text{Dec}(k, \text{Enc}(k, m; \$)) = m] = 1,$$

donde Gen, Enc y Dec son “eficientes”.

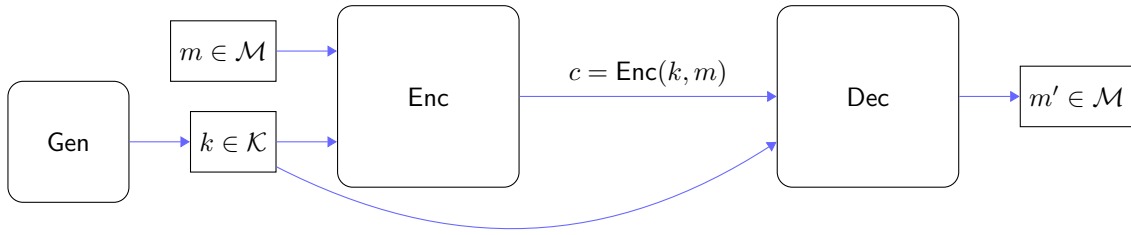


Figure 2: Uso de un cifrado simétrico. Consideramos el cifrado c ser expuesto al adversario.

Comentario 4. Generalmente consideramos $\mathcal{K}, \mathcal{M}, \mathcal{C} \subset \{0, 1\}^*$ en la criptografía simétrica.

Comentario 5. Generalmente consideramos $\mathcal{K}$ finito, y $\text{Gen}()$ consiste en muestrear la distribución uniforme $k \xleftarrow{\$} U(\mathcal{K})$.

Dada la sintaxis de un SKES, queremos encontrar definiciones que capturen la “seguridad” del mensaje cifrado $c = \text{Enc}(k, m)$.

PREGUNTA: Alguna idea de como definir la seguridad de c ?

1. Podríamos pedir que dado c , un adversario no aprenda k . Y si aprende m ?
2. Podríamos pedir que dado c , un adversario no aprenda m . Y si aprende la primera mitad de m ?
3. Podríamos pedir que dado c , no aprenda ninguna información sobre m . Y si alguna información es publica, por ejemplo que todo $m \in \mathcal{M}$ está escrito en castellano?

Definición 4 (Secreto perfecto, Shannon). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un SKES. Decimos que Π ofrece secreto perfecto si dada la variable aleatoria $k \sim \text{Gen}()$,

$$\forall m_0, m_1 \in \mathcal{M}, c \in \mathcal{C}, \quad \Pr_{k \sim \text{Gen}()} [\text{Enc}(k, m_0) = c] = \Pr[\text{Enc}(k, m_1) = c].$$

Comentario 6. Un SKES que otorga secreto perfecto leakea que $m \in \mathcal{M}$. Por ejemplo, si $\mathcal{M} = \{0, 1\}^\ell$, leakea que $|m| = \ell$.

Ejemplo 1 (Cifrario de Cesar). Sea $\mathcal{K} = \{0, 1, \dots, 9\}$, $\mathcal{M} = \{0, 1, \dots, 9\}^*$. Supongamos $m = m_0, m_1, m_2 \in \mathcal{M}$. Para cifrar m con llave k , sumamos $m_i + k \bmod 10$ para $i = 1, 2, 3$. Por ejemplo, $m = 123, k = 3 \implies c = 456$.

El cifrado de Cesar otorga secreto perfecto? No! Por ejemplo, dado $m_0 = 123$, $m_1 = 555$, $c = 567$, podemos observar que

$$\begin{aligned} \Pr[c = \text{Enc}(k, m_0)] &> 0 & (\text{existe una llave } k \text{ valida}) \\ \Pr[c = \text{Enc}(k, m_1)] &= 0. \end{aligned}$$

Una definición equivalente de secreto perfecto requiere

$$\forall m \in \mathcal{M}, c \in \mathcal{C}, \quad \Pr[M = m \mid C = c] = \Pr[M = m].$$

En nuestro ejemplo, ver $C = 567$ implica la imposibilidad de que $M = 555$, por lo cual aprendemos algo sobre M viendo C , $\Pr[M = m \mid C = c] \neq \Pr[M = m]$.

Definición 5 (One-Time Pad). Sean $n \in \mathbb{Z}_{>0}$, $\mathcal{K} = \mathcal{M} = \mathcal{C} = \{0, 1\}^n$. $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ es un one-time pad (OTP), si

$\text{Gen}()$	$\text{Enc}(k, m)$	$\text{Dec}(k, c)$
1: $k \xleftarrow{\$} \mathcal{K}$	1: $c \leftarrow m \oplus k$	1: $m' \leftarrow c \oplus k$
2: return k	2: return c	2: return m'

Teorema 3. El OTP ofrece secreto perfecto.

Proof. Sean $n \in \mathbb{Z}_{>0}$, $m_0, m_1 \in \{0, 1\}^n$, $c \in \{0, 1\}^n$.

$$\begin{aligned} \Pr_{k \sim U(\mathcal{K})} [\text{Enc}(k, m_0) = c] &= \Pr[k \oplus m_0 = c] & (\text{def. de Enc}) \\ &= \Pr[k = c \oplus m_0] \\ &= 2^{-n} & (\text{dado } k \sim U(\mathcal{K})) \\ &= \Pr[k = c \oplus m_1] \\ &= \Pr[k \oplus m_1 = c] \\ &= \Pr[\text{Enc}(k, m_1) = c]. \end{aligned}$$

□

Teorema 4 (Teorema de Shannon). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un SKES con secreto perfecto. Entonces $|\mathcal{K}| \geq |\mathcal{M}|$.

Proof. Por contradicción, supongamos $|\mathcal{K}| < |\mathcal{M}|$. Idea: vamos a construir m_0, m_1, c , tales que $\Pr[\text{Enc}(k, m_0) = c] > 0$ y $\Pr[\text{Enc}(k, m_1) = c] = 0$.

Fijemos $k_0 \leftarrow \text{Gen}()$, $m_0 \in \mathcal{M}$, $c = \text{Enc}(k_0, m_0)$.

$$\implies \Pr_{k \sim \text{Gen}()} [\text{Enc}(k, m_0) = c] \geq \Pr_{k \sim \text{Gen}()} [k = k_0] > 0.$$

Ahora definamos el conjunto $S := \{\text{Dec}(k, c) \mid \forall k \in \mathcal{K}\} \subset \mathcal{M}$. Notamos que $|S| \leq |\mathcal{K}| < |\mathcal{M}|$. Dado que $S \subset \mathcal{M}$ y $|S| < |\mathcal{M}|$, sabemos que $\mathcal{M} \setminus S = \{m \mid m \in \mathcal{M} \text{ y } m \notin S\} \neq \emptyset$. Elijamos $m_1 \in \mathcal{M} \setminus S$.

Supongamos que $\exists k \in \mathcal{K}$ tal que $\text{Enc}(k, m_1) = c$. Si así fuera, dado que Π es un SKES, sabemos que $\text{Dec}(k, c) = \text{Dec}(k, \text{Enc}(k, m_1)) = m_1$, lo que implicaría que $m_1 \in S$. ✗ Por lo tanto $\nexists k \in \mathcal{K}$ tal que $\text{Enc}(k, m_1) = c$

$$\implies \Pr_{k \sim \text{Gen}()} [\text{Enc}(k, m_1) = c] = 0.$$

Esto contradice la suposición que Π brinde secreto perfecto. ✗ Por lo tanto, $|\mathcal{K}| \geq |\mathcal{M}|$. □

PREGUNTA: Por que “one-time”? Alguna idea? Vamos a ver la razón próximamente.

Comentario 7. El teorema de Shannon nos dice que para obtener secreto perfecto, se requiere una llave secreta k grande al menos tanto cuanto el mensaje que se quiere ocultar. Esto nos pondría frente a la situación paradójica por la cual comunicar de manera segura un mensaje requiere comunicar de manera segura una clave no mas breve.

La única ventaja del OTP es la a-sincronía. La clave puede ser comunicada de antemano, y el mensaje determinado solo luego. Esto probablemente aun se usa en algunas embajadas.²

En la próxima clase vamos a ver nuevos postulados que nos van a permitir superar las limitaciones del teorema de Shannon.

Ataque 1 (OTP). Supongamos $\mathcal{A}$ ve $c \in \mathcal{C}$. Que aprende sobre m ? Dado cualquier $m \in \mathcal{M}$, si $k \leftarrow m \oplus c$, $\text{Dec}(k, c) = k \oplus c = (m \oplus c) \oplus c = m$. $\mathcal{A}$ solamente puede aprender que $m \in \mathcal{M}$ ($\Rightarrow$ aprende $|m|$), y que comunicación cifrada ocurrió. Usando padding extra se podría “obfuscar” $|m|$, y comunicando a intervalos regulares información “nula”, podría “obfuscarse” que la comunicación ocurrida fue significativa. Notamos que estos ataques están “fuera” de la definición de secreto perfecto, y no la contradicen.

²<https://archive.is/mmbtX#selection-1025.0-1025.238>

Clase 3: seguridad computacional y pseudoaleatoriedad

Fernando Virdia, versión: 0.1, mayo 2025

Ahora vamos a ver relajaciones de la noción de *secreto perfecto*, y vamos a usar un nuevo postulado para construir algoritmos de cifrado mas eficientes del OTP.

Es común en criptografía *relajar* las definiciones de seguridad, en cambio de eficiencia. Vemos un ejemplo.

Definición 6 (One-time ε -secrecy). Decimos que un SKES tiene one-time ε -secrecy (*OT ε -secrecy*) si dada la variable aleatoria $k \sim \text{Gen}()$,

$$\forall m_0, m_1 \in \mathcal{M}, c \in \mathcal{C}, \quad |\Pr[\text{Enc}(k, m_0) = c] - \Pr[\text{Enc}(k, m_1) = c]| \leq \varepsilon.$$

Comentario 8. Un SKES con OT ε -secrecy con $\varepsilon = 0$ otorga secreto perfecto.

EJERCICIO: [BS23, Exercise 2.5]: OT ε -secrecy $\implies |\mathcal{K}| \geq (1 - \varepsilon)|\mathcal{M}|$

Comentario 9. Lo mas cercano ε a 0, lo mas cercano OT ε -secrecy al secreto perfecto, y lo mas cercano lo que un adversario $\mathcal{A}$ puede hacer dado un cifrado salido de un SKES con OT ε -secrecy a lo que puede hacer con un cifrado salido de un cifrado con secreto perfecto.

En la practica, tratamos que $\varepsilon \in [2^{-256}, 2^{-128}]$. Si $\varepsilon < 2^{-\lambda}$, decimos que un SKES ofrece λ -bits de seguridad estadística.

Comentario 10. Por que “one-time”? Supongamos que Alicia quiere comunicar con Bob, y comparte una clave de OTP k muestreada $k \sim \text{Gen}()$. Alicia y Bob deciden de cifrar dos mensajes usando la misma clave:

$$\begin{aligned} c_1 &= k \oplus m_1 \\ c_2 &= k \oplus m_2. \end{aligned}$$

Si un adversario $\mathcal{A}$ observa los cifrados c_1 y c_2 en transito, que puede aprender sobre m_1 y m_2 ?

$$\begin{aligned} c_1 \oplus c_2 &= (k \oplus m_1) \oplus (k \oplus m_2) \\ &= m_1 \oplus m_2. \end{aligned}$$

Esto es un problema! Por ejemplo, si m_1 es un mensaje predecible, digamos $\Pr[m_1 = \overline{m}_1] \approx 1$, $\mathcal{A}$ puede adivinar, y determinar que $m_2 = c_1 \oplus c_2 \oplus \overline{m}_1$ con probabilidad ≈ 1 .

El OTP otorga seguridad si la llave (“pad”) k se usa una sola vez (“one-time”). No garantiza nada si la llave se la usa mas de una vez.

Comentario 11. OT ε -secrecy solo garantiza confidencialidad. No otorga garantías de integridad. Si Alicia usa un OTP para comunicarle a su banco que quiere pagar $m = \$4 = \100_2 a un negociante, y este intercepta el cifrado

$$\begin{aligned} c &= c_0 c_1 \dots c_9 = k_0 \ k_1 \dots k_6 \ k_7 \ k_8 \ k_9 \\ &\oplus \ 0 \ 0 \dots 0 \ 1 \ 0 \ 0, \end{aligned}$$

lo intercepta y transmite $c' = c \oplus (10 \dots 0)$ al banco, este descifraría

$$m' = k \oplus c' = (\$1000000100)_2 = \$2^9 + 2^2 = \$516,$$

costandole a Alicia mucho mas.

3 PRGs, PRFs, PRPs

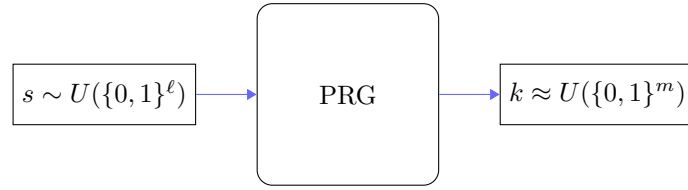
Ahora vamos a definir 3 familias de objetos criptográficos que vamos a utilizar para construir mejores cifrados.

Definición 7 (PRG). Sea $G: \{0,1\}^\ell \rightarrow \{0,1\}^m$ una función, implementable como un algoritmo determinista eficiente. Generalmente $\ell < m$. Dado un adversario aleatorio $\mathcal{A}$ y un bit $b \in \{0,1\}$, definimos el siguiente experimento $\text{Exp}^{\text{PRG}}(\mathcal{A}, b)$

$\text{Exp}^{\text{PRG}}(\mathcal{A}, 0)$	$\text{Exp}^{\text{PRG}}(\mathcal{A}, 1)$
1: $s \xleftarrow{\$} \{0,1\}^\ell$	1: $y \xleftarrow{\$} \{0,1\}^m$
2: $b' \leftarrow \mathcal{A}(G(s))$	2: $b' \leftarrow \mathcal{A}(y)$
3: return b'	3: return b'

Decimos que G es un generador (ε, t) -pseudoaleatorio (PRG, “pseudorandom generator”), si para todo adversario $\mathcal{A}$ que corre en tiempo $\leq t$, la ventaja de $\mathcal{A}$ es

$$\text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$



Comentario 12. Este tipo de definición dice que $\mathcal{A}$ no logra distinguir el experimento “ $b = 0$ ” del experimento “ $b = 1$ ”, si $\varepsilon \approx 0$. Esta intuición puede ser formalizada usando la equivalente definición de “bit-guessing”: bajo las condiciones de Definition 7, G es un (ε, t) -PRG si dado el experimento $\overline{\text{Exp}}^{\text{PRG}}$,

$\overline{\text{Exp}}^{\text{PRG}}(\mathcal{A})$
1: $b \xleftarrow{\$} \{0,1\}$
2: $b' \leftarrow \text{Exp}^{\text{PRG}}(\mathcal{A}, b)$
3: return $\llbracket b = b' \rrbracket$

$$\text{Adv}(\overline{\text{Exp}}^{\text{PRG}}, \mathcal{A}) := \left| \Pr[\overline{\text{Exp}}^{\text{PRG}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| \leq \frac{\varepsilon}{2}.$$

Lemma 3. Bajo las condiciones de Definition 7,

$$\text{Adv}(\overline{\text{Exp}}^{\text{PRG}}, \mathcal{A}) = \frac{1}{2} \text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{A}).$$

Proof. Usando el teorema de la probabilidad total,

$$\begin{aligned} \Pr[\overline{\text{Exp}}^{\text{PRG}}(\mathcal{A}) \Rightarrow 1] &= \Pr[b' = b] \\ &= \Pr[b' = b \mid b = 0] \Pr[b = 0] + \Pr[b' = b \mid b = 1] \Pr[b = 1]. \end{aligned}$$

Ahora notamos que por definición de Exp^{PRG} ,

$$\Pr[b' = b \mid b = 0] = \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 0] = 1 - \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1],$$

$$\text{y } \Pr[b' = b \mid b = 1] = \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1].$$

De consecuencia,

$$\begin{aligned} \text{Adv}(\overline{\text{Exp}}^{\text{PRG}}, \mathcal{A}) &= \left| \Pr[b' = b] - \frac{1}{2} \right| \\ &= \left| \frac{1}{2} (\Pr[b' = b \mid b = 0] + \Pr[b' = b \mid b = 1]) - \frac{1}{2} \right| \\ &= \left| \frac{1}{2} (1 - \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1] + \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1]) - \frac{1}{2} \right| \end{aligned}$$

$$\begin{aligned}
&= \frac{1}{2} \left| \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1] \right| \\
&= \frac{1}{2} \text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{A}).
\end{aligned}$$

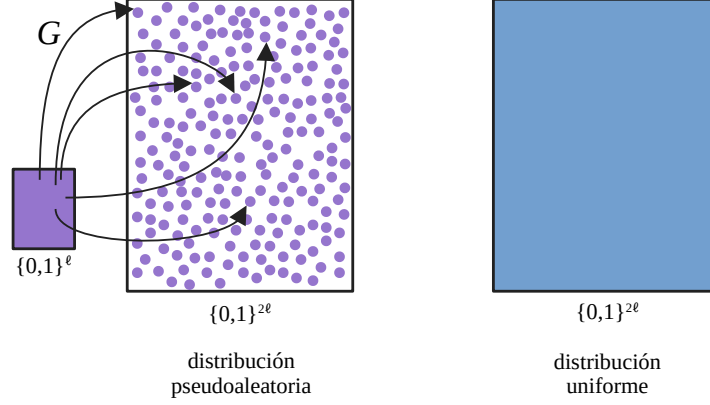
□

Comentario 13. *Este misma equivalencia vale para la mayor parte de los experimentos que consideramos en criptografía.*

Clase 4: Bit security y OTP computacional

Fernando Virdia, versión: 0.1, mayo 2025

Comentario 14. *Que quieren decir exactamente (ε, t) -pseudoaleatorio?*



El output de $G(\cdot) \in \{0, 1\}^m$ es muy lejano de ser uniforme en $\{0, 1\}^m$, si uno puede ver “de una” la distribución.

La definición de (ε, t) -PRG no dice que $G(s \xleftarrow{\$} \{0, 1\}^\ell) \sim U(\{0, 1\}^m)$ si no que un adversario $\mathcal{A}$ que corre en tiempo $\leq t$ no puede distinguirlas!

Definición 8 ((ε, t) -atacante). Sean $\varepsilon \in [0, 1]$, $t \geq 1$. Dado una experimento de seguridad Exp , un (ε, t) -Exp-atacante contra Exp es un algoritmo (posiblemente aleatorio) $\mathcal{A}$ que corre en tiempo $\leq t$ con ventaja $\text{Adv}(\text{Exp}, \mathcal{A}) \geq \varepsilon$.

Definición 9 ((ε, t) -seguridad). Sea Π una primitiva criptografica, y sea “X” una noción de seguridad con experimento Exp^X . Decimos que Π otorga (ε, t) -X-seguridad si no existe ningún (ε, t) - Exp^X -atacante contra Π .

En otras palabras, cualquier algoritmo $\mathcal{A}$ que juegue Exp^X ,

- o si corre en tiempo $\leq t$ pero obtiene solamente ventaja $< \varepsilon$,
- o corre en tiempo $> t$ para obtener ventaja $\geq \varepsilon$.

Comentario 15. Contra un sistema (ε, t) -seguro pueden existir atacantes “triviales”. Estos sugieren un máximo nivel de seguridad que un tipo de primitiva puede obtener.

- Por ejemplo, si $G: \{0, 1\}^\ell \rightarrow \{0, 1\}^m$ con $m = 2\ell$ es una función aleatoria en $\text{Func}(\{0, 1\}^\ell, \{0, 1\}^m)$, el siguiente atacante corre en tiempo $\approx 2^\ell$ y gana con ventaja $\approx 1 - 2^{-\ell}$:

$\mathcal{A}(y \in \{0, 1\}^m)$
1: $L = \{\}$
2: for $x \in \{0, 1\}^\ell$:
3: $L \leftarrow L \cup \{G(x)\}$
4: if $y \in L$ return 0 ($b' = 0$, “PRG”)
5: else return 1 ($b' = 1$, “random”).

Claramente, $\mathcal{A}$ corre en tiempo $\approx 2^\ell$.

EJERCICIO: Demuestren que $\text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{A}) \approx 1 - 2^{-\ell}$

Su ventaja es $\text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{A}) = \left| \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1] \right|$, con $\Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1] = 0$ (si $b = 0$, entonces $y \in L$ y $\mathcal{A}$ adivina $b' = 0$ siempre correctamente), y con $\Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1] = \Pr[y \notin L]$, que dado $b = 1 \Leftrightarrow y \notin L$ pasa con probabilidad $\approx (1 - 2^{-m})^{2^\ell} \approx 1 - 2^\ell \cdot 2^{-m} = 1 - 2^{-\ell}$.

Esto quiere decir que contra un PRG $G: \{0, 1\}^\ell \rightarrow \{0, 1\}^{2\ell}$ siempre existe un $(\varepsilon \approx 1 - 2^{-\ell}, t \approx 2^\ell)$ -atacante.

- En el otro extremo, se considere el siguiente atacante:

$\mathcal{A}(y \in \{0, 1\}^m)$
1: $z \leftarrow G(00\dots 0)$
2: if $y = z$ return 0 ($b' = 0$, “PRG”)
3: else return 1 ($b' = 1$, “random”).

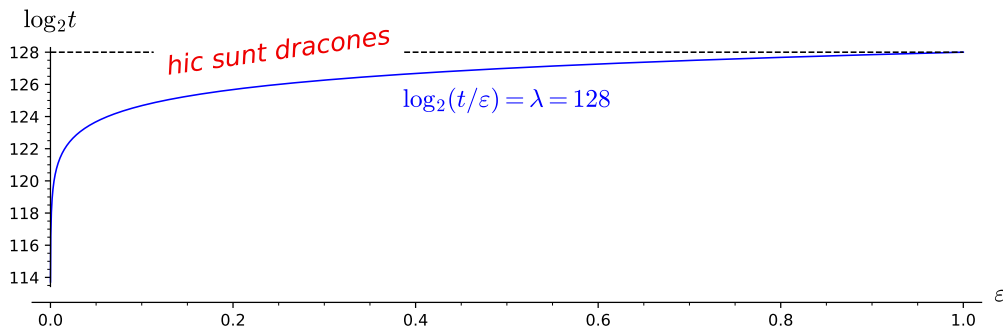
EJERCICIO: Demuestren que $\mathcal{A}$ es un $(\varepsilon \approx 2^{-\ell}, t \approx 1)$ -atacante. (Idea: el espacio $G(\{0, 1\}^\ell)$ es mas chico de $\{0, 1\}^{2\ell}$.)

Aquí $\mathcal{A}$ corre en tiempo $t \approx 1$, pero con pésima ventaja: $\Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1] = \Pr[G(s) \xrightarrow{\$} \{0, 1\}^\ell \neq G(0\dots 0)] \approx 1 - 2^{-\ell}$, mientras $\Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1] = \Pr[y \neq G(0\dots 0)] \approx 1 - 2^{-2\ell}$, que implica $\text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{A}) \approx 2^{-\ell} - 2^{-2\ell} \approx 2^{-\ell}$. Por lo cual existe un $(\varepsilon \approx 2^{-\ell}, t \approx 1)$ -atacante.

Definición 10 (λ -bits de seguridad). Sea Π una primitiva criptografica, y sea “ X ” una nocion de seguridad con experimento Exp^X . Supongamos que existe $\lambda \in \mathbb{Z}_{>0}$ tal que para cualquier $\varepsilon \in [0, 1]$, Π otorga (ε, t) - X -seguridad con $t \geq 2^\lambda \cdot \varepsilon$. Entonces decimos que Π otorga λ bits de X -seguridad.

Comentario 16. Si una primitiva otorga λ -bits de seguridad, esto quiere decir que

- es necesario trabajar un tiempo $\geq 2^\lambda$ para obtener ventaja ≈ 1 ,
- trabajando un tiempo $t \approx 1$ solamente se puede obtener ventaja $\approx 2^{-\lambda}$,
- trabajando un tiempo $t \approx \sqrt{2^\lambda}$ se puede obtener una ventaja $\approx 1/\sqrt{2^\lambda}$.



Regresemos al problema del OTP: para cifrar mensajes $m \in \{0, 1\}^m$ precisamos transmitir de manera segura una clave $k \sim U(\{0, 1\}^m)$.

PREGUNTA: Como podemos usar un PRG?

Idea: Podemos remplazar la clave aleatoria $k \xleftarrow{\$} \{0, 1\}^m$ con una clave pseudoaleatoria $k \leftarrow G(s)$, donde $s \xleftarrow{\$} \{0, 1\}^\ell$ es mas corta que k ($= m$).

Definición 11 (PR-OTP). Sean $\ell, m \in \mathbb{Z}_{>0}$ con $\ell < m$, $\mathcal{K} = \{0, 1\}^\ell$, $\mathcal{M} = \mathcal{C} = \{0, 1\}^m$. Sea $G: \{0, 1\}^\ell \mapsto \{0, 1\}^m$ un PRG. $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ es un one-time pad pseudoaleatorio (PR-OTP, “pseudorandom OTP”), si

$\text{Gen}()$	$\text{Enc}(k, m)$	$\text{Dec}(k, c)$
1: $k \xleftarrow{\$} \mathcal{K}$	1: $c \leftarrow m \oplus G(k)$	1: $m' \leftarrow c \oplus G(k)$
2: return k	2: return c	2: return m'

Comentario 17. El nuevo cifrado no otorga secreto perfecto. En la demostración del OTP, el pasaje clave era

$$\Pr[k = c \oplus m_0] = \Pr[k = c \oplus m_1] \quad \forall m_0, m_1, c$$

Dado $G: \{0, 1\}^5 \rightarrow \{0, 1\}^{10}$, no todo valor en $\{0, 1\}^{10}$ tiene una “pre-imagen” en $\{0, 1\}^5$. Es posible que $\exists m_0, m_1, c$ tales que $\Pr[G(k) = c \oplus m_0] > 0$ y $\Pr[G(k) = c \oplus m_1] = 0$.

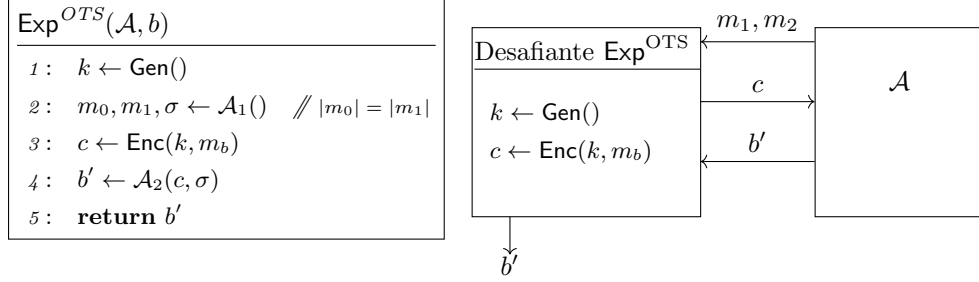
PREGUNTA: Alguien tiene alguna idea de que podemos intentar demostrar?

Idea: Podríamos intentar demostrar OT ε -secrecy?

Como vimos, dándole suficiente tiempo al adversario, todo PRG es inseguro. Las demostraciones en criptografía son del tipo “si $\exists$ un ataque en el esquema $\Rightarrow \exists$ un ataque en el componente”, y solo tienen sentido si creemos que $\nexists$ un ataque en el componente.

Necesitamos una definición de seguridad que considere un adversario con tiempo limitado.

Definición 12 (One-time (ε, t) -secrecy). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un SKES con espacio de mensajes $\mathcal{M} = \{0, 1\}^*$. Dado un adversario $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ y un bit $b \in \{0, 1\}$, definimos el experimento $\text{Exp}^{\text{OTS}}(\mathcal{A}, b)$,



Decimos que Π otorga one-time (ε, t) -secrecy³ si para cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ y produce mensajes de igual de largo $|m_0| = |m_1|$, su ventaja (“advantage”) en distinguir $b = 0$ de 1 es

$$\left| \Pr[\text{Exp}^{\text{OTS}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{OTS}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

Hasta ahora habíamos visto definiciones de seguridad “information-theoretical” (secreto perfecto; “para cualquier $\mathcal{A} \dots \Pr = \Pr$ ”), y “estadística” (ε -secrecy, “para cualquier $\mathcal{A} \dots |\Pr - \Pr| \leq \varepsilon$ ”). Finalmente llegamos a una definición de seguridad “computacional”:

“para cualquier $\mathcal{A}$ en tiempo $\leq t \dots |\Pr - \Pr| \leq \varepsilon$ ”.

Es esta relajación de requisitos que nos permite de obtener la mayor parte de las garantías criptográficas.

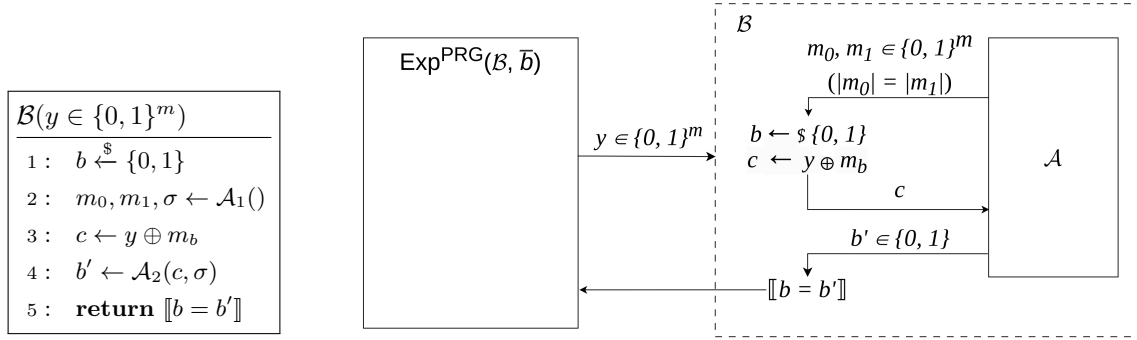
³“EAV-security” en [KL20], “semantic security” en [BS23].

Clase 5: “one-time-pad” pseudoaleatorio

Fernando Virdia, versión: 0.1, mayo 2025

Lemma 4. Sea G un (ε, t) -PRG. Un PR-OTP Π^G que utiliza G , otorga (ε', t') -secrecy con $\varepsilon' = 2\varepsilon$ y $t' \approx t$.

Proof. Supongamos de haber encontrado un (ε', t') -atacante: o sea un adversario $\mathcal{A}$ que corre en tiempo $\leq t'$ y que tiene ventaja $\text{Adv}(\text{Exp}^{\text{OTS}}, \mathcal{A}) > \varepsilon'$ cuando “juega” Exp^{OTS} . A partir de $\mathcal{A}$ construiremos $\mathcal{B}$ que corre en tiempo $t \approx t'$ y que juega Exp^{PRG} .



Calculemos $\text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{B})$:

- Si $\mathcal{B}$ recibe $y \leftarrow G(s)$ en Exp^{PRG} , internamente crea para $\mathcal{A}$ el mismo ambiente de la versión “bit-guessing” de Exp^{OTS} . Por lo tanto,

$$\Pr[\text{Exp}^{\text{PRG}}(\mathcal{B}, 0) \Rightarrow 1] = \Pr[\overline{\text{Exp}_{\Pi^G}^{\text{OTS}}}(\mathcal{A}) \Rightarrow 1].$$

- Si $\mathcal{B}$ recibe $y \leftarrow_{\$} \{0, 1\}^m$, simula nuevamente $\overline{\text{Exp}^{\text{OTS}}}$ para $\mathcal{A}$, pero en este caso es con un cifrado OTP!

$$\Pr[\text{Exp}^{\text{PRG}}(\mathcal{B}, 1) \Rightarrow 1] = \Pr[\overline{\text{Exp}_{\text{OTP}}^{\text{OTS}}}(\mathcal{A}) \Rightarrow 1].$$

Por definicion del OTP, la distribucion de $y \oplus m_0$ y la de $y \oplus m_1$ son identicas, uniformes en $\mathcal{C}$. Abusando la notación, sea $b' \leftarrow \mathcal{A}_2(\sigma, c \leftarrow y \oplus m_b; r)$, donde σ es independiente de b (el bit mostrado internamente por $\mathcal{B}$), así como lo son las monedas $r \sim U(R)$, y repliquemos la demostración de Lemma 3, para obtener

$$\begin{aligned} \forall b \in \{0, 1\}, \quad \Pr[\text{Exp}^{\text{PRG}}(\mathcal{B}, 1) \Rightarrow 1] &= \Pr[b' = b] \\ &= \frac{1}{2} \left(\Pr[b' = b \mid b = 0] + \Pr[b' = b \mid b = 1] \right) \\ &= \frac{1}{2} + \frac{1}{2} \left(\Pr[b' = 1 \mid b = 1] - \Pr[b' = 1 \mid b = 0] \right). \end{aligned}$$

Dado que $y \sim U(\{0, 1\}^m)$,⁴ $y \oplus m_0 \sim y \oplus m_1 \sim U(\{0, 1\}^m)$.⁵ De acuerdo, las variables aleatorias $\mathcal{A}_2(y \oplus m_0)$ y $\mathcal{A}_2(y \oplus m_1)$ tienen la misma distribución.

$$\begin{aligned} \Pr_y[\mathcal{A}_2(y \oplus m_0; R) = b \mid R = r] &= \Pr[(y \oplus m_0) \in \mathcal{A}_2^{-1}(b) \mid R = r] \\ &= \sum_{c \in \mathcal{A}_2^{-1}(b) \mid R=r} \Pr_y[c = y \oplus m_0] \\ &= \sum_{c \in \mathcal{A}_2^{-1}(b) \mid R=r} \Pr_y[c = y \oplus m_1] \\ &= \Pr[(y \oplus m_1) \in \mathcal{A}_2^{-1}(b) \mid R = r] \\ &= \Pr_y[\mathcal{A}_2(y \oplus m_1; R) = b \mid R = r]. \end{aligned}$$

⁴O equivalentemente, como corolario del secreto perfecto del OTP.

⁵ $y \oplus m_0$ y $y \oplus m_1$ no son independientes, pero tienen la misma distribución. $\mathcal{A}_2$ nunca ve ambas variables al mismo tiempo.

Sigue que,

$$\begin{aligned}\Pr[b' = 1 \mid b = 1] &= \Pr[\mathcal{A}_2(y \oplus m_1) = 1] \\ &= \Pr[\mathcal{A}_2(y \oplus m_0) = 1] \\ &= \Pr[b' = 1 \mid b = 0],\end{aligned}$$

y por lo tanto $\Pr[\text{Exp}^{\text{PRG}}(\mathcal{B}, 1) \Rightarrow 1] = \frac{1}{2}$, y

$$\begin{aligned}\varepsilon' &\leq \text{Adv}(\text{Exp}^{\text{OTS}}, \mathcal{A}) = 2 \cdot \text{Adv}(\overline{\text{Exp}^{\text{OTS}}}, \mathcal{A}) \\ &= 2 \cdot \left| \Pr[\overline{\text{Exp}^{\text{OTS}}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| \\ &= 2 \cdot \left| \Pr[\text{Exp}^{\text{PRG}}(\mathcal{B}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRG}}(\mathcal{B}, 1) \Rightarrow 1] \right| \\ &= 2 \cdot \text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{B}) \leq 2\varepsilon\end{aligned}$$

dado que por suposición G es un (ε, t) -PRG. Claramente, $t' \approx t$ y $\varepsilon' \leq 2\varepsilon$. Por lo tanto PR-OTP otorga $(\varepsilon' = 2\varepsilon, t' \approx t)$ -secrecy. $\square$

Comentario 18. Otorgar $(\varepsilon' = 2\varepsilon, t' \approx t)$ -secrecy quiere decir que un atacante contra Π^G tiene al máximo doble de la ventaja de un atacante contra G . En términos de bit-security, $\log_2(\frac{t}{2\varepsilon}) = \log_2(\frac{t}{\varepsilon}) - 1$, i.e. solamente “un bit” de seguridad es perdido.

Si quisiéramos determinar la seguridad concreta en bits, necesitaríamos una construcción concreta de G , la cual determinaría (ε, t) .

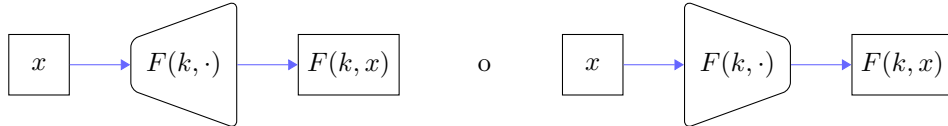
Ahora vamos a ver dos primitivas relacionadas al PRG, utilizadas para construir PRGs y mas.

Definición 13 (PRF). Sean $\mathcal{K}$, $\mathcal{R}$ y $\mathcal{D}$ conjuntos finitos. Sea $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$ una función implementable como algoritmo determinista eficiente.

$\text{Exp}^{\text{PRF}}(\mathcal{A}, b)$	$R(x)$	$\mathcal{O}(x)$
1: $k \xleftarrow{\$} \mathcal{K}$	1: if $x \notin L$:	1: if $b = 0$
2: $L \leftarrow []$	2: $y \xleftarrow{\$} \mathcal{R}$	2: $y \leftarrow F(k, x)$
3: $b' \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}()$	3: $L[x] \leftarrow y$	3: else
4: return b'	4: return $L[x]$	4: $y \leftarrow R(x)$
		5: return y

Decimos que F es una familia de funciones (ε, t, q) -pseudoaleatoria indexada por $k \in \mathcal{K}$ (PRF, “pseudorandom function family”), si dado cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ y hace $\leq q$ queries (consultas) a $\mathcal{O}(\cdot)$, tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$



Comentario 19. Bajo el perfil teórico, un PRG requiere un input aleatorio, y retorna un solo output pseudoaleatorio. Una PRF $F(k, \cdot): \mathcal{D} \rightarrow \mathcal{R}$ tolera q inputs $x_i \in \mathcal{D}$ no necesariamente aleatorios, retornando siempre output pseudoaleatorio.

Clase 6: cifrados de bloque

Fernando Virdia, versión: 0.1, mayo 2025

Definición 14 (PRP). Sean $\mathcal{K}, \mathcal{D}$ dos conjuntos finitos. Sea $\pi: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{D}$ una función, tal que $\forall k \in \mathcal{K}, \pi(k, \cdot): \mathcal{D} \rightarrow \mathcal{D}$ tiene una inversa $\pi^{-1}(k, \cdot): \mathcal{D} \rightarrow \mathcal{D}$ (tal que $\forall x \in \mathcal{D}, \pi(k, \pi^{-1}(k, x)) = \pi^{-1}(k, \pi(k, x)) = x$). Sean π y π^{-1} implementables como algoritmos deterministas eficientes.

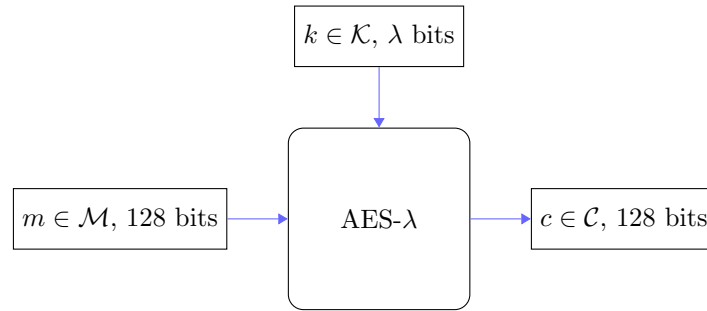
$\text{Exp}^{\text{PRP}}(\mathcal{A}, b)$	$P(x)$	$\mathcal{O}(x)$
1: $k \xleftarrow{\$} \mathcal{K}$	1: if $x \notin L$:	1: if $b = 0$
2: $L \leftarrow []$	2: $y \xleftarrow{\$} \mathcal{D} \setminus \{L[x] \mid \forall x \in L\}$	2: $y \leftarrow \pi(k, x)$
3: $b' \leftarrow \mathcal{A}^{\mathcal{O}(\cdot)}()$	3: $L[x] \leftarrow y$	3: else
4: return b'	4: return $L[x]$	4: $y \leftarrow P(x)$
		5: return y

Decimos que π es una familia de permutaciones (ε, t, q) -pseudoaleatoria indexada por $k \in \mathcal{K}$ (PRP, “pseudorandom permutation family”), o un cifrado de bloques, si dado cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ y hace $\leq q$ queries a $\mathcal{O}(\cdot)$, tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{PRP}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

Comentario 20. Una permutación π con $\mathcal{D} = \{0, 1\}^d$ permuta cadenas en $\mathcal{D}$, no bits en una cadena. Por ejemplo, dado $d = 3$, $\pi(k, 010)$ podría ser 111, aunque estas cadenas tengan diferentes números de bits = 1.

Comentario 21. Por motivos históricos y prácticos, los algoritmos de cifrado de bloque, o PRP, son entre los componentes mas usados (correctamente y no) y estudiados de la criptología. Hoy en día, el mas utilizado es el Advanced Encryption Standard (AES), que tolera $\mathcal{D} = \{0, 1\}^{128}$ y presenta tres variantes: AES-128, con $|k| = 128$, AES-192, con $|k| = 192$, AES-256, con $|k| = 256$. Se estima que AES- λ ofrece $\lambda = \log_2(t/\varepsilon)$ bits de seguridad (ε, t, q) -PRP.



Los algoritmos de cifrado de bloque se utilizan a menudo en “modos de operacion”. Pueden haber oído de ECB-mode (un modo prácticamente siempre inseguro), CBC-, CTR-, GCM-... No vamos a estudiar todos estos, dados que a menudo pensar en términos de modos de operación, en lugar que en términos de modelos de seguridad, puede resultar en la introducción de vulnerabilidades en protocolos. En efecto, vamos a cubrir métodos equivalentes al CTR y GCM.

PRGs, PRFs y PRPs son equivalentes, o sea:

- PRP $\Rightarrow$ PRF: “PRP-PRF switching lemma”
- PRF $\Rightarrow$ PRP: “Feistel networks”
- PRF $\Rightarrow$ PRG: $F(k, x_1) || \dots || F(k, x_q)$
- PRG $\Rightarrow$ PRF: “GGM”

Definición 15 (λ -bits de seguridad bajo q queries). Sea Π una primitiva criptografica, y sea “X” una noción de seguridad con experimento Exp^X , que le otorga al adversario q queries a un oraculo. Supongamos que existe $\lambda \in \mathbb{Z}_{>0}$ tal que para cualquier $\varepsilon \in [0, 1]$, Π otorga (ε, t, q) -X-seguridad con $t \geq 2^\lambda \cdot \varepsilon$. Entonces decimos que Π otorga λ bits de X-seguridad bajo q queries.

En el resto del curso, usaremos este postulado.

Postulado 2. *Dados $q, \lambda \geq 0$, existe un cifrado de bloque Π que otorga λ -bits de seguridad PRP bajo q queries.*

Completamos este tema, construyendo PRFs de PRPs, y PRGs de PRFs.

De PRP a PRF.

La idea de esta construcción es que una PRP es una PRF, si $\mathcal{A}$ observa pocos outputs. Para demostrarlo, primero vemos dos lemas requeridos.

Lemma 5 (Difference lemma, lema de la diferencia). *Sean E_1, E_2, Z tres eventos donde*

$$E_1 \wedge \bar{Z} \iff E_2 \wedge \bar{Z}.$$

Entonces $|\Pr[E_1] - \Pr[E_2]| \leq \Pr[Z]$.

Comentario 22. “Si se necesita Z para distinguir E_1 de E_2 , la probabilidad de distinguirlos es al máximo la probabilidad de que Z suceda.”

Proof. Calculamos

$$\begin{aligned} |\Pr[E_1] - \Pr[E_2]| &= |(\Pr[E_1 \wedge Z] + \cancel{\Pr[E_1 \wedge \bar{Z}]}) - (\Pr[\cancel{E_2 \wedge Z}] + \Pr[E_2 \wedge \bar{Z}])| \\ &= |\Pr[E_1 \wedge Z] - \Pr[E_2 \wedge Z]|. \end{aligned}$$

Dado que $0 \leq \Pr[E_i \wedge Z] \leq \Pr[Z]$, con $i = 1, 2$,

$$\implies |\Pr[E_1 \wedge Z] - \Pr[E_2 \wedge Z]| \leq \Pr[Z].$$

□

Lemma 6 (Birthday bound, paradoja del cumpleaños). *Dados $q \in \mathbb{Z}_{>0}$, $\mathcal{R}$ un conjunto finito, y $z_1, \dots, z_q \in \mathcal{R}$ muestreados $z_i \sim U(\mathcal{R})$ de manera independiente,*

$$\Pr[\exists i, j \in [q] : i \neq j \wedge z_i = z_j] \leq \frac{q(q-1)}{2|\mathcal{R}|}$$

Proof. Calculamos

$$\begin{aligned} \Pr[\exists i, j \in [q] : i \neq j \wedge z_i = z_j] &= \Pr \left[\bigvee_{i \neq j} z_i = z_j \right] \\ &\leq \sum_{i \neq j} \Pr[z_i = z_j] \quad (\text{por la cota de la unión}) \\ &= \sum_{i \neq j} \frac{1}{|\mathcal{R}|} = \binom{q}{2} \frac{1}{|\mathcal{R}|} = \frac{q(q-1)}{2|\mathcal{R}|}. \end{aligned}$$

□

Comentario 23. *El birthday bound es una cota superior “tight” o “ajustada”, o sea que el valor efectivo de la probabilidad de colisión es muy cercano a la cota. Efectivamente, también se puede demostrar una cota inferior muy cercana, de $\approx \frac{q(q-1)}{4|\mathcal{R}|}$ [KL20, Lemma A.15].*

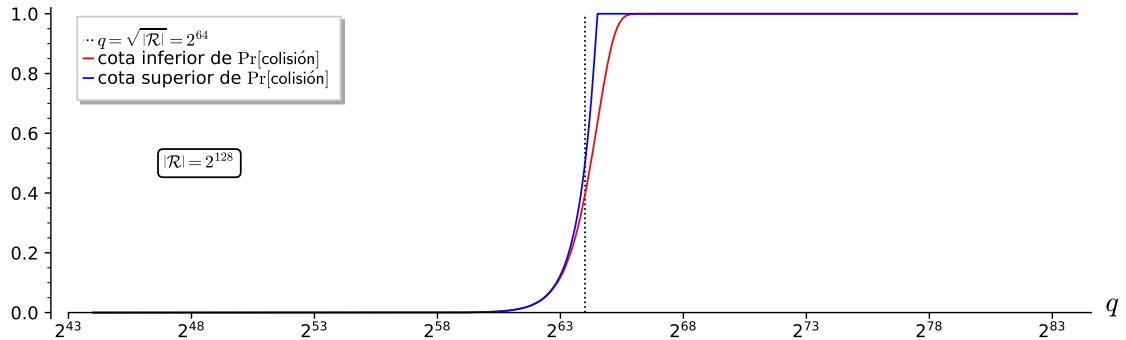


Figure 3: Visualizando las cotas en función de q , vemos que la probabilidad “salta” de ≈ 0 a ≈ 1 alrededor de $q = \sqrt{|\mathcal{R}|}$.

Clase 7: PRP $\Rightarrow$ PRF

Fernando Virdia, versión: 0.1, mayo 2025

Lemma 7 (PRP-PRF switching lemma). Sea $\pi: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{D}$ una (ε, t, q) -PRP. Entonces π es una (ε', t', q') -PRF con $t' = t$, $q' = q$, $\varepsilon' = \varepsilon + \frac{q(q-1)}{2|\mathcal{D}|}$.

Proof. Supongamos que $\mathcal{A}$ sea un adversario contra π usado como una PRF con $\mathcal{R} = \mathcal{D}$, y que evalúa $\mathcal{O}(\cdot)$ sobre inputs distintos, sin pérdida de generalidad. Observando $\text{Exp}^{\text{PRF}}(\mathcal{A}, 0)$ y $\text{Exp}^{\text{PRP}}(\mathcal{A}, 0)$, notamos que del punto de vista de $\mathcal{A}$ son experimentos idénticos si $F = \pi$ en Exp^{PRF} ,

$$\Rightarrow \Pr[\text{Exp}^{\text{PRF}}(\mathcal{A}, 0) \Rightarrow 1] = \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 0) \Rightarrow 1].$$

Calculamos,

$$\begin{aligned} \text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{A}) &= \left| \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{A}, 1) \Rightarrow 1] \right| \\ &= \left| \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 0) \Rightarrow 1] - \underbrace{\Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 1) \Rightarrow 1] + \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 1) \Rightarrow 1]}_{=0} - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{A}, 1) \Rightarrow 1] \right| \\ &\leq \text{Adv}(\text{Exp}^{\text{PRP}}, \mathcal{A}) + \left| \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 1) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{A}, 1) \Rightarrow 1] \right|. \quad (\text{por desig. triangular}) \end{aligned}$$

Ahora notamos que $\text{Exp}^{\text{PRP}}(\mathcal{A}, 1)$ y $\text{Exp}^{\text{PRF}}(\mathcal{A}, 1)$ son idénticos desde el punto de vista de $\mathcal{A}$, mientras tanto que el siguiente evento no sucede:

$$Z = \text{"}\exists i, j \in [q] : i \neq j \wedge R(x_i) = R(x_j) \text{ durante } \text{Exp}^{\text{PRF}}(\mathcal{A}, 1)\text{"}$$

Usando el lema de la diferencia,

$$\left| \Pr[\text{Exp}^{\text{PRP}}(\mathcal{A}, 1) \Rightarrow 1] - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \Pr[Z].$$

Dado que $R(\cdot)$ muestrea $U(\mathcal{R}) = U(\mathcal{D})$ de manera independiente si los inputs son distintos (en cuanto π es una PRP), por la paradoja del cumpleaños $\Pr[Z] \leq \frac{q(q-1)}{2|\mathcal{R}|}$, y de consecuencia

$$\text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{A}) \leq \text{Adv}(\text{Exp}^{\text{PRP}}, \mathcal{A}) + \frac{q(q-1)}{2|\mathcal{D}|}.$$

□

Comentario 24. El PRP-PRF switching lemma nos permite construir una PRF con $\mathcal{R} = \mathcal{D}$. Existen construcciones con $\mathcal{R} \neq \mathcal{D}$, como la PRF “GGM”.

Comentario 25. Este tipo de diferencia en ventaja en función de número de queries implica a menudo la necesidad de limitar la duración de las claves secretas usadas. Si en este caso $q \approx \sqrt{\mathcal{D}}$ permitiría distinguir una PRP de una PRF. A menudo esto implica concretamente ataques concretos, como <https://sweet32.info>.

Lemma 8 (PRF $\Rightarrow$ PRG). Sea F una (ε, t, q) -PRF con $\mathcal{R} = \{0, 1\}^r$. Sean $x_1, \dots, x_q \in \mathcal{D}$ elementos distintos ($x_i \neq x_j, \forall i \neq j$). Definimos $G: \mathcal{K} \rightarrow \{0, 1\}^{r \cdot q}$,

$\begin{aligned} &G(k \in \mathcal{K}) \\ &1: \quad \textbf{for } i = 1, \dots, q: \\ &2: \quad \quad y_i \leftarrow F(k, x_i) \\ &3: \quad \textbf{return } y_1 \dots y_q \end{aligned}$

G es un (ε', t') -PRG con $\varepsilon' = \varepsilon$, $t' \approx t$.

Proof. Supongamos un adversario $\mathcal{A}$ que juega $\text{Exp}^{\text{PRG}}(\mathcal{A}, b)$. Sea $\mathcal{B}$ el siguiente adversario contra la PRF.

$\mathcal{B}^{\mathcal{O}}$
1 : for $i = 1, \dots, q$:
2 : $y_i \leftarrow \mathcal{O}(x_i)$
3 : $b' \leftarrow \mathcal{A}(y_1 \dots y_q)$
4 : return b'

Si $b = 1$ en Exp^{PRF} , $\mathcal{O}(\cdot) = \mathcal{R}(\cdot)$,

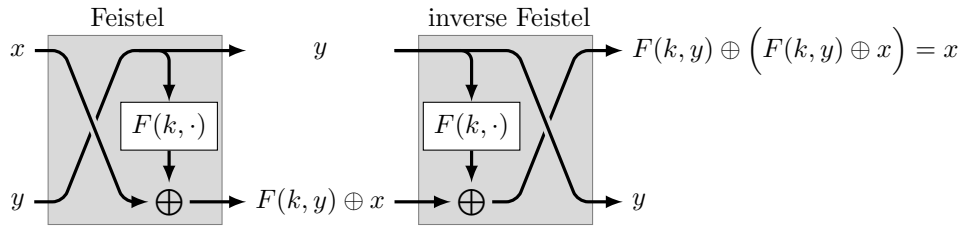
$$\begin{aligned} &\Rightarrow y_i \sim U(\{0, 1\}^r) \\ &\Rightarrow y \sim U(\{0, 1\}^{r \cdot q}) \\ &\Rightarrow \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1] = \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 1) \Rightarrow 1]. \end{aligned}$$

Si $b = 0$, $\mathcal{O}(\cdot) = F(k, \cdot)$ con $k \sim U(\mathcal{K})$,

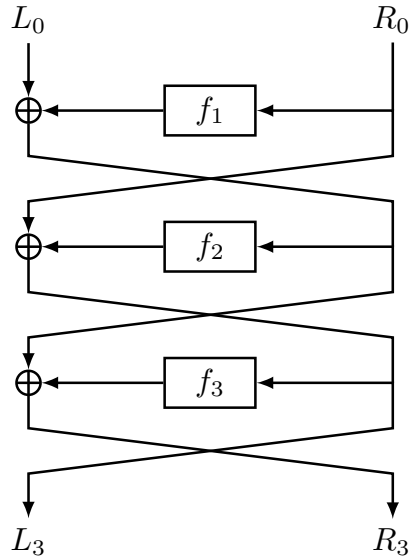
$$\begin{aligned} &\Rightarrow y_i = F(k, x_i) \\ &\Rightarrow y = G(k) \\ &\Rightarrow \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 1] = \Pr[\text{Exp}^{\text{PRG}}(\mathcal{A}, 0) \Rightarrow 1]. \end{aligned}$$

Sigue que $\text{Adv}(\text{Exp}^{\text{PRG}}, \mathcal{A}) = \text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B})$, y claramente $t' \approx t$. $\square$

No vamos a ver GGM y Feistel en detalle, dado que las demostraciones se vuelven bastante difíciles.



(a) Una ronda de Feistel, junto a su inversa.



(b) Múltiples rondas.

Figure 4: PRF $\Rightarrow$ PRP: Feistel network. Imágenes adaptadas de [Ros21, Constructions 6.9, 6.11], usadas con permiso del autor.

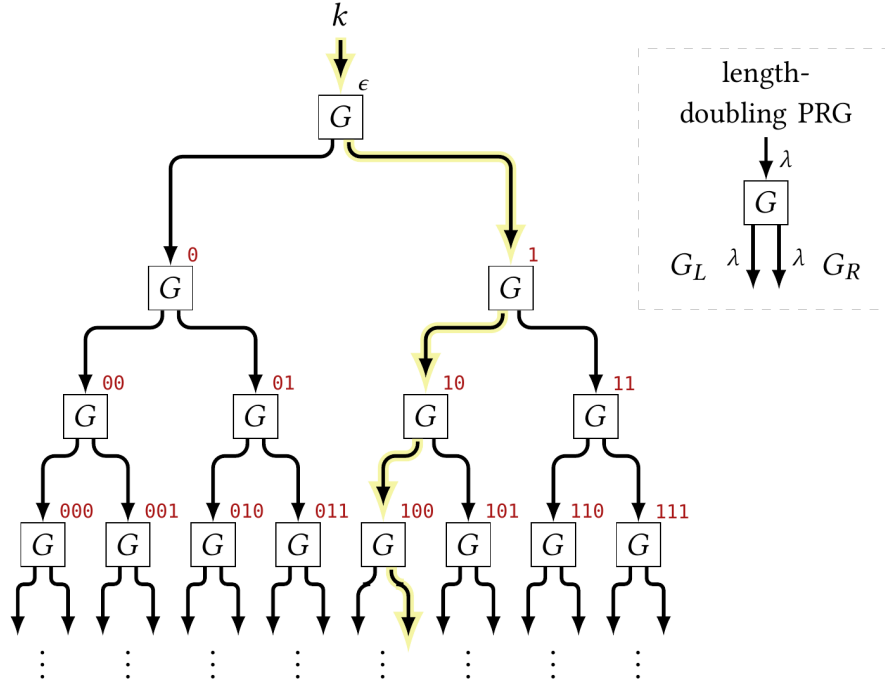


Figure 5: PRG $\Rightarrow$ PRF: construcción GGM (Goldreich-Goldwasser-Micali). Credito de imagen: [Ros21].

Comentario 26. No toda combinación de primitivos otorga seguridad. Por que lo siguiente no funciona?

- Sea G un PRG. $F(k, x) := G(k) \oplus x$ no es una PRF.
- Sea F una PRF. $G(k) := F(1, k) || \dots || F(q, k)$ no es (necesariamente) un PRG.
- Sea G un PRG. $G'(k) := G(k) || G(k)$ no es un PRG.
- Sea G un PRG. $G'(k_1 || k_2) := G(k_1) \wedge G(k_2)$ no es un PRG.

Un resumen:

Inicialmente empezamos con el OTP: secreto perfecto, pero $|k| = |m|$.

Introducimos PRG/PRF/PRP: (ϵ, t) -secrecy, pero $|k| < |m|$.

Aun tenemos dos problemas:

- No podemos cifrar mas de un mensaje: $c_1 \oplus c_2 = m_1 \oplus m_2$.
- No podemos garantizar la integridad de los mensajes: $\text{Dec}(k, c \oplus \Delta) = m \oplus \Delta$.

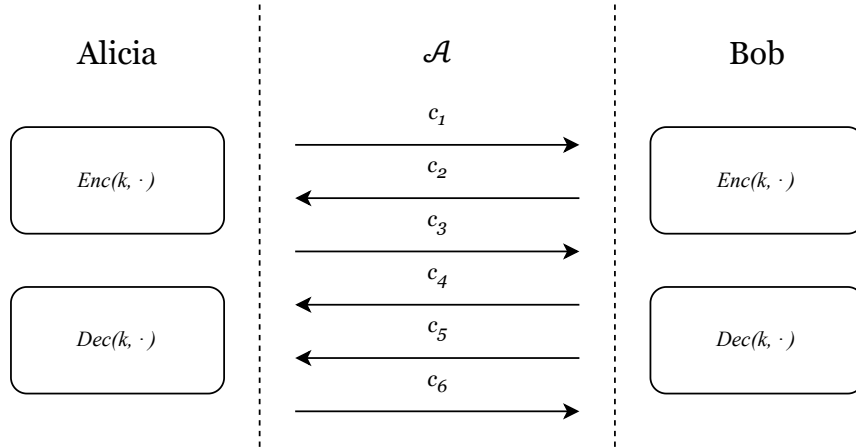
Usando PRFs, podemos resolver ambos.

Clase 8: confidencialidad para mensajes múltiples

Fernando Virdia, versión: 0.1, mayo 2025

4 Cifrar más de un mensaje

Como anteriormente, queremos poder usar un SKES, pero generar mas de un mensaje.



PREGUNTA: Ideas sobre como definiré seguridad “multi-mensajes”?

Podríamos permitirle a $\mathcal{A}$ de ver $c_1, \dots, c_n$. Vamos a hacer algo mas: le vamos a permitir pedir el cifrado de mensajes arbitrarios! O sea: sin darle la clave k , le vamos a otorgar un oráculo que calcula $\text{Enc}(k, \cdot)$.

PREGUNTA: Esto podría parecer insólito: como podría $\mathcal{A}$ acceder a $\text{Enc}(k, \cdot)$? Ideas?

- Los mensajes podrían ser previsibles, dándole a $\mathcal{A}$ m y $\text{Enc}(k, m)$.
- $\mathcal{A}$ podría tener control del input directamente, por ejemplo si es un comerciante que introduce en el POS el valor a cobrar, y quiere usar $(m, \text{Enc}(k, m))$ para aprender la clave k .
- $\mathcal{A}$ podría saber que el mensaje cifrado es consecuente a alguna acción propia.

Nos inspiramos a la definición de (ϵ, t) -secrecy (c.f. Definition 12), donde el adversario trata de distinguir $\text{Enc}(k, m_0)$ de $\text{Enc}(k, m_1)$. En este caso imponemos $|m_0| = |m_1|$ dado que toleraremos $\mathcal{M} = \{0, 1\}^*$.

Definición 16 (IND-CPA security). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un SKES.

$\text{Exp}^{\text{CPA}}(\mathcal{A}, b)$	$\mathcal{O}(m_0, m_1)$
1: $k \leftarrow \text{Gen}()$	1: if $ m_0 \neq m_1 $
2: $b' \leftarrow \mathcal{A}^{\mathcal{O}}()$	2: return $\perp$
3: return b'	3: $c \leftarrow \text{Enc}(k, m_b)$
	4: return c

Decimos que Π otorga (ϵ, t, q) -indistinguibilidad bajo ataques de mensaje elegido (IND-CPA, “indistinguishability under chosen-plaintext attacks”), si para todo adversario $\mathcal{A}$ que corre en tiempo $\leq t$ y hace $\leq q$ queries a $\mathcal{O}$, su ventaja es

$$\text{Adv}(\text{Exp}^{\text{CPA}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{CPA}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{CPA}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \epsilon.$$

Comentario 27. Las definiciones de (ε, t) -secret y $(\varepsilon, t, 1)$ -IND-CPA son equivalentes, por lo que (ε, t) -secrety a veces se lo llama (ε, t) -OT-IND-CPA.

Comentario 28. Existen definiciones “intermedias” entre (ε, t) -secrety y (ε, t, q) -IND-CPA. Como referencia, vean [KL20] o [BS23].

Ejemplo 2.

- El OTP no otorga $(\varepsilon, t, 2)$ -IND-CPA, dado que $\mathcal{A}$ puede pedir $\mathcal{O}(0^n, 0^n) \Rightarrow k$, y obtenida k , puede pedir $\mathcal{O}(0^n, 1^n) \Rightarrow b^n \oplus k$, y así recuperar b .
- Ningún SKES donde $\text{Enc}(k, \cdot)$ es determinista puede otorgar $(\varepsilon, t, 2)$ -IND-CPA! En cuanto el siguiente ataque aplica:

1. $c_1 \leftarrow \mathcal{O}(0^n, 0^n)$
2. $c_2 \leftarrow \mathcal{O}(0^n, 1^n)$
3. Si $c_1 = c_2$, entonces $b = 0$. Si $c_1 \neq c_2$, entonces $b = 1$.

Comentario 29 (SKE OT-IND-CPA $\not\Rightarrow$ SKE $(q \geq 2)$ -IND-CPA). Como vimos en Example 2, es fácil identificar algoritmos de SKE que no otorgan seguridad IND-CPA cuando mas de 1 query es otorgada al adversario. De otro lado, el OTP puede otorgar (ε, t) -secrety, o sea “OT-IND-CPA” (c.f. Remark 27), o $(\varepsilon, t, q = 1)$ -IND-CPA. Esto quiere decir que en el caso de cifrado simétrico, o SKE,

$$(\varepsilon, t, q = 1)\text{-IND-CPA} \not\Rightarrow (\varepsilon', t'q' > 1)\text{-IND-CPA con } t'/\varepsilon' \approx t/\varepsilon.$$

Definición 17 (PRF-CTR). Sea $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$ una (ε, t, q) -PRF con $\mathcal{D} = \{0, 1\}^d$ y $\mathcal{R} = \{0, 1\}^r$. Definimos un SKES $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ con $\mathcal{M} = \{0, 1\}^{\ell \cdot r}$ y $\mathcal{C} = \{0, 1\}^{\ell \cdot r + d}$, para algún $\ell \leq 2^d/2$.

Gen()	Enc(k, m)	Dec($k, c = (c_0, c_1, \dots, c_\ell)$)
1: $k \xleftarrow{\$} \mathcal{K}$	1: $m_1 \dots m_\ell \leftarrow m$, donde $ m_i = r$	1: for $i = 1, \dots, \ell$:
2: return k	2: $c_0 \xleftarrow{\$} [0, 2^d)$, equiv. $\{0, 1\}^d$	2: $y_i \leftarrow F(k, c_0 + i - 1 \bmod 2^d)$
	3: for $i = 1, \dots, \ell$:	3: $m'_i \leftarrow y_i \oplus c_i$
	4: $y_i \leftarrow F(k, c_0 + i - 1 \bmod 2^d)$	4: $m' \leftarrow m'_1 \dots m'_\ell$
	5: $c_i \leftarrow y_i \oplus m_i$	5: return m'
	6: $c \leftarrow (c_0, c_1, \dots, c_\ell)$	
	7: return c	

Si usamos una PRP/cifrado de bloques en lugar de F , Π es el modo de cifrado “counter mode” (CTR).

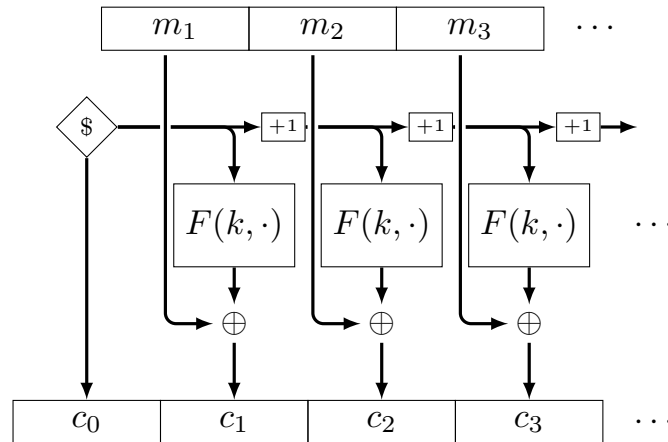


Figure 6: Diagrama de Enc en PRF-CTR. Imagen adaptada de [Ros21, Construction 8.3], usada con permiso del autor.

Comentario 30. El PRF-CTR puede ser paralelizado de manera simple, dado que los y_i pueden ser calculados de manera independiente. Por lo tanto tiene muy buena eficiencia al cifrar mensajes muy largos.

Comentario 31. *Para descifrar, no es necesario “invertir” F . Esto sugiere que si m es largo entre $(\ell - 1) \cdot r + 1$ y $\ell \cdot r$, se pueden “cortar” los últimos bits de y_n en Enc , resultando en un cifrado mas corto.*

Clase 9: seguridad del cifrado “modo counter”

Fernando Virdia, versión: 0.1, mayo 2025

Teorema 5. Sean F, Π como en Definition 17. Si F es una (ε, t, q) -PRF, Π es (ε', t', q') -IND-CPA con $t' \approx t$, $q' = q/\ell$, $\varepsilon' = 2 \cdot \varepsilon + 2 \cdot q^2 \cdot \ell/|\mathcal{D}|$.

Proof. Dado un adversario $\mathcal{A}$ que juega Exp^{CPA} , definimos $\mathcal{B}$ que juega Exp^{PRF} teniendo acceso a un oráculo $\mathcal{O}$ definido como en Definition 13,

$\mathcal{B}^{\mathcal{O}(\cdot)}$	$E(m_0, m_1)$	$\widetilde{\text{Enc}}^{\mathcal{O}}(m)$
1: $\hat{b} \xleftarrow{\$} \{0, 1\}$	1: $c \leftarrow \widetilde{\text{Enc}}^{\mathcal{O}}(m_{\hat{b}})$	Lo mismo de Enc en Definition 17,
2: $\hat{b}' \leftarrow \mathcal{A}^E()$	2: return c	pero reemplazando $F(k, \cdot)$ con $\mathcal{O}(\cdot)$
3: return $[\hat{b}' = \hat{b}]$		

Notamos que el entorno de $\mathcal{A}$ en $\text{Exp}^{\text{PRF}}(\mathcal{B}, 0)$ es idéntico a la variante “bit-guessing” $\widetilde{\text{Exp}}^{\text{CPA}}(\mathcal{A})$. Sigue que

$$\Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 1] = \Pr[\widetilde{\text{Exp}}^{\text{CPA}}(\mathcal{A}) \Rightarrow 1].$$

Lamentablemente, no es claro como calcular $\Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1]$ directamente. Por lo tanto, definimos un nuevo experimento $\widetilde{\text{Exp}}(\mathcal{B})$, idéntico a $\text{Exp}^{\text{PRF}}(\mathcal{B}, 1)$ pero donde reemplazamos el oráculo $\mathcal{O}$ que le viene dado a $\mathcal{B}$:

- En $\text{Exp}^{\text{PRF}}(\mathcal{B}, 1)$, $\forall x \mathcal{O}(x) := R(x)$, donde R es una función muestreada aleatoriamente.
- En $\widetilde{\text{Exp}}(\mathcal{B})$, reemplazamos $\mathcal{O}(x)$ con un **algoritmo** aleatorio $\hat{R}(x) = \text{“return } \$ \xleftarrow{\$} \{0, 1\}^r \text{”}$ que retorna elementos muestreados de $U(\{0, 1\}^r)$, ignorando el valor x en input. Esto quiere decir que mientras $\Pr[R(x) \neq R(x)] = 0$, $\Pr[\hat{R}(x) \neq \hat{R}(x)] = 1 - 2^{-r}$, en cuanto $\hat{R}$ **no es una función**.

Sea $G := \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1] - \Pr[\widetilde{\text{Exp}}(\mathcal{B}) \Rightarrow 1] \right|$. Definir $\widetilde{\text{Exp}}$ nos permite calcular,

$$\begin{aligned}
 \Pr[\widetilde{\text{Exp}}^{\text{CPA}}(\mathcal{A}) \Rightarrow 1] &= \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 1] \right| \\
 &= \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 1] - \underbrace{\Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1] + \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1]}_{=0} \right| \\
 &\leq \text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B}) + \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1] - \underbrace{\Pr[\widetilde{\text{Exp}}(\mathcal{B}) \Rightarrow 1] + \Pr[\widetilde{\text{Exp}}(\mathcal{B}) \Rightarrow 1]}_{=0} \right| \\
 &\leq \text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B}) + G + \Pr[\widetilde{\text{Exp}}(\mathcal{B}) \Rightarrow 1]
 \end{aligned}$$

Empezamos calculando $\Pr[\widetilde{\text{Exp}}(\mathcal{B}) \Rightarrow 1]$. La vista de $\mathcal{A}$ en $\widetilde{\text{Exp}}(\mathcal{B})$ es tal que la i -ésima query a E retorna

$$E(m_{0,i}, m_{1,i}) = (c_{0,i}, m_{\hat{b},i,1} \oplus \hat{R}(z_{i,1}), \dots, m_{\hat{b},i,j} \oplus \hat{R}(z_{i,j}), \dots, m_{\hat{b},i,\ell} \oplus \hat{R}(z_{i,\ell}))$$

donde $z_{i,j} = c_{0,i} + j - 1 \bmod 2^d$, y donde $\hat{R}(z_{i,j}) \sim_{\text{iid}} U(\{0, 1\}^r)$. En particular, esta vista es independiente de $\hat{b}$ dado que cada bit de $E(m_{0,i}, m_{1,i})$ esta uniformemente distribuido por definicion de $c_{0,i}$ y de $\hat{R}$,

$$\begin{aligned}
 \Pr[m_{0,i,j} \oplus \hat{R}(z_{i,j}) = c_{j,i}] &= \Pr[\hat{R}(z_{i,j}) = c_{j,i} \oplus m_{0,i,j}] \\
 &= 2^{-r} \\
 &= \Pr[\hat{R}(z_{i,j}) = c_{j,i} \oplus m_{1,i,j}] \\
 &= \Pr[m_{1,i,j} \oplus \hat{R}(z_{i,j}) = c_{j,i}].
 \end{aligned}$$

Siendo la vista de $\mathcal{A}$ completamente independiente de $\hat{b}$ en $\widetilde{\text{Exp}}(\mathcal{B})$, así tambien lo es la salida $\hat{b}' \leftarrow \mathcal{A}^E()$, por lo que

$$\Pr[\hat{b}' = \hat{b}] = \Pr[\widetilde{\text{Exp}}(\mathcal{B}) \Rightarrow 1] = \frac{1}{2},$$

y de consecuencia

$$\text{Adv}(\overline{\text{Exp}}^{\text{CPA}}, \mathcal{A}) = \left| \Pr[\overline{\text{Exp}}^{\text{CPA}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| \leq \text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B}) + G.$$

Nos queda calcular G , la ventaja de distinguir $\text{Exp}^{\text{PRF}}(\mathcal{B}, 1)$ de $\widetilde{\text{Exp}}(\mathcal{B})$. Notamos que la sola diferencia entre estos juegos es que $R(\cdot)$ es una función dentro de $\text{Exp}^{\text{PRF}}(\mathcal{B}, 1)$, por lo que $R(x)$ es un valor fijo, y $\hat{R}$ dentro de $\widetilde{\text{Exp}}(\mathcal{B})$ muestra uniformemente de $\{0, 1\}^r$, por lo que $\hat{R}(x)$ no tiene un valor fijo. Esta diferencia puede solo ser notada si $\mathcal{O}(\cdot)$ viene llamado sobre un mismo input $z_{i,j}$ mas de una vez durante el experimento. Sea

$$Z := “\exists i, i' \in [q], j, j' \in [\ell] : (i, j) \neq (i', j') \wedge z_{i,j} = z_{i',j'}, \text{ en } \text{Exp}^{\text{PRF}}(\mathcal{B}, 1)”$$

el evento por el cual $\mathcal{O}(\cdot)$ viene llamado sobre un input $z_{i,j} = z_{i',j'}$ repetido durante el experimento $\text{Exp}^{\text{PRF}}(\mathcal{B}, 1)$. Notamos que $\text{Exp}^{\text{PRF}}(\mathcal{B}, 1)$ y $\widetilde{\text{Exp}}(\mathcal{B})$ son idénticos si Z no ocurre. Por lo tanto, usando el lema de la diferencia,

$$G = \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1] - \Pr[\widetilde{\text{Exp}}(\mathcal{B}) \Rightarrow 1] \right| \leq \Pr[Z].$$

Sea

$$\text{coll}_{i,i'} := “\exists j, j' \in [\ell] : (i, j) \neq (i', j') \wedge z_{i,j} = z_{i',j'} \text{ en } \text{Exp}^{\text{PRF}}(\mathcal{B}, 1)”.$$

Por suposición, $\ell \leq 2^d/2$ (c.f. Definition 17). Esto quiere decir que

$$\Pr[\text{coll}_{i,i'} \mid i' = i] = 0,$$

dado que

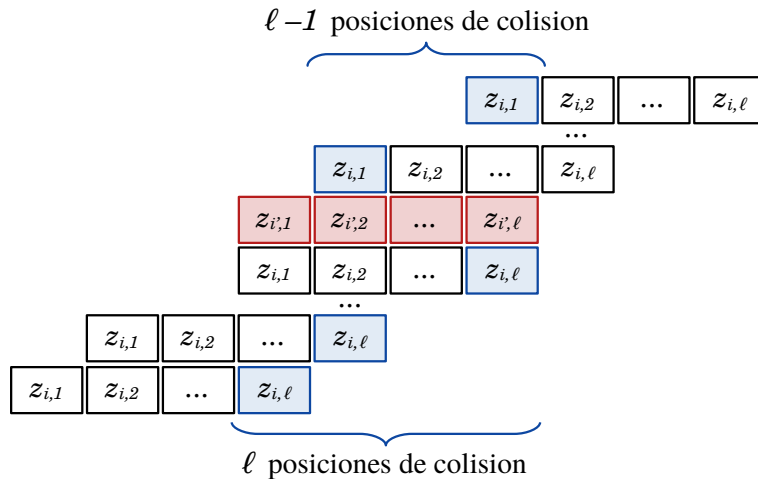
$$\begin{aligned} z_{i,j} = z_{i,j'} \wedge j \neq j' &\iff c_{0,i} + j - 1 = c_{0,i} + j' - 1 \bmod 2^d \wedge j \neq j' \\ &\iff j = j' \bmod 2^d \wedge j \neq j' \\ &\iff j = j' + \Delta 2^d \text{ donde } \Delta \in \mathbb{Z} \setminus \{0\} \\ &\iff \text{la } i\text{-ésima query es } \ell \geq 2^d \text{ bloques larga,} \end{aligned}$$

pero $\ell \leq 2^d/2$. Por lo tanto

$$\Pr[\text{coll}_{i,i'}] = \Pr[\text{coll}_{i,i'} \mid i = i'] \Pr[i = i'] + \Pr[\text{coll}_{i,i'} \mid i \neq i'] \Pr[i \neq i'] \leq \Pr[\text{coll}_{i,i'} \mid i \neq i'].$$

Supongamos entonces que $i \neq i'$. Queremos calcular la probabilidad de una colisión entre

$$(z_{i,1}, \dots, z_{i,\ell}) \quad \text{y} \quad (z_{i',1}, \dots, z_{i',\ell}).$$



Para que haya una colisión, $z_{i,1}$ tiene que tener un valor que sobreponga las dos secuencias. Solo hay $2 \cdot \ell - 1$ tales valores posibles de $z_{i,1} = c_{0,i}$ para que suceda una colisión, donde $c_{0,i}$ es muestrado aleatoriamente en $\widetilde{\text{Enc}}$. Esto implica que $\Pr[\text{coll}_{i,i'} \mid i \neq i'] = \frac{2 \cdot \ell - 1}{2^d}$. Finalmente, podemos calcular

$$\Pr[Z] = \Pr \left[\bigvee_{i \neq i'} \text{coll}_{i,i'} \right] \leq \sum_{i \neq i'} \Pr[\text{coll}_{i,i'}] \leq \binom{q}{2} \frac{2 \cdot \ell - 1}{2^d} \leq \frac{q \cdot (q - 1)}{2} \cdot \frac{2 \cdot \ell}{2^d} \leq \frac{q^2 \cdot \ell}{2^d}$$

Con esto, podemos completar el calculo de la ventaja de $\mathcal{A}$, recordando que

$$\text{Adv}(\text{Exp}^{\text{CPA}}, \mathcal{A}) = 2 \cdot \text{Adv}(\overline{\text{Exp}^{\text{CPA}}}, \mathcal{A}) \leq 2 \cdot \left(\text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B}) + \Pr[Z] \right) \leq 2 \cdot \left(\text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B}) + \frac{q^2 \cdot \ell}{2^d} \right),$$

lo que nos da $t' \approx t$, $q' = q/\ell$, y $\varepsilon' = 2 \cdot \varepsilon + \frac{2 \cdot q^2 \cdot \ell}{2^d}$, donde $|\mathcal{D}| = 2^d$. $\square$

Comentario 32. Nuevamente vemos una diferencia en ventaja relacionada a una probabilidad de colisión. Este tipo de ataque es verosímil, como en el caso del ataque <https://sweet32.info>.

Finalmente podemos cifrar múltiples mensajes de ℓ “bloques” de r bits de largo. Si un mensaje es $> \ell$ bloques, se lo puede transmitir como dos mensajes. Formalmente, se lo puede demostrar como un lema.

Lemma 9 (fixed-length IND-CPA $\Rightarrow$ variable-length IND-CPA). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un SKES que otorga seguridad (ε, t, q) -IND-CPA. Sea $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ un SKES con

- $\text{Gen}' = \text{Gen}$,
- $\text{Enc}'(k, (m_1, m_2)) := (\text{Enc}(k, m_1), \text{Enc}(k, m_2))$,
- $\text{Dec}(k, (c_1, c_2)) := (\text{Dec}(k, c_1), \text{Dec}(k, c_2))$.

Π' es (ε', t', q') -IND-CPA donde $\varepsilon' = \dots$, $t' \approx \dots$, $q' = \dots$.

EJERCICIO: Determina (ε', t', q') .

EJERCICIO: El Lemma 9 dice que podemos utilizar multiples cifrados IND-CPA para obtener fixed-length IND-CPA. El OTP es OT-IND-CPA, pero no puede ser re-utilizado. Por que esta no es una contradicción?

Clase 10: seguridad informal y funciones de hash

Fernando Virdia, versión: 0.1, mayo 2025

5 Funciones de hash

Las funciones de hash son el primitivo criptográfico probablemente mas conocido. Si alguna vez intentarlo programar el login de una pagina web, probablemente las hayan encontrado (MD5, SHA1-3 las mas conocidas). Son también un primitivo a menudo usado incorrectamente, poniendo cuentas online en riesgo.

Definición 18 (Seguridad de funciones de hash, informal). *Una función $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ se dice de “hash criptografico” si satisface las siguientes propiedades:*

- *Puede ser implementada con un algoritmo determinista eficiente.*
- *“Preimage resistance”:* dado $h \in H(\{0, 1\}^*) \subset \{0, 1\}^\ell$, es difícil encontrar x tal que $H(x) = h$.
- *“Second preimage resistance”:* dado $(x, H(x))$, es difícil encontrar $x' \neq x$ tal que $H(x') = H(x)$.
- *“Collision-resistance” (CR):* es difícil encontrar $x \neq x'$ tal que $H(x) = H(x')$.

Comentario 33. *Por que definiciones informales? La propiedad generalmente requerida de una función de hash es la de collision-resistance. En la practica, las funciones de hash son fijas, no requiriendo ninguna clave. Para una tal funcion, existe un adversario eficiente que recupera con probabilidad maxima una colisión!*

- *Dado que $|\{0, 1\}^*| > |\{0, 1\}^\ell|$, colisiones existen necesariamente. Como H es una función fija, supongamos que x, x' sean una colisión. $\boxed{\mathcal{A} := \text{return } (x, x')}$ es un algoritmo eficiente que retorna una colisión con probabilidad 1. Simplemente, no lo conocemos.*

Es posible obtener definiciones formales de (second-)preimage resistance parametrizando x' y h como variables aleatorias. De todos modos, en la practica las funciones de hash a menudo son utilizadas en entornos de baja entropia, por ejemplo para processar contraseñas. Esto limita la seguridad teorica otorgable dado que generalmente las contraseñas son muestradas de distribuciones con muy baja entropia.

Sin definiciones formales, demostrar seguridad de construcciones que las usan no es posible de manera estrictamente formal. Estrategias comunes incluyen:

- Usar el “modelo del oráculo aleatorio” (ROM, “random oracle model”), donde suponemos que H es una función completamente aleatoria.
- Suponer que la funcion de hash toma en input una clave (publica), aunque en realidad no es así.
- Construir un adversario generico (respecto a H) que encuentra colisiones como parte de la demostración, y suponer que en la practica este adversario efectivamente no contiene suficiente informacion sobre H para resultar en colisiones.

Lemma 10 (informal). *Sea H una función de hash.*

1. *Collision resistance $\Rightarrow$ second-preimage resistance.*
2. *Second-preimage resistance $\Rightarrow$ preimage resistance.*
3. *Collision resistance $\not\Rightarrow$ “partial” preimage resistance.*

Demostración (informal). Por casos:

1. Sea $\mathcal{A}$ un adversario que rompe second-preimage resistance: dado $(x, H(x))$ retorna $x' \neq x$ tal que $H(x') = H(x)$. Podemos elegir un $x \in \{0, 1\}^*$ cualquiera, calcular $x' \leftarrow \mathcal{A}(x, H(x))$. Siendo x' una second-preimage, $x' \neq x$. Por lo tanto podemos retornar la colisión (x, x') .

2. Sea $\mathcal{B}$ un adversario que rompe preimage resistance: dado h retorna x tal que $H(x) = h$. En input $(x, H(x))$ podemos calcular $x' \leftarrow \mathcal{B}(H(x))$. Siendo el dominio de H infinito, probablemente $x \neq x'$, y por lo tanto x' es una second preimage.
3. Aunque collision-resistance $\implies$ second-preimage resistance, y esta $\implies$ preimage resistance, es posible construir funciones que son collision resistant pero para las cuales existe un adversario que rompe preimage resistance sobre un subconjunto bien definido del espacio de input.

Sea G una función de hash CR. Definimos $H(x) := \begin{cases} 0||x & \text{si } |x| = n, \\ 1||G(x) & \text{si } |x| \neq n. \end{cases}$

Por inspección, H es CR dado que colisiones donde $H(x) = 0 \dots$ son imposibles y colisiones con $H(x) = 1 \dots$ implican colisiones en G . Sin embargo, si $h = H(x) = 0 \dots$, x puede ser recuperado dado que $h = 0||x$.

□

Una construcción clásica: Merkle-Damgard.

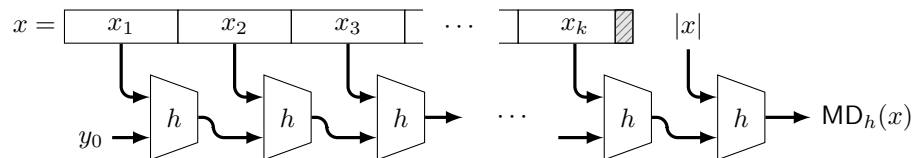


Figure 7: Una manera común de construir funciones de hash es usando PRPs o PRFs iteradas según el diseño de Merkle-Damgard (MD). Imagen tomada de [Ros21, Construction 11.2], usada con permiso del autor.

En la practica, es comun que funciones de hash sean construidas a partir de permutaciones de funciones sobre conjuntos de input finito. Por ejemplo, un paradigma comun de construcción es el de Merkle-Damgard (Figure 7), donde una “funcion de compresión” h comprime iteradametne bloques de bits. Es posible demostrar que si h es collision-resistant, así mismo lo es H .

5.1 Side quest: hashing de contraseñas

Imaginen tener que desarrollar un sistema de login para una pagina web. **Idealmente:** quieren investigar como se usan las librerías de “2-factor authentication”, y protocolos tipo FIDO. Tradicionalmente, se usan contraseñas. Los siguientes son dos escenarios comunes e inseguros.

- El servidor guarda una base de datos con una tabla

usuario	clave
maria@gmail.com	“messi10”
⋮	⋮

Cual es el problema? Probablemente Maria utiliza esa clave en otras paginas. Si el server viene vulnerado y el DB es leakado, $\mathcal{A}$ podría obtener acceso a cuentas mas importantes.

- Solución que se encuentra en demaciados tutoriales:

usuario	clave
jorge@gmail.com	$H(\text{“03/14/1994”})$
⋮	⋮

$\text{SHA-256}(\text{“03/14/1994”}) = \text{ac144fd70bcfcc4} \dots \text{c83b1e0a36ca}$, y H es preimage- y collision-resistant, por lo cual recuperar la clave (o una equivalente) es difícil.

Cual es el problema?

1. H es una función fija
2. La clave tiene “poca entropía”.

Si $\mathcal{A}$ leakea el DB, puede comparar los hashes con valores de input comunes para los cuales precomputó el output. Tablas de tales valores se llaman comúnmente “rainbow tables”. Software tipo “John the Ripper”⁶ puede ser usado junto a rainbow tables para recuperar inputs. Si $\mathcal{A}$ conoce $H(\text{“01/01/1900”})$, $H(\text{“02/01/1900”})$, $\dots$, $H(\text{“31/12/2024”})$, va a poder reconocer la clave de Jorge en el DB.

Ahora una solución valida:

- “salt and hash”,

usuario	salt	clave
alicia@protonmail.com	$s \xleftarrow{\$} \{0, 1\}^\ell$	$H(s \text{“mafalda”})$
mateo@protonmail.com	$s' \xleftarrow{\$} \{0, 1\}^\ell$	$H(s' \text{“dragonball”})$
$\vdots$	$\vdots$	$\vdots$

Al registrar cada usuario, una nueva cadena aleatoria (salt) $s \xleftarrow{\$} \{0, 1\}^\ell$ es generada. El hash es hecho sobre “ $s||\text{clave}$ ”, y el resultado es guardado junto al salt. De esta manera el login puede ser verificado, pero cada usuario recibe una función de hash “diferente”. Es improbable que el adversario haya precomputado $H(s||x)$, para el s específico del usuario, volviendo el ataque mas improbable (aunque x tenga poca entropía, $s||x$ tiene mucha).

Nota: esta es una estrategia “folclórica”. No hemos demostrado seguridad, en cuanto no tenemos una definición de seguridad. Formalizaciones y demostraciones sobre este escenario son recientes (Farshim & Tessaro [FT21], EUROCRYPT 2021).

- Aun mejor: “salt and hash”, donde el hash es una función diseñada para guardar contraseñas: Argon2 (IETF RFC9106).

Comentario 34. *No hay salt and hash bueno lo suficiente para protegerte del usar una clave previsible como “dragonball”. De otra parte, las soluciones de password managing pueden ser complicadas para el usuario común. Por esa razón, soluciones “multi-factor” son recomendables.*

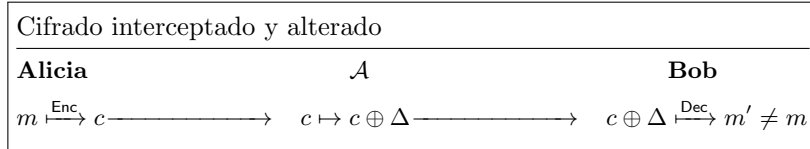
⁶<https://www.openwall.com/john/>

Clase 11: adversarios activos e integridad de datos

Fernando Virdia, versión: 0.1, mayo 2025

6 Integridad de mensajes

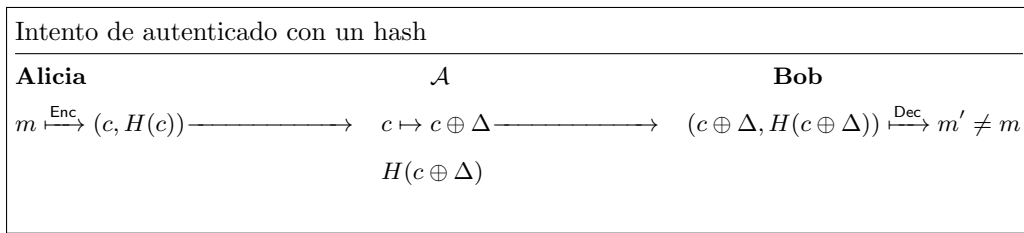
Hasta ahora pudimos garantizar la seguridad si $\mathcal{A}$ se limita a observar cifrados en transito. Ahora consideren el siguiente escenario



En este caso, un SKES no nos brinda garantías.

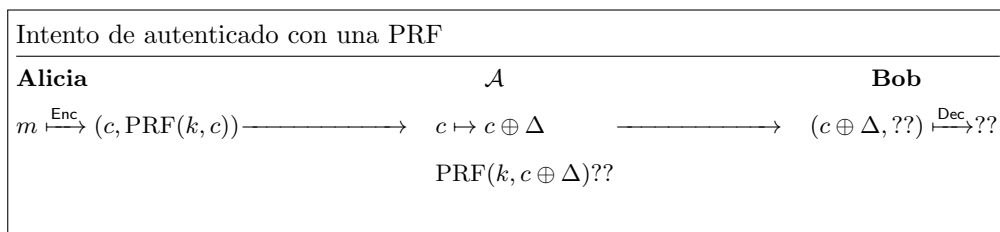
PREGUNTA: Alguien conoce algún método para chequear la integridad de datos?

Idea: Usar un checksum/hash.

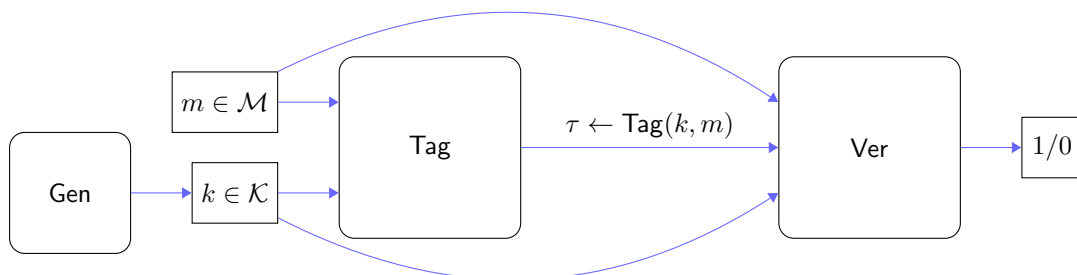


No funciona (a lo sumo, hashing ofrece “integridad sin adversarios”, aunque un código de corrección de errores es una mejor solución a ese problema). Ideas?

Idea: Y si... usamos una PRF?



Para saber si funciona, necesitamos una definición de seguridad. Que sintaxis queremos? Sirve para verificar la autenticidad de un “mensaje” (que puede ser un cifrado).



Definición 19 (MAC). Sean $\mathcal{M}, \mathcal{T}$ conjuntos, $\mathcal{K}$ un conjunto finito. Decimos que $\Pi = (\text{Gen}, \text{Tag}, \text{Ver})$ es un código de autenticación de mensajes (MAC, “message authentication code”) con espacio de llave $\mathcal{K}$, de mensajes $\mathcal{M}$, y de etiquetas (tags) $\mathcal{T}$, si

- Gen es un algoritmo aleatorio $\text{Gen}: \emptyset \rightarrow \mathcal{K}$,
- Tag es un algoritmo (posiblemente aleatorio) $\text{Tag}: \mathcal{K} \times \mathcal{M} \rightarrow \mathcal{T}$,
- Ver es un algoritmo (posiblemente aleatorio) $\text{Ver}: \mathcal{K} \times \mathcal{M} \times \mathcal{T} \rightarrow \{0, 1\}$,

donde

$$\forall k \in \mathcal{K}, m \in \mathcal{M}, \quad \Pr[\text{Ver}(k, m, \text{Tag}(k, m)) = 1] = 1.$$

Comentario 35. Para proteger comunicaciones, va a ser necesario transmitir un tag τ junto a los datos transmitidos. Por eso, idealmente queremos que τ sea lo mas corto posible. Posiblemente $|\tau|$ seria independiente de $|m|$.

Comentario 36. Un sistema con Tag determinista tiene la ventaja de que Ver es muy simple:

$$\text{Ver}(k, m, \tau) := \text{return } \llbracket \text{Tag}(k, m) = \tau \rrbracket$$

Dada la sintaxis de MACs, cualquier $\Pi = (\text{Gen}, \text{Tag}, \text{Ver})$ es un MAC si $\text{Ver}(k, m, \tau) := 1$. Necesitamos definir una noción de seguridad para que “ $1 \leftarrow \text{Ver}(k, m, \tau)$ ” tenga un significado.

Definición 20 (MAC security). Sea $\Pi = (\text{Gen}, \text{Tag}, \text{Ver})$ un MAC con espacios $\mathcal{K}, \mathcal{M}, \mathcal{T}$.

$\text{Exp}^{\text{MAC}}(\mathcal{A})$	$T(m)$
1: $k \leftarrow \text{Gen}()$	1: $\tau \leftarrow \text{Tag}(k, m)$
2: $Q \leftarrow \{\}$	2: $Q \leftarrow Q \cup \{(m, \tau)\}$
3: $(m^*, \tau^*) \leftarrow \mathcal{A}^T()$	3: return τ
4: $b \leftarrow \llbracket \text{Ver}(k, m^*, \tau^*) = 1 \wedge (m^*, \tau^*) \notin Q \rrbracket$	
5: return b	

Decimos que Π es (ε, t, q) -fuertemente infalsificable en un ataque de mensajes elegidos (SUF-CMA, “strongly unforgeable under chosen-message attacks”), si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ y usa $\leq q$ queries a $T(\cdot)$, tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{A}) := \Pr[\text{Exp}^{\text{MAC}}(\mathcal{A}) \Rightarrow 1] \leq \varepsilon.$$

Comentario 37. Hay cosas que se pueden cuestionar en esta definición.

- $\mathcal{A}$ gana si genera un cualquier nuevo tag, inclusive sobre un mensaje que ha ya sido autenticado. Que problema hay en generar una nueva etiqueta para algo que sabemos ser autentico?
- Al $\mathcal{A}$ le permitimos generar tags. Por que también no verificarlos? ‘Podemos notar que para Π con $\text{Ver}(k, m, \tau) := \llbracket \text{Tag}(k, m) = \tau \rrbracket$, $T(\cdot)$ puede ser usado también para verificar. Usaremos esta definición y tipo de MAC para simplificar la exposición. Mas allá de la estructura de Ver , se puede demostrar que otorgarle oráculos de verificación al adversario no lo ayuda de manera significativa [BS23, § 6.2].
- Obtener integridad de comunicaciones no necesariamente excluye todo ataque por parte de adversarios activos. Por ejemplo, la noción de seguridad MAC no cubre ataques donde $\mathcal{A}$ produce “replays” de mensajes previamente autenticados. Estas no son falsificaciones del MAC, pero de cualquier manera pueden si ser peligrosas para el usuario. Piensen en el replay de una orden de compra.

Ahora si podemos demostrar que usar una PRF nos permite autenticar datos.

Lemma 11 (PRF $\Rightarrow$ fixed-length MAC). Sea $F: \mathcal{K} \times \mathcal{D} \rightarrow \mathcal{R}$ una (ε, t, q) -PRF. Sea $\Pi = (\text{Gen}, \text{Tag}, \text{Ver})$ un MAC con espacio de claves $\mathcal{K}$, de mensajes $\mathcal{M} = \mathcal{D}$, y de tags $\mathcal{T} = \mathcal{R}$ donde

$\text{Gen}()$	$\text{Tag}(k, m)$	$\text{Ver}(k, m, \tau)$
1: $k \xleftarrow{\$} \mathcal{K}$	1: $\tau \leftarrow F(k, m)$	1: $\tau' \leftarrow F(k, m)$
2: return k	2: return τ	2: return $\llbracket \tau' = \tau \rrbracket$

Π ofrece (ε', t', q') -seguridad MAC con $\varepsilon' = \varepsilon + \frac{1}{|\mathcal{R}|}$, $t' \approx t$, $q' = q - 1$.

Demostración.

Construimos un adversario $\mathcal{B}$ que juega Exp^{PRF} .

$\mathcal{B}^{\mathcal{O}}()$	$T(m)$
1: $(m^*, \tau^*) \leftarrow \mathcal{A}^T()$	1: $\tau \leftarrow \mathcal{O}(m)$
2: if $\mathcal{O}(m^*) = \tau^* \wedge (m^*, \tau^*) \notin Q$:	2: return τ
3: return 0 (“PRF”)	
4: else :	
5: return 1 (“random”)	

Consideremos $\text{Exp}^{\text{PRF}}(\mathcal{B}, 1)$, donde $\mathcal{O}$ es una función R realmente aleatoria.

$$\begin{aligned}
\Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 0] &= \Pr[R(m^*) = \tau^* \wedge (m^*, \tau^*) \notin Q] \\
&= \Pr[R(m^*) = \tau^* \mid (m^*, \tau^*) \notin Q] \cdot \Pr[(m^*, \tau^*) \notin Q] \\
&\leq \Pr[R(m^*) = \tau^* \mid (m^*, \tau^*) \notin Q]
\end{aligned}$$

Al ser una función, R tiene una sola imagen de m^* . Dado que estamos condicionando sobre $(m^*, \tau^*) \notin Q$, estamos suponiendo que $\mathcal{A}$ nunca ha evaluado $T(m^*) = R(m^*) = \tau^*$, como para poderlo retornar. Por ende, tuvo que adivinar el valor de $\tau^* \in \mathcal{R}$. Dado que R es una función aleatoria, la probabilidad de que $\mathcal{A}$ adivine correctamente es $\Pr[R(m^*) = \tau^* \mid (m^*, \tau^*) \notin Q] = \frac{1}{|\mathcal{R}|}$.

Ahora consideramos $\text{Exp}^{\text{PRF}}(\mathcal{B}, 0)$. Aquí $\mathcal{O} = F(k, \cdot)$, una PRF. Por construcción, el entorno de $\mathcal{A}$ dentro de $\mathcal{B}$ en $\text{Exp}^{\text{PRF}}(\mathcal{B}, 0)$ es idéntico al de $\mathcal{A}$ en Exp^{MAC} , cuando el algoritmo de etiquetado evalúa $\text{Tag}(k, \cdot) = F(k, \cdot)$. Sigue que

$$\begin{aligned}
\Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 0] &= \Pr[F(k, m^*) = \tau^* \wedge (m^*, \tau^*) \notin Q] \\
&= \Pr[\text{Ver}(k, m^*, \tau^*) = 1 \wedge (m^*, \tau^*) \notin Q] \\
&= \Pr[\text{Exp}^{\text{MAC}}(\mathcal{A}) \Rightarrow 1].
\end{aligned}$$

Usando la desigualdad triangular,

$$\begin{aligned}
\mathcal{A}(\text{Exp}^{\text{MAC}}, \mathcal{A}) &= \Pr[\text{Exp}^{\text{MAC}}(\mathcal{A}) \Rightarrow 1] = \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 0] \\
&= \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 0] - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 0] + \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 0] \right| \\
&\leq \left| \left(\lambda - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 0) \Rightarrow 1] \right) - \left(\lambda - \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 1] \right) \right| + \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 0] \right| \\
&= \text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B}) + \left| \Pr[\text{Exp}^{\text{PRF}}(\mathcal{B}, 1) \Rightarrow 0] \right| \\
&\leq \text{Adv}(\text{Exp}^{\text{PRF}}, \mathcal{B}) + \frac{1}{|\mathcal{R}|}.
\end{aligned}$$

Dado que $\mathcal{A}$ y $\mathcal{B}$ corren en tiempos parecidos, y que $\mathcal{B}$ hace una query mas a $\mathcal{O}$ que $\mathcal{A}$ a T , Π es (ε', t', q') -MAC con $\varepsilon' = \varepsilon + \frac{1}{|\mathcal{R}|}$, $t' \approx t$, $q' + 1 = q$. $\square$

Comentario 38. Un MAC con espacios de mensajes $\{0, 1\}^d$ con $d < \infty$ (“fixed-length”), puede ser limitante al tratar de proteger la integridad de mensajes arbitrariamente largos. Hay varias maneras de extender la amplitud de los mensajes. Vamos a ver una técnica que utiliza funciones de hash.

Lemma 12 (fixed-length MAC + CR hash $\Rightarrow$ arbitrary-length MAC, hash-then-MAC). Sea $\Pi = (\text{Gen}, \text{Tag}, \text{Ver})$ un fixed-length (ε, t, q) -MAC con $\mathcal{M} = \{0, 1\}^\ell$, y $H: \{0, 1\}^* \rightarrow \{0, 1\}^\ell$ un CR-hash. Definimos un MAC $\Pi' = (\text{Gen}', \text{Tag}', \text{Ver}')$,

$\text{Gen}'()$	$\text{Tag}'(k, m)$	$\text{Ver}'(k, m, \tau)$
1: $k \xleftarrow{\$} \text{Gen}$	1: $h \leftarrow H(m)$	1: $h \leftarrow H(m)$
2: return k	2: $\tau \leftarrow \text{Tag}(k, h)$	2: $b \leftarrow \text{Ver}(k, h, \tau)$
	3: return τ	3: return b

Π' ofrece (ε', t', q') -seguridad MAC con $t' \approx t$, $q' = q$, $\varepsilon' \leq \varepsilon + \Pr[\text{coll}]$, donde coll es la probabilidad que un adversario contra el MAC Π produzca una colisión en H como parte de su funcionamiento.

Demostración. Usamos un adversario $\mathcal{A}$ contra Π' para definir un adversario $\mathcal{B}$ contra Π .

$\mathcal{B}^T$	$\mathcal{O}(m)$
1: $Q \leftarrow \{\}, S \leftarrow \{\}$	1: $h \leftarrow H(m)$
2: $(m^*, \tau^*) \leftarrow \mathcal{A}^\mathcal{O}()$	2: $\tau \leftarrow T(h)$
3: $h^* \leftarrow H(m^*)$	3: $Q \leftarrow Q \cup \{(h, \tau)\}$
4: if $(h^*, \tau^*) \in Q$:	4: $S \leftarrow S \cup \{(m, \tau)\}$
5: return $(\perp, \perp)$	5: return τ
6: else return (h^*, τ^*)	

NB: El conjunto S en $\mathcal{O}(m)$ simplifica el análisis, pero no es requerido por la implementación de $\mathcal{B}$.

La idea de esta demostración es que si $\mathcal{A}$ falsifica correctamente $((m^*, \tau^*) \notin S)$ una etiqueta y $(h^*, \tau^*) \notin Q$, entonces $\mathcal{B}$ también falsifica correctamente. Por otro lado, si $\mathcal{A}$ falsifica correctamente $((m^*, \tau^*) \notin S)$ pero $(h^*, \tau^*) \in Q$, entonces existe algún mensaje $\bar{m} \neq m^*$ con $(\bar{m}, \tau^*) \in S$ tal que $h^* = H(m^*) = H(\bar{m})$, por lo que $\mathcal{A}$ produjo una colisión $(\bar{m}, m^*)$ de H .

Formalmente, empezamos expandiendo la ventaja de $\mathcal{A}$ contra Π' en términos de Π .

$$\begin{aligned}
\text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{A}) &= \Pr[\text{Exp}^{\text{MAC}}(\mathcal{A}) \Rightarrow 1] = \Pr[\text{Ver}'(k, m^*, \tau^*) = 1 \wedge (m^*, \tau^*) \notin S] \\
&= \Pr[\text{Ver}(k, H(m^*), \tau^*) = 1 \wedge (m^*, \tau^*) \notin S] \\
&= \Pr[\text{Ver}(k, h^*, \tau^*) = 1 \wedge (m^*, \tau^*) \notin S] \\
&= \Pr[\text{Ver}(k, h^*, \tau^*) = 1 \wedge (m^*, \tau^*) \notin S \wedge (h^*, \tau^*) \notin Q] \\
&\quad + \Pr[\text{Ver}(k, h^*, \tau^*) = 1 \wedge (m^*, \tau^*) \notin S \wedge (h^*, \tau^*) \in Q] \\
&\leq \Pr[\text{Ver}(k, h^*, \tau^*) = 1 \wedge (h^*, \tau^*) \notin Q] + \Pr[(m^*, \tau^*) \notin S \wedge (h^*, \tau^*) \in Q] \\
&= \Pr[\text{Exp}^{\text{MAC}}(\mathcal{B}) \Rightarrow 1] + \Pr[(m^*, \tau^*) \notin S \wedge (h^*, \tau^*) \in Q] \\
&= \text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{B}) + \Pr[(m^*, \tau^*) \notin S \wedge (h^*, \tau^*) \in Q]
\end{aligned}$$

Por suposición, los conjuntos S y Q están formados por mensajes m_i y hashes $h_i = H(m_i)$, donde los m_i fueron generados por $\mathcal{A}$, así como lo fue m^* . Sabemos que $h^* = H(m^*)$. Si $(m^*, \tau^*) \notin S$ pero $(h^*, \tau^*) \in Q = \{(H(m_i), \tau_i)\}_i$, entonces existe un $(m_i, \tau_i) \in S$ tal que $\tau_i = \tau^*$ y $H(m_i) = h^* = H(m^*)$ donde $m^* \neq m_i$.

Lo que quiere decir que $\mathcal{A}$ genera una colisión (m_i, m^*) de H , lo que implica que

$$\Pr[(m^*, \tau^*) \notin S \wedge (h^*, \tau^*) \in Q] \leq \Pr[\text{coll}].$$

Juntando las tres desigualdades,

$$\begin{aligned}
\text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{A}) &\leq \text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{B}) + \Pr[(m^*, \tau^*) \notin S \wedge (h^*, \tau^*) \in Q] \\
&\leq \text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{B}) + \Pr[\text{coll}].
\end{aligned}$$

Por inspección de $\mathcal{A}$ y $\mathcal{B}$, $t' \approx t$, $q' = q$ y $\varepsilon' \leq \varepsilon + \Pr[\text{coll}]$. □

Comentario 39. *Requerir un MAC y un hash contruidos de manera independiente puede ser costoso en hardware. Construcciones tipo HMAC se especializan en construir hash-then-MAC compactos a partir de funciones de hash de tipo Merkle-Damgard.*

Clase 12: confidencialidad e integridad, el cifrado autenticado

Fernando Virdia, versión: 0.1, mayo 2025

7 Cifrado Autenticado

Resumen: hemos conseguido

- SKES para mensajes múltiples y arbitrariamente largos
- MACs para cadenas de bits arbitrariamente largas

Que nos falta: integrar ambos mecanismos.

Vamos a protegernos de ambos escenarios limitando la posibilidad de $\mathcal{A}$ de modificar cifrados.

Definición 21 (INT-CTXT). Sea $\perp$ un símbolo que indica rechazo. Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un SKES con $\perp \in \mathcal{M}$, tal que Dec puede retornar $\perp$ si recibe un cifrado no válido.

$\text{Exp}^{\text{CTXT}}(\mathcal{A})$	$E(m)$
1: $k \xleftarrow{\$} \text{Gen}()$	1: $c \leftarrow \text{Enc}(k, m)$
2: $Q_c \leftarrow \{\}$	2: $Q_c \leftarrow Q_c \cup \{c\}$
3: $c^* \leftarrow \mathcal{A}^E()$	3: return c
4: $m^* \leftarrow \text{Dec}(k, c^*)$	
5: $b \leftarrow \llbracket m^* \neq \perp \wedge c^* \notin Q_c \rrbracket$	
6: return b	

Decimos que Π otorga (ε, t, q) -integridad de cifrado (INT-CTXT, “ciphertext integrity”), si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ y hace $\leq q$ queries a $\mathcal{O}$, tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{CTXT}}, \mathcal{A}) := \Pr[\text{Exp}^{\text{CTXT}}(\mathcal{A}) \Rightarrow 1] \leq \varepsilon.$$

Comentario 40. Claramente, PRF-CTR no satisface INT-CTXT, dado que $(c_0, c_1 \oplus \Delta)$ descifra a $m \oplus \Delta$ para cualquier Δ .

Definición 22 (AE). Sea Π un SKES. Decimos que Π otorga cifrados $(\varepsilon_{\text{CPA}}, \varepsilon_{\text{CTXT}}, t, q)$ -autenticados (AE, “authenticated encryption”), si otorga seguridad $(\varepsilon_{\text{CPA}}, t, q)$ -IND-CPA y $(\varepsilon_{\text{CTXT}}, t, q)$ -INT-CTXT.

Comentario 41. Bajo definición de seguridad AE, $\mathcal{A}$ no puede distinguir entre cifrados de mensajes diferentes por IND-CPA, y no puede tampoco generar nuevos cifrados válidos (de la nada, o modificando cifrados que obtuvo a través de un oráculo de cifrado) por INT-CTXT.

Realizar un SKES que otorga seguridad AE es relativamente simple.

Comentario 42. Un SKES que otorga AE, también otorga una noción que no vamos a explorar de inmediato, IND-CCA. Exp^{CCA} es igual a Exp^{CPA} , pero $\mathcal{A}$ tiene acceso a oráculos de cifrado $E(m_0, m_1)$ y de descifrado $D(c)$, donde el segundo puede ser utilizado solamente para descifrar cifrados no retornados por $E(\cdot, \cdot)$. En el ámbito de la criptografía simétrica, $\text{AE} \Rightarrow \text{IND-CCA}$ dado que $\mathcal{A}$ no puede generar cifrados válidos a menos de no obtenerlos de $E(\cdot, \cdot)$. De consecuencia, un adversario IND-CCA contra un SKES AE no puede generar cifrados para pasarle al oráculo de descifrado D que no resulten en un rechazo $D(c) = \perp$.

Definición 23 (EtM). Sean $\Pi = (\text{Gen}^\Pi, \text{Enc}, \text{Dec})$ un SKES, y $\Sigma = (\text{Gen}^\Sigma, \text{Tag}, \text{Ver})$ un MAC. Definimos el SKES $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ “encrypt-then-MAC” (EtM).

$\text{Gen}'()$	$\text{Enc}'(k, m)$	$\text{Dec}'(k, c)$
1: $k_e \leftarrow \text{Gen}^\Pi()$	1: $(k_e, k_m) \leftarrow k$	1: $(k_e, k_m) \leftarrow k$
2: $k_m \leftarrow \text{Gen}^\Sigma()$	2: $c_e \leftarrow \text{Enc}(k_e, m)$	2: $(c_e, \tau) \leftarrow c$
3: $k \leftarrow (k_e, k_m)$	3: $\tau \leftarrow \text{Tag}(k_m, c_e)$	3: $b \leftarrow \text{Ver}(k_m, c_e, \tau)$
4: return k	4: $c \leftarrow (c_e, \tau)$	4: $m \leftarrow \text{Dec}(k_e, c_e)$
	5: return c	5: if $b = 1$:
		6: return m
		7: else return $\perp$

Lemma 13. Sean Π , Σ , Π' como en Definition 23. Supongamos que Π otorga $(\varepsilon_{\text{CPA}}, t, q)$ -IND-CPA y Σ otorga $(\varepsilon_{\text{MAC}}, t, q)$ -MAC. Entonces Π' otorga $(\varepsilon'_{\text{CPA}}, \varepsilon'_{\text{CTXT}}, t', q')$ -AE con $\varepsilon'_{\text{CPA}} = \varepsilon_{\text{CPA}}$, $\varepsilon'_{\text{CTXT}} = \varepsilon_{\text{MAC}}$, $t' \approx t$, $q' = q$.

Demostración. Por definición de AE, necesitamos demostrar que

- Π' otorga $(\varepsilon'_{\text{CPA}}, t', q')$ -IND-CPA,
- Π' otorga $(\varepsilon'_{\text{CTXT}}, t', q')$ -INT-CTXT.

IND-CPA: Dado un adversario $\mathcal{A}$ contra Π' que juega Exp^{CPA} , construimos $\mathcal{B}$ contra Π que juega Exp^{CPA} ,

$\mathcal{B}^E()$	$\mathcal{O}(m_0, m_1)$
1 : $k_m \leftarrow \text{Gen}^\Sigma()$	1 : $c_e \leftarrow E(m_0, m_1)$
2 : $b' \leftarrow \mathcal{A}^\mathcal{O}()$	2 : $\tau \leftarrow \text{Tag}(k_m, c_e)$
3 : return b'	3 : $c \leftarrow (c_e, \tau)$
	4 : return c

Del punto de vista de $\mathcal{A}^\mathcal{O}$, el ambiente dentro de $\mathcal{B}$ es idéntico al de $\text{Exp}^{\text{CPA}}(\mathcal{A}, b)$. Dado que $\mathcal{B}$ retorna el output de $\mathcal{A}$,

$$\Pr[\text{Exp}^{\text{CPA}}(\mathcal{B}, b) \Rightarrow 1] = \Pr[\text{Exp}^{\text{CPA}}(\mathcal{A}, b) \Rightarrow 1], \quad \forall b \in \{0, 1\}.$$

De consecuencia, $\text{Adv}(\text{Exp}^{\text{CPA}}, \mathcal{B}) = \text{Adv}(\text{Exp}^{\text{CPA}}, \mathcal{A})$, $t' \approx t$ y $q' = q$, y Π' otorga $(\varepsilon_{\text{CPA}}, \approx t, q)$ -IND-CPA.

INT-CTXT: Ahora supongamos que un adversario $\mathcal{A}$ contra Exp^{CTXT} , y construyamos $\mathcal{B}$ contra Exp^{MAC} .

$\mathcal{B}^T()$	$\mathcal{O}(m)$
1 : $k_e \leftarrow \text{Gen}^\Pi()$	1 : $c_e \leftarrow \text{Enc}(k_e, m)$
2 : $c^* \leftarrow \mathcal{A}^\mathcal{O}()$	2 : $\tau \leftarrow T(c_e)$
3 : $(c_e^*, \tau^*) \leftarrow c^*$	3 : $c \leftarrow (c_e, \tau)$
4 : // c_e^* es el mensaje “tagueado”	4 : return c
5 : return (c_e^*, τ^*)	

Fundamentalmente, c^* consiste de un “mensaje” arbitrario c_e^* y de una etiqueta τ^* . Por inspección del algoritmo Dec' del cifrado EtM, notamos que para que este retorne un mensaje $m^* \neq \perp$, es necesario que $\text{Ver}(k_m, c_e^*, \tau^*) = 1$. Si $\mathcal{A}$ logra producir una tupla c^* nueva y valida, efectivamente ha falsificado una etiqueta del MAC.

Formalmente, supongamos que $\mathcal{A}$ se encuentra jugando Exp^{CTXT} contra el cifrado EtM con clave de cifrado k_e . Sigue que por definición

$$\text{Adv}(\text{Exp}^{\text{CTXT}}, \mathcal{A}) = \Pr[\text{Dec}'(k_e, c^*) \neq \perp \wedge c^* \notin Q_c]$$

Sea c^* un cifrado EtM con componentes $c^* = (c_e^*, \tau^*)$. Por inspección del algoritmo Dec' del cifrado EtM, notamos que para que este retorne un mensaje $m^* \neq \perp$, es necesario que $\text{Ver}(k_m, c_e^*, \tau^*) = 1$, por lo que

$$\Pr[\text{Dec}'(k_e, c^*) \neq \perp \wedge c^* \notin Q_c] \leq \Pr[\text{Ver}(k_m, c_e^*, \tau^*) = 1 \wedge c^* \notin Q_c]$$

Ahora notemos que el entorno de $\mathcal{A}$ dentro de $\mathcal{B}^T$ cuando este ultimo juega el experimento Exp^{MAC} es esactamente identico al entorno de $\mathcal{A}$ dentro de Exp^{CTXT} . En particular, cada vez que $\mathcal{A}$ le pasa un mensaje m a $\mathcal{O}$, un cifrado $c_e \leftarrow \text{Enc}(k_e, m)$ viene pasado como input al oraculo de etiquetado T en Exp^{MAC} , el cual retorna una etiqueta τ sobre c_e . Esto quiere decir que $c = (c_e, \tau) \in Q_c$ dentro de $\mathcal{B} \iff (c_e, \tau) \in Q$ dentro de $\text{Exp}^{\text{MAC}}(\mathcal{B})$. Por lo tanto,

$$\begin{aligned} & \Pr[\text{Ver}(k_m, c_e^*, \tau^*) = 1 \wedge c^* \notin Q_c \text{ dentro de } \text{Dec}' \text{ dentro de } \text{Exp}^{\text{CTXT}}] \\ &= \Pr[\text{Ver}(k_m, c_e^*, \tau^*) = 1 \wedge (c_e^*, \tau^*) \notin Q \text{ dentro de } \mathcal{B} \text{ dentro de } \text{Exp}^{\text{MAC}}] \\ &= \Pr[\text{Exp}^{\text{MAC}}(\mathcal{B}) \Rightarrow 1] = \text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{B}). \end{aligned}$$

Juntando las desigualdades, obtenemos que $\text{Adv}(\text{Exp}^{\text{CTXT}}, \mathcal{A}) \leq \text{Adv}(\text{Exp}^{\text{MAC}}, \mathcal{B})$. Por lo tanto, tenemos $\varepsilon'_{\text{CTXT}} \leq \varepsilon_{\text{MAC}}$, y por construcción de $\mathcal{B}$, $t' \approx t$, $q' = q$, y Π' otorga $(\varepsilon_{\text{MAC}}, \approx t, q)$ -INT-CTXT.

Y por lo tanto, Π' otorga $(\varepsilon_{\text{CPA}}, \varepsilon_{\text{MAC}}, \approx t, q)$ -AE. $\square$

Comentario 43. Si definiéramos la seguridad MAC de manera que a $\mathcal{A}$ le fuera permitido generar nuevos tags τ' a partir de un mensaje ya etiquetado por $T(\cdot)$ (o sea, el MAC no sería “fuertemente” infalsificable), el resultante cifrado EtM no otorgaría AE. Esto es porque $\mathcal{A}$ podría ver un válido cifrado $c = (c_e, \tau)$ y generar un nuevo cifrado $c' = (c_e, \tau')$. Esto podría ser un problema en el caso específico de EtM, pero la definición de IND-CTXT es general, y no tendría que solamente funcionar para EtM.

Ejemplo 3. Combinar esquemas AE puede no otorgar trivialmente AE. Considérese el esquema natural obtenido concatenando cifrados: $\text{Enc}'(k, m_1 || m_2) := \text{Enc}(k, m_1) || \text{Enc}(k, m_2)$, donde Enc es parte de un SKES que otorga AE, y cada componente del cifrado $c = (c_1, c_2)$ es descifrado individualmente. ‘Este sistema no otorga AE, por que?’

Dado un cifrado (c_1, c_2) válido, el cifrado (c_2, c_1) es también válido. Esto quiere decir que $\mathcal{A}$ puede generar nuevos cifrados válidos, invalidando INT-CTXT.

Comentario 44. AE nos permite de detectar si un cifrado es autentico. Pero no impide que $\mathcal{A}$ reordene, elimine o repita cifrados en transito desde Alicia a Bob. Sin embargo, podemos intentar detectar este tipo de ataque en el nivel aplicativo, definiendo números de secuencia. Esto generalmente resulta en consideraciones dependientes de la aplicación, y que pueden encontrarse al menos parcialmente fuera del dominio de la criptografía.

En algunas aplicaciones a nivel de red, puede ser necesario transmitir datos cifrados junto a meta-datos (e.g., “packet headers”) públicos. Por ejemplo, los meta-datos podrían ser útiles para “route” los paquetes de datos. En esta situación, una primitiva conocida como “authenticated encryption with associated data” (AEAD) puede ser útil. Esta garantiza confidencialidad de datos, e integridad del cifrado y de los meta-datos. Los dos AEAD más famosos son AES-GCM y Chacha20-Poly1305.

Definición 24 (Un AEAD de ejemplo). Sean $\Pi = (\text{Gen}^\Pi, \text{Enc}, \text{Dec})$ un SKES, $\Sigma = (\text{Gen}^\Sigma, \text{Tag}, \text{Ver})$ un arbitrary-length MAC. Definimos un AEAD

$\text{Gen}'()$	$\text{Enc}'(k, d, m)$	$\text{Dec}'(k, d, c)$
1: $k_e \leftarrow \text{Gen}^\Pi()$	1: $(k_e, k_m) \leftarrow k$	1: $(k_e, k_m) \leftarrow k$
2: $k_m \leftarrow \text{Gen}^\Sigma()$	2: $c_e \leftarrow \text{Enc}(k_e, m)$	2: $(c_e, \tau) \leftarrow c$
3: $k \leftarrow (k_e, k_m)$	3: $\ell \leftarrow d $	3: $\ell \leftarrow d $
4: return k	4: $\tau \leftarrow \text{Tag}(k_m, \ell d c_e)$	4: $b \leftarrow \text{Ver}(k_m, \ell d c_e, \tau)$
	5: return (c_e, τ)	5: $m \leftarrow \text{Dec}(k_e, c_e)$
		6: if $b = 1$:
		7: return m
		8: else return $\perp$

Comentario 45. No proponemos una definición de seguridad AEAD. Invitamos el lector a consultar [BS23, § 9.5].

Con esto terminamos la componente simétrica. Hemos aprendido a cifrar y transmitir cifrados de manera segura entre dos partes Alicia y Bob que comparten una misma clave secreta k . Pero como pueden llegar Alicia y Bob a compartir tal clave? Claro, podrían compartirla de antemano de persona, de manera de estar seguras que ningún adversario $\mathcal{A}$ pueda verla, pero esto no es conveniente. Todos los días navegamos a paginas web, posiblemente por la primera vez, y hacemos pagos online con diferentes entidades. Pre-compartir claves con todas sería prácticamente imposible.

En la segunda parte del curso vamos a ver como transmitir mensajes de manera segura a entidades con las cuales no nos hemos nunca comunicado.

Clase 13: preliminares matemáticos II

Fernando Virdia, versión: 0.1, mayo 2025

8 Grupos, y Suposiciones de Dificultad

A la base de gran parte de la PKC hay objetos de naturaleza algebraica: grupos, anillos, cuerpos. Nosotros vamos a intentar cubrir la menor cantidad posible, quedando cerca de lo utilizado en la practica.

Definición 25 (grupo abeliano). Sea G un conjunto, $\times$ una operación binaria $\times: G \times G \rightarrow G$. Decimos que $\mathbb{G} = (G, \times)$ es un grupo abeliano si

1. $\times$ es asociativa: $(a \times b) \times c = a \times (b \times c)$, $\forall a, b, c \in G$.
2. Existe un elemento neutro $e \in G$ tal que $a \times e = e \times a = a$, $\forall a \in G$.
3. Cada elemento a tiene un inverso a^{-1} , tal que $a \times a^{-1} = a^{-1} \times a = e$, $\forall a \in G$.
4. $\times$ es conmutativa: $a \times b = b \times a$, $\forall a, b \in G$.

Si $|G| < \infty$, decimos que $\mathbb{G}$ es finito, y que $|G|$ es el orden de $\mathbb{G}$. A menudo abusamos notación y escribimos $a \in \mathbb{G}$ en lugar de $a \in G$.

Ejemplo 4. • $(\mathbb{Z}, +)$ es un grupo abeliano

- $(\mathbb{Z}, \cdot)$ **no es** un grupo abeliano. $(\mathbb{Q}, \cdot)$ tampoco. $(\mathbb{Q} \setminus \{0\}, \cdot)$ lo es. $(\mathbb{R} \setminus \{0\}, \cdot)$ también.
- Sea $\mathbb{Z}_n$ el conjunto de los enteros “modulo n ”. $(\mathbb{Z}_n, +)$ es un grupo abeliano.

Comentario 46. En un grupo podemos “exponenciar”: si $a \in \mathbb{G}$, $a^n := a \times a \times \cdots \times a$ (n veces). Las comunes reglas valen:

- $a^n \times a^m = a^{n+m}$,
- $(a^n)^m = a^{nm} = a^{mn} = (a^m)^n$,
- $a^1 \cdot a^{-1} = a^0 = e$.

En el caso de un grupo donde usamos “+”, como $(\mathbb{Z}, +)$, el exponente es un factor: $a + \cdots + a$ (n veces) $= n \times a$.

Definición 26 (subgrupo). Sea $\mathbb{G} = (G, \times)$ un grupo, y sea $S \subset G$ un subconjunto de G . Si $\mathbb{S} = (S, \times)$ forma un grupo, decimos que $\mathbb{S}$ es un subgrupo de $\mathbb{G}$.

Ejemplo 5. Sea $\mathbb{G} = (\mathbb{Z}, +)$. $\mathbb{S} = (2\mathbb{Z}, +) = (\{2n \mid n \in \mathbb{Z}\}, +)$ es un subgrupo de $\mathbb{G}$.

Definición 27 (grupo cíclico). Un grupo $\mathbb{G} = (G, \times)$ se dice cíclico si existe $g \in G$ tal que el grupo $\langle g \rangle$ “generado” g ,

$$\langle g \rangle := (\{g^n \mid n \in \mathbb{Z}\}, \times),$$

es $\mathbb{G}$, o sea $\mathbb{G} = \langle g \rangle$.

Ejemplo 6. $(\mathbb{Z}, +)$ es cíclico, y generado por ± 1 .

Lemma 14. Dado un grupo $\mathbb{G} = (G, \times)$ de orden primo $|G| = p$, $\mathbb{G}$ es cíclico y generado por cualquier elemento no neutro.

Ejemplo 7. $(\mathbb{Z}_p, +)$ es cíclico. Cualquier $1 \leq g < p$ lo genera. Por ejemplo, sea $p = 3$.

1 es un generador		2 es un generador	
1	mod 3 = 1	2	mod 3 = 2
1 + 1	mod 3 = 2	2 + 2	mod 3 = 4 mod 3 = 1
1 + 1 + 1	mod 3 = 3	2 + 2 + 2	mod 3 = 6 mod 3 = 0
1 + 1 + 1 + 1	mod 3 = 4 mod 3 = 1	2 + 2 + 2 + 2	mod 3 = 8 mod 3 = 2

Aquí podríamos adentrarnos en mas detalles sobre los grupos usados en criptografía, pero se vuelve muy matemático. En su lugar, vamos a definir tres problemas matemáticos relacionados, y un postulado. En criptografía, usamos familias de grupos que satisfacen el postulado.

Definición 28 (DLOG, informal). Sea $\mathbb{G}$ un grupo cíclico. Dados $g, h \in \mathbb{G}$, el problema del logaritmo discreto (DLOG, “discrete logarithm”) requiere recuperar $x \in \mathbb{Z}$ tal que $g^x = h$, o sea “ $x = \log_g h$ ”.

Comentario 47. DLOG es fácil de resolver en $(\mathbb{Z}, +)$ y $(\mathbb{R} \setminus \{0\}, \times)$:

- $(\mathbb{Z}, +)$: recordando que “ g^x ” es $x \cdot g$, $x \cdot g = h \iff x = h/g$.
- $(\mathbb{R} \setminus \{0\}, \times)$: $g^x = h \iff x = \log_g h$, donde $\log$ es el logaritmo “comun”.

Definición 29 (CDH, informal). Sea $\mathbb{G}$ un grupo cíclico finito con generador g . Sean $x \xleftarrow{\$} [|\mathbb{G}|]$, $y \xleftarrow{\$} [|\mathbb{G}|]$ muestreados independientemente. El problema computacional de Diffie-Hellman (CDH, “computational Diffie-Hellman”) requiere, dados (g, g^x, g^y) , calcular $g^{xy} = (g^x)^y = (g^y)^x$.

Comentario 48. CDH no puede ser mas difícil de DLOG (o sea, resolver DLOG $\Rightarrow$ resolver CDH). Supongamos que $\mathcal{O}$ es un oráculo que calcula $(g, g^x) \mapsto x$. Dados (g, g^x, g^y) , calculamos $(g^y)^{\mathcal{O}(g, g^x)} = (g^y)^x = g^{xy}$.

Lemma 15. Sea $\mathbb{G} = \langle g \rangle$ un grupo finito. Dada la variable aleatoria $z \sim U([|\mathbb{G}|])$, g^z es una variable aleatoria $g^z \sim U(\mathbb{G})$.

Proof.

$$\forall h \in \mathbb{G}, \quad \Pr_{z \sim U([|\mathbb{G}|])} [g^z = h] = \Pr[z = \log_g h] = \frac{1}{|\mathbb{G}|}.$$

□

Definición 30 (DDH). Sea $\mathbb{G} = \langle g \rangle$ un grupo cíclico finito. Para simplificar notación, supongamos que g es un “parámetro publico”, o sea un elemento fijo y disponible públicamente. Definimos el juego Exp^{DDH} .

$\text{Exp}^{\text{DDH}}(\mathcal{A}, b)$	
1:	$x \xleftarrow{\$} [\mathbb{G}]$
2:	$y \xleftarrow{\$} [\mathbb{G}]$
3:	if $b = 0$
4:	$z \leftarrow x \cdot y$
5:	else
6:	$z \xleftarrow{\$} [\mathbb{G}]$
7:	$b' \leftarrow \mathcal{A}(g^x, g^y, g^z)$
8:	return b'

Decimos que $\mathbb{G}$ otorga seguridad (ε, t) -DDH si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{DDH}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{DDH}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{DDH}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

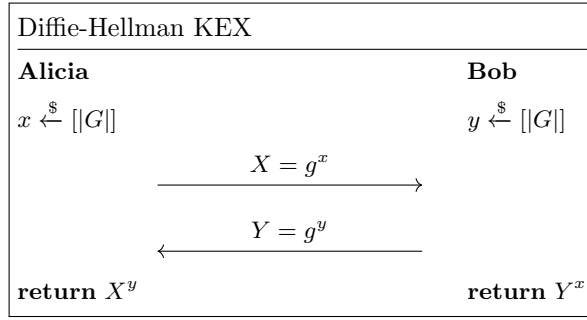
Finalmente introducimos un nuevo postulado:

Postulado 3. Dado $\lambda > 0$, existe un grupo cíclico finito $\mathbb{G}$ que otorga λ -bits de seguridad DDH.

Comentario 49. Por motivos criptoanalíticos, es generalmente deseable que $\mathbb{G}$ tenga orden primo.

Vamos ahora a introducir brevemente el intercambio de claves (KEX, key exchange) de Diffie y Hellman. Lo hacemos por motivos informacionales e históricos, al ser el primer protocolo de PKC considerado seguro. No vamos a intentar demostrar su seguridad en detalle, porque los modelos de seguridad de KEX pueden ser muy complicados.

Supongamos que dos partes Alicia y Bob se ponen de acuerdo publicamente sobre un grupo cíclico finito $\mathbb{G}$ y sobre un generador g , tal que $\mathbb{G} = \langle g \rangle$.

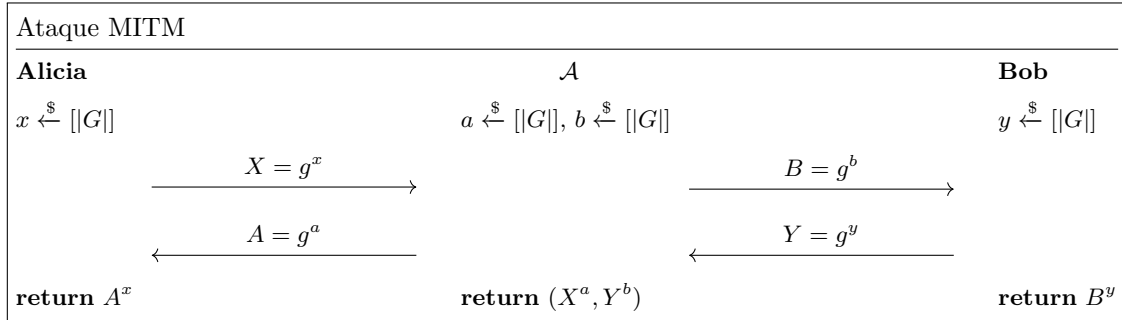


Notamos que $X^y = (g^x)^y = g^{xy} = g^{yx} = (g^y)^x = Y^x$, por lo cual Alicia y Bob retornan el mismo elemento $g^{xy} \in \mathbb{G}$. Este puede ser utilizado para generar una clave simétrica k , de modo de continuar comunicando a través de un SKES seguro, por ejemplo un AEAD. Una opción común es definir una función de hash $H: \mathbb{G} \rightarrow \{0, 1\}^\ell$, y utilizar $H(g^{xy})$ como clave.

Definir la seguridad del KEX puede ser complicado. Supongamos un adversario que se limita a observar pasivamente la comunicación (el “transcript”) (g^x, g^y) . Si CDH es difícil en $\mathbb{G}$, sabemos que calcular g^{xy} es difícil. Es esto suficiente para garantizar que DH es seguro? CDH no excluye que “mitad de los bits” de g^{xy} sean aprendibles. Esto podría teóricamente resultar en vulnerar la siguiente fase de comunicación.

De aquí la idea de usar DDH: si (g^x, g^y, g^{xy}) es indistinguible de (g^x, g^y, g^z) con z aleatorio, $\mathcal{A}$ no podría averiguar si... De otro lado, si $\mathcal{A}$ tuviera una suposición $h \in \mathbb{G}$ para el valor de g^{xy} , aunque DDH sea difícil, podría utilizar la siguiente comunicación con clave $H(g^{xy})$ para determinar si adivino h correctamente.

Notamos también que $\mathcal{A}$ podría no limitarse simplemente a observar tráfico, y también interceptarlo si tiene control sobre la red. Este tipo de ataque se llama machine-in-the-middle (MITM).



En este caso $\mathcal{A}$ podría fingir de ser Bob a Alice, y Alicia a Bob. Por ejemplo, podría interceptar $\text{Enc}(H(g^{ax}), m)$ de Alicia a Bob, descifrarlo, volverlo a cifrar como $\text{Enc}(H(g^{by}), m)$, y finalmente mandarlo a Bob. Alicia y Bob no podrían detectar el ataque.

Este tipo de táctica no funciona si Alice y Bob intercambian claves secretas de persona, y utilizan exclusivamente criptografía simétrica. Defendernos de este tipo de táctica requiere nuevos instrumentos, para construir una “infraestructura de claves públicas” (PKI, public-key infrastructure).

Clase 14: intercambio de claves de Diffie–Hellman

Fernando Virdia, versión: 0.1, mayo 2025

Esta clase se encuentra en forma de diapositivas en <https://apuntes.indcpa.com>.

Clase 15: cifrado de clave publica “one-time”

Fernando Virdia, versión: 0.1, mayo 2025

9 Cifrado de Clave Publica

Como en el caso simétrico, empezamos definiendo nociones de cifrado.

Definición 31 (PKES, cifrado asimétrico o de clave publica). Sean $\mathcal{K}_p$, $\mathcal{K}_s$, $\mathcal{M}$, $\mathcal{C}$ conjuntos. Decimos que $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ forman un cifrado asimétrico o de clave publica (PKES, “public-key encryption scheme”) con espacio de claves publicas $\mathcal{K}_p$, de claves privadas $\mathcal{K}_s$, de mensajes $\mathcal{M}$, y de cifrados $\mathcal{C}$, si

- Gen es un algoritmo aleatorio, $\text{Gen}: \emptyset \rightarrow \mathcal{K}_s \times \mathcal{K}_p$ que retorna una tupla (sk, pk) . Decimos que sk es una clave privada, o secreta, (“secret key”) y que pk es una clave publica (“public key”).
- Enc es un algoritmo (posiblemente aleatorio), $\text{Enc}: \mathcal{K}_p \times \mathcal{M} \rightarrow \mathcal{C}$.
- Dec es un algoritmo determinista $\text{Dec}: \mathcal{K}_s \times \mathcal{C} \rightarrow \mathcal{M}$.

Decimos que Π es δ -correcto si

$$\Pr_{\substack{(\text{sk}, \text{pk}) \sim \text{Gen}(), \\ \$ \text{ en Enc}}} [\text{Dec}(\text{sk}, \text{Enc}(\text{pk}, m; \$)) \neq m] \leq \delta,$$

donde Gen, Enc y Dec son “eficientes” (corren en tiempo $\approx \log_2(|m|)$).

Comentario 50. Al definir SKES, requerimos que fueran δ -correcto con $\delta = 0$. No es así con PKES, porque varias técnicas para construir PKE resulta en sistemas eficientes si $\delta \gtrsim 0$ pero menos eficientes con $\delta = 0$. En el curso vamos a ver solo sistemas con $\delta = 0$.

Comentario 51. En nuestra definición usamos conjuntos $\mathcal{M}$ y $\mathcal{C}$. En practica, es común que al generar claves, implícitamente fijemos $\mathcal{M}_{\text{pk}} \subset \mathcal{M}$ y $\mathcal{C}_{\text{pk}} \subset \mathcal{C}$ donde se encuentran los mensajes y cifrados.

Comentario 52. Diferentemente de SKE, a menudo $\mathcal{K}_s$, $\mathcal{K}_p$, $\mathcal{M}$, $\mathcal{C}$ **no son** conjuntos de cadenas de bits.

Muchas de las ideas desarrolladas para SKE pueden ser adaptadas a PKE. Una diferencia crucial es que $\mathcal{A}$ recibe pk, al ser una clave publica, y esto le permite cifrar mensajes sin requerir un oráculo.

Definición 32 (OT-IND-CPA PKE). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES.

$\text{Exp}^{\text{OT-CPA}}(\mathcal{A}, b)$	$\mathcal{O}(m_0, m_1)$
1: $(\text{sk}, \text{pk}) \leftarrow \text{Gen}()$	1: if disabled = 1 :
2: disabled $\leftarrow 0$	2: return $\perp$
3: $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\text{pk})$	3: disabled $\leftarrow 1$
4: return b'	4: $c^* \leftarrow \text{Enc}(\text{pk}, m_b)$
	5: return c^*

Decimos que Π otorga one-time (ε, t) -indistinguibilidad bajo ataques de mensaje elegido (OT-IND-CPA, “one-time indistinguishability under chosen-plaintext attacks”), si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

Comentario 53. Aunque esta definición le ofrece a $\mathcal{A}$ un solo cifrado para distinguir b , a menudo la preferimos a una definición que ofrezca $q \geq 1$ cifrados. Esto es porque demostrar OT-CPA es mas fácil, y porque en el caso de PKE, $\text{OT-IND-CPA} \Rightarrow \text{IND-CPA}$.

Vamos ahora a definir el PKES de ElGamal. Puede ser visto como una versión no interactiva del KEX de Diffie-Hellman.

Definición 33 (ElGamal PKE). Sea $\mathbb{G} = (G, \times)$ un grupo cíclico generado por g . Sean $\mathcal{K}_s = [|G|]$, $\mathcal{K}_{pk} = \mathcal{M} = \mathcal{C} = G$.

Gen()	Enc(pk = g^x, m)	Dec(sk = $x, c = (c_0, c_1)$)
1: $x \xleftarrow{\$} [G]$	1: $y \xleftarrow{\$} [G]$	1: $m' \leftarrow c_1 \times c_0^{-x}$
2: $sk \leftarrow x$	2: $c_0 \leftarrow g^y$	2: return m'
3: $pk \leftarrow g^x$	3: $c_1 \leftarrow (g^x)^y \times m$	
4: return (sk, pk)	4: $c \leftarrow (c_0, c_1)$	
	5: return c	

Observamos que Π es δ -correcto con $\delta = 0$, dado que

$$m' = c_1 \times c_0^{-x} = (g^x)^y \times m \times (g^y)^{-x} = m \quad \forall x, y \in [|G|].$$

Lemma 16 (ElGamal PKE + (ε, t) -DDH $\Rightarrow$ OT-IND-CPA). Sea $\mathbb{G} = \langle g \rangle$ un grupo cíclico tal que DDH es (ε, t) -difícil. Sea Π el PKES de ElGamal definido sobre $\mathbb{G}$ con generador g . Entonces Π otorga (ε', t') -OT-IND-CPA con $\varepsilon' = 2 \cdot \varepsilon$, $t' \approx t$.

Proof. Supongamos un adversario $\mathcal{A}$ contra $\text{Exp}^{\text{OT-CPA}}$. Definimos un adversario $\mathcal{B}$ contra Exp^{DDH} .

$\mathcal{B}(g^x, g^y, g^z)$	$\mathcal{O}(m_0, m_1)$
1: disabled $\leftarrow 0$	1: if disabled = 1 :
2: $\hat{b} \xleftarrow{\$} \{0, 1\}$	2: return $\perp$
3: $\hat{b}' \leftarrow \mathcal{A}^{\mathcal{O}}(g^x)$	3: disabled $\leftarrow 1$
4: if $\hat{b}' = \hat{b}$:	4: $c^* \leftarrow (g^y, g^z \cdot m_{\hat{b}})$
5: return 0 // “ $z = x \cdot y$ ”	5: return c^*
6: else	
7: return 1 // “ $z \sim U(G)$ ”	

Recordemos la versión “bit-guessing” de $\text{Exp}^{\text{OT-CPA}}$:

$\text{Exp}^{\text{OT-CPA}}(\mathcal{A})$
1: $\hat{b} \xleftarrow{\$} \{0, 1\}$
2: $\hat{b}' \leftarrow \text{Exp}^{\text{OT-CPA}}(\mathcal{A}, \hat{b})$
3: return $[\hat{b}' = \hat{b}]$

Usando la misma demostración de Lemma 3, sabemos que

$$\text{Adv}(\overline{\text{Exp}^{\text{OT-CPA}}}, \mathcal{A}) := \left| \Pr[\overline{\text{Exp}^{\text{OT-CPA}}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| = \frac{1}{2} \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{A}).$$

Por inspección, podemos ver que el entorno de $\mathcal{A}$ en $\text{Exp}^{\text{DDH}}(\mathcal{B}, 0)$, es idéntico al de $\mathcal{A}$ en $\overline{\text{Exp}^{\text{OT-CPA}}}(\mathcal{A})$ si $pk = g^x$ y $c_0^* = g^y$. Por ende,

$$\begin{aligned} \Pr[\text{Exp}^{\text{DDH}}(\mathcal{B}, 0) \Rightarrow 1] &= \Pr[\overline{\text{Exp}^{\text{OT-CPA}}}(\mathcal{A}) \Rightarrow 0] = 1 - \Pr[\overline{\text{Exp}^{\text{OT-CPA}}}(\mathcal{A}) \Rightarrow 1] \\ \iff \Pr[\text{Exp}^{\text{DDH}}(\mathcal{B}, 0) \Rightarrow 1] - \frac{1}{2} &= \frac{1}{2} - \Pr[\overline{\text{Exp}^{\text{OT-CPA}}}(\mathcal{A}) \Rightarrow 1] \\ \iff \text{Adv}(\overline{\text{Exp}^{\text{OT-CPA}}}, \mathcal{A}) &= \left| \Pr[\overline{\text{Exp}^{\text{OT-CPA}}}(\mathcal{A}) \Rightarrow 1] - \frac{1}{2} \right| = \left| \Pr[\text{Exp}^{\text{DDH}}(\mathcal{B}, 0) \Rightarrow 1] - \frac{1}{2} \right|. \end{aligned}$$

Dentro de $\text{Exp}^{\text{DDH}}(\mathcal{B}, 1)$, $\mathcal{A}$ recibe $(g^x, g^y, g^z \cdot m_{\hat{b}})$, donde $g^z \sim U(\mathbb{G})$, independientemente de (g^x, g^y) . Por lo tanto, la vista de $\mathcal{A}$ es independiente de $\hat{b}$, dado que

$$\Pr[h = g^z \cdot m_0] = \Pr[g^z = h \cdot m_0^{-1}] = \frac{1}{|\mathbb{G}|} = \Pr[g^z = h \cdot m_1^{-1}] = \Pr[h = g^z \cdot m_1]$$

Al ser el valor $\hat{b}'$ retornado por $\mathcal{A}$ independiente de $\hat{b}$, y al ser $\hat{b} \sim U(\{0, 1\})$, sigue que

$$\Pr[\hat{b}' \neq \hat{b}] = \Pr[\text{Exp}^{\text{DDH}}(\mathcal{B}, 1) \Rightarrow 1] = \frac{1}{2}.$$

Finalmente,

$$\begin{aligned} \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{A}) &= 2 \cdot \text{Adv}(\overline{\text{Exp}^{\text{OT-CPA}}}, \mathcal{A}) \\ &= 2 \cdot \left| \Pr[\text{Exp}^{\text{DDH}}(\mathcal{B}, 0) \Rightarrow 1] - \frac{1}{2} \right| \\ &= 2 \cdot \left| \Pr[\text{Exp}^{\text{DDH}}(\mathcal{B}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{DDH}}(\mathcal{B}, 1) \Rightarrow 1] \right| \\ &= 2 \cdot \text{Adv}(\text{Exp}^{\text{DDH}}, \mathcal{B}). \end{aligned}$$

Claramente, $t' \approx t$, $\varepsilon' = 2 \cdot \varepsilon$, y Π ofrece $(2\varepsilon, \approx t)$ -OT-IND-CPA. □

Clase 16: cifrado de clave publica multi-mensaje

Fernando Virdia, versión: 0.1, mayo 2025

Definición 34 (IND-CPA). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES.

$\text{Exp}^{\text{CPA}}(\mathcal{A}, b)$	$\mathcal{O}(m_0, m_1)$
1: $(\text{sk}, \text{pk}) \leftarrow \text{Gen}()$	1: $c \leftarrow \text{Enc}(\text{pk}, m_b)$
2: $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\text{pk})$	2: return c
3: return b'	

Decimos que Π otorga (ε, t, q) -indistinguibilidad bajo ataques de mensaje elegido (IND-CPA, “indistinguishability under chosen-plaintext attacks”), si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$ y usa $\leq q$ queries a $\mathcal{O}$, tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{CPA}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{CPA}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{CPA}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

Comentario 54. Respecto a la definición simétrica, no escribimos explícitamente el requisito $|m_1| = |m_2|$ simplemente por que generalmente $m_i \notin \{0, 1\}^*$, y tienen un tamaño predeterminado.

Lemma 17 (PKE OT-IND-CPA $\Rightarrow$ PKE IND-CPA). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES que ofrece (ε, t) -OT-IND-CPA. Entonces ofrece (ε', t', q) -IND-CPA con $\varepsilon' = q \cdot \varepsilon$, $t' \approx t$.

Proof. Supongamos un adversario $\mathcal{A}$ que usa q queries contra Exp^{CPA} . Definimos q adversarios $\mathcal{B}_1, \dots, \mathcal{B}_q$ contra $\text{Exp}^{\text{OT-CPA}}$.

$\mathcal{B}_i^E(\text{pk})$	$\mathcal{O}(m_0, m_1)$
1: $\text{cnt} \leftarrow 0$	1: $\text{cnt} \leftarrow \text{cnt} + 1$
2: $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\text{pk})$	2: if $\text{cnt} < i$:
3: return b'	3: $c \leftarrow \text{Enc}(\text{pk}, m_0)$
	4: elseif $\text{cnt} = i$:
	5: $c \leftarrow E(m_0, m_1)$
	6: else :
	7: $c \leftarrow \text{Enc}(\text{pk}, m_1)$
	8: return c

Notemos que siendo Π (ε, t) -OT-IND-CPA,

$$\left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_i, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_i, 1) \Rightarrow 1] \right| \leq \varepsilon \quad \forall i \in [q].$$

Inspeccionemos el patrón de como los $\mathcal{B}_i$ les responden queries a $\mathcal{A}$ dentro de $\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_i, b)$. En lo siguiente escribimos $\mathcal{E}(m)$ en lugar de $\text{Enc}(\text{pk}, m)$.

adversario	b	numero de query						
		1	...	$i-1$	i	$i+1$	...	q
$\mathcal{B}_1$	1	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$
$\mathcal{B}_1$	0	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$
$\vdots$								
$\mathcal{B}_{i-1}$	1	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$
$\mathcal{B}_{i-1}$	0	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_0)$	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$
$\mathcal{B}_i$	1	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_0)$	$\mathcal{E}(m_1)$	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$
$\mathcal{B}_i$	0	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_0)$	$\mathcal{E}(m_0)$	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$
$\mathcal{B}_{i+1}$	1	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_0)$	$\mathcal{E}(m_0)$	$\mathcal{E}(m_1)$	...	$\mathcal{E}(m_1)$
$\vdots$								
$\mathcal{B}_q$	0	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_0)$	$\mathcal{E}(m_0)$	$\mathcal{E}(m_0)$	...	$\mathcal{E}(m_0)$

Notamos el siguiente patrón:

$$\forall i < q, \quad \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_i, 0) \Rightarrow 1] = \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_{i+1}, 1) \Rightarrow 1]. \quad (\star)$$

también notamos que del punto de vista de $\mathcal{A}$, $\text{Exp}^{\text{CPA}}(\mathcal{A}, 1)$ es idéntico a $\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 1)$, y $\text{Exp}^{\text{CPA}}(\mathcal{A}, 0)$ es idéntico a $\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_q, 0)$. Por lo tanto,

$$\begin{aligned} \text{Adv}(\text{Exp}^{\text{CPA}}, \mathcal{A}) &= \left| \Pr[\text{Exp}^{\text{CPA}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{CPA}}(\mathcal{A}, 1) \Rightarrow 1] \right| \\ &= \left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_q, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 1) \Rightarrow 1] \right| \\ &= \left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_q, 0) \Rightarrow 1] \right. \\ &\quad \left. - \underbrace{\Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_q, 1) \Rightarrow 1] + \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_{q-1}, 0) \Rightarrow 1]}_{= 0 \text{ por } (\star)} \right. \\ &\quad \left. - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 1) \Rightarrow 1] \right| \\ &\leq \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{B}_q) + \left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_{q-1}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 1) \Rightarrow 1] \right| \\ &\leq \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{B}_q) + \left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_{q-1}, 0) \Rightarrow 1] \right. \\ &\quad \left. - \underbrace{\Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_{q-1}, 1) \Rightarrow 1] + \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_{q-2}, 0) \Rightarrow 1]}_{= 0 \text{ por } (\star)} \right. \\ &\quad \left. - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 1) \Rightarrow 1] \right| \\ &\leq \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{B}_q) + \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{B}_{q-1}) + \left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_{q-2}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 1) \Rightarrow 1] \right| \\ &\quad \vdots \\ &\leq \sum_{k=q}^2 \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{B}_k) + \left| \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{OT-CPA}}(\mathcal{B}_1, 1) \Rightarrow 1] \right| \\ &= \sum_{k=q}^1 \text{Adv}(\text{Exp}^{\text{OT-CPA}}, \mathcal{B}_k) \leq \sum_{k=q}^1 \varepsilon = q \cdot \varepsilon. \end{aligned}$$

Notamos que $\mathcal{B}_i$ y $\mathcal{A}$ corren en tiempo parecido, por lo que $t' \approx t$, y que si $\mathcal{A}$ usa $\leq q$ queries, $\varepsilon' \leq q \cdot \varepsilon$. Por lo tanto, si Π es (ε, t) -OT-IND-CPA, entonces también es $(q \cdot \varepsilon, \approx t, q)$ -IND-CPA. $\square$

Comentario 55. La técnica que acabamos de usar se llama argumento híbrido (“hybrid argument”), porque requiere definir experimentos “híbridos” entre $\text{Exp}^{\text{CPA}}(\mathcal{A}, 0)$ y $\text{Exp}^{\text{CPA}}(\mathcal{A}, 1)$ con los cuales calculamos la ventaja total, utilizando la ventaja de distinguir pares de híbridos.

Comentario 56. Lemma 17 implica un modo de cifrar mensajes $m \in \{0, 1\}^*$ arbitrariamente largos. Sea $m = m_1, m_2, \dots, m_n$ una cadena de n bits. Podemos generar una cadena de cifrados

$$\text{Enc}'(\text{pk}, m \in \{0, 1\}^n) := (\text{Enc}(\text{pk}, g^{m_1}), \text{Enc}(\text{pk}, g^{m_2}), \dots, \text{Enc}(\text{pk}, g^{m_n}))$$

donde g^{m_i} es g^1 si $m_i = 1$ y $g^0 = e$ si $m_i = 0$. Si Π es (ε, t, q) -IND-CPA con $q \geq n$, el cifrado es seguro. Notablemente, no solo esta técnica es muy ineficiente (vamos a ver una mejor alternativa), si no que también no funciona con nociones de seguridad mas fuertes, como IND-CCA.

Comentario 57. A diferencia de SKE (c.f. Remark 29), en el caso de PKE, OT-IND-CPA $\implies$ IND-CPA.

EJERCICIO: Por que existe esta diferencia?

Por que en el caso de PKE, el adversario siempre puede evaluar $\text{Enc}(\text{pk}, m)$ internamente. Esto le permite implementar $\mathcal{E}(m)$ en la demostración.

En el caso SKE, para evaluar $\text{Enc}(k, m)$ el adversario $\mathcal{B}$ contra $\text{Exp}^{\text{OT-CPA}}$ tiene que utilizar la sola query al oraculo de cifrado $E(\cdot, \cdot)$ otorgadole por $\text{Exp}^{\text{OT-CPA}}$ para realizar $\mathcal{E}(m) = E(\cdot, \cdot)$ en el primer “hop”. Inmediatamente $\mathcal{B}$ se quedaria sin acceso al oraculo, impidiendole completar la demostración.

Clase 17: PKE y adversarios activos

Fernando Virdia, versión: 0.1, mayo 2025

9.1 Adversarios activos contra PKE

Al estudiar cifrado simétrico seguro contra adversarios activos, introducimos la noción de seguridad INT-CTXT (Definition 21). Esta propone que un adversario no debería ser capaz de crear cifrados simétricos válidos de manera independiente, a menos utilizar un oráculo de cifrado. Intuitivamente, esto quiere decir que si el adversario modifica un cifrado c en tránsito, estas modificaciones van a ser detectadas por quien los recibe, que rechazará c .

Ejemplo 8. *Adversarios activos son una preocupación también en el caso de PKE. Por ejemplo, dado un cifrado ElGamal $c = (c_1, c_2) = (g^y, g^{xy} \cdot m)$ de un mensaje $m \in \mathbb{G}$, $\mathcal{A}$ puede elegir cualquier elemento $\Delta \in \mathbb{G}$, y generar $c' = (c_1, c_2 \cdot \Delta)$, que descifraría a $m' = m \cdot \Delta$.*

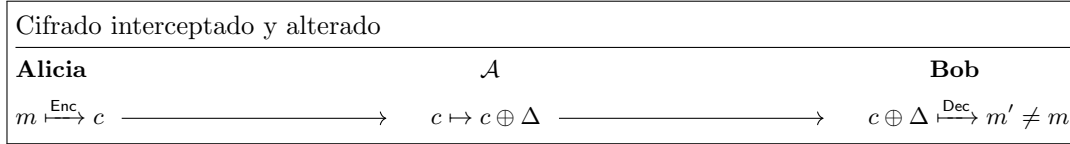
Lamentablemente, no es posible para un PKES otorgar seguridad INT-CTXT.

Ejemplo 9. *Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES. Dada una clave pública pk , cualquier adversario $\mathcal{A}$ con acceso a pk puede evaluar $c \leftarrow \text{Enc}(\text{pk}, m)$ internamente, y obtener un cifrado c válido, sin requerir un oráculo de cifrado. Por lo tanto, una definición análoga a INT-CTXT no es inmediatamente traducible al caso PKE.*

Comentario 58. *En este ejemplo, aunque $\mathcal{A}$ ha generado cifrados válidos independientemente, estos no fueron obtenidos modificando cifrados en tránsito.*

Es posible obtener una definición de seguridad “intermedia” entre IND-CPA y AE que otorgue seguridad activa, aunque no otorgue INT-CTXT?

La respuesta es positiva, y pasa por capturar de qué manera un cifrado $c \oplus \Delta$ modificado por $\mathcal{A}$ puede resultar en un ataque.



En sí, $c \oplus \Delta$ no le da ninguna información a $\mathcal{A}$. Para serle útil, $\mathcal{A}$ tiene que mandarle $c \oplus \Delta$ a su víctima Bob, y observar como esta reacciona. De cierta manera, $\mathcal{A}$ necesita un *oráculo de descifrado* otorgado por Bob, que le permita observar la reacción de Bob al descifrar $c \oplus \Delta$.

Definición 35 (OT-IND-CCA PKE). *Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES.*

Exp $\text{OT-IND-CCA}(\mathcal{A}, b)$	$\mathcal{O}(m_0, m_1)$	$D(c)$
1: $(\text{sk}, \text{pk}) \leftarrow \text{Gen}()$	1: if disabled = 1 :	1: if $c \in Q$:
2: disabled $\leftarrow 0$	2: return $\perp$	2: return $\perp$
3: $Q \leftarrow \{\}$	3: disabled $\leftarrow 1$	3: $m \leftarrow \text{Dec}(\text{sk}, c)$
4: $b' \leftarrow \mathcal{A}^{\mathcal{O}(\cdot), D(\cdot)}(\text{pk})$	4: $c \leftarrow \text{Enc}(\text{pk}, m_b)$	4: return m
5: return b'	5: $Q \leftarrow Q \cup \{c\}$	
	6: return c	

Decimos que Π otorga $(\varepsilon, t, q_e, q_d)$ -indistinguibilidad bajo ataques de cifrado elegido (IND-CCA, “indistinguishability under chosen-ciphertext attacks”), si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$, realiza $\leq q_e$ queries a $\mathcal{O}$ (one-time: $q_e = 1$) y realiza $\leq q_d$ queries a D , tiene ventaja

$$\text{Adv}(\text{Exp}_{\text{OT-IND-CCA}}^{\text{OT-IND-CCA}}, \mathcal{A}) := \left| \Pr[\text{Exp}_{\text{OT-IND-CCA}}^{\text{OT-IND-CCA}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}_{\text{OT-IND-CCA}}^{\text{OT-IND-CCA}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

Lemma 18 (PKE OT-IND-CCA $\Rightarrow$ PKE IND-CCA). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES que ofrece $(\varepsilon, t, 1, q_d)$ -IND-CCA (i.e. OT-IND-CCA). Entonces ofrece $(q_e \cdot \varepsilon, \approx t, q_e, q_d)$ -IND-CCA con $q_e \in \mathbb{Z}_{>0}$.

Demostración. La demostración es idéntica a la de PKE OT-IND-CPA $\Rightarrow$ PKE IND-CPA, Lemma 17. $\square$

Ejemplo 10. En la practica, un oráculo de descifrado podría suceder en sistemas de cifrado de email. Un proveedor de correo electrónico “Daniel” podría modificar un correo “De Alicia para Bob: $\text{Enc}(\text{pk}_{\text{Bob}}, m)$ ” en “De Daniel para Bob: $\text{Enc}(\text{pk}_{\text{Bob}}, m)$ ”. Al responderle a Dan, Bob podría citar los contenidos originales de m , resultando en un oráculo de descifrado.

Lemma 19 (AE $\Rightarrow$ IND-CCA $\Rightarrow$ IND-CPA). La noción de seguridad IND-CCA es “intermedia” entre AE e IND-CPA.

Boceto de demostración. En ambos casos de cifrado simétrico y de clave pública, IND-CCA $\Rightarrow$ IND-CPA dado que los juegos Exp^{CPA} y Exp^{CCA} son idénticos, excepto que Exp^{CCA} le otorga menos oráculos al adversario $\mathcal{A}$. En lo siguiente suponemos un SKES es correcto. En el caso de cifrado simétrico, queremos demostrar que AE $\Rightarrow$ IND-CCA. Dado que AE requiere INT-CTXT, $\mathcal{A}$ puede solamente generar cifrados c_i válidos, tales que $\text{Dec}(k, c_i) \neq \perp$, solo a través del oráculo $E(\cdot)$ de cifrado. Esto quiere decir que si c_i es válido, $\mathcal{A}$ ya conoce el valor $m_i = \text{Dec}(k, c_i)$, dado que obtuvo c_i llamando $E(m_i)$. De otro lado, si c_i no es válido, $\mathcal{A}$ ya sabe que $\text{Dec}(k, c_i) = \perp$. Si así no fuera, $\mathcal{A}$ habría falsificado un cifrado, contradiciendo IND-CTXT y por lo tanto AE. En otras palabras, el oráculo $D(\cdot)$ no le da ninguna ventaja a $\mathcal{A}$, dado que no puede llamarlo sobre cifrados para los que ya no conozca $\text{Dec}(k, c_i)$. Si $\mathcal{A}$ tuviera uso para el oráculo $D(\cdot)$, es porque $\mathcal{A}$ podría generar cifrados c_i para los cuales prevee $m_i := \text{Dec}(k, c_i) \neq \perp$ pero no conoce m_i , contradiciendo INT-CTXT. $\square$

La noción de IND-CCA es una opción válida para proteger PKE de adversarios activos. En la practica, es mas comun que protocolos criptografico usen alguna noción (mas difícil de formalizar) de intercambio de claves autenticado (*authenticated key exchange*).

En la siguiente clase vamos a ver como obtener IND-CCA PKE, usando la metodología KEM-DEM.

Clase 18: cifrado híbrido, la metodología KEM-DEM

Fernando Virdia, versión: 0.1, mayo 2025

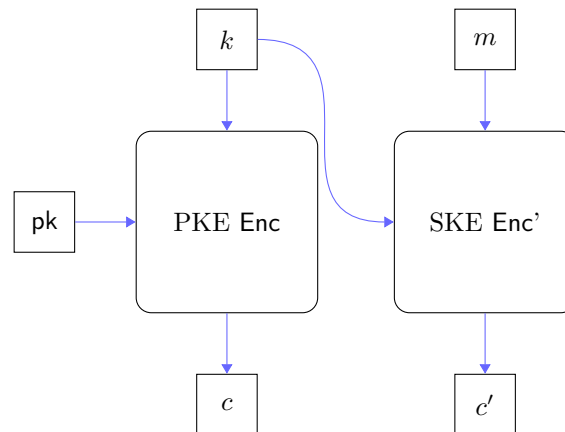
9.2 Cifrado híbrido

Es interesante observar que en practica, el costo de un cifrado a clave publica es generalmente mas alto del de un cifrado simétrico. Lanzando `openssl speed rsa aes` en mi laptop:

- En 10s, 6852 descifrados y 344141 cifrados de RSA-3072 (“128 bits de seguridad”), mensaje de ≈ 3000 bits.
- En 10s, 941746 cifrados o descifrados de AES-128 CBC-mode, mensaje de ≈ 131072 bits.

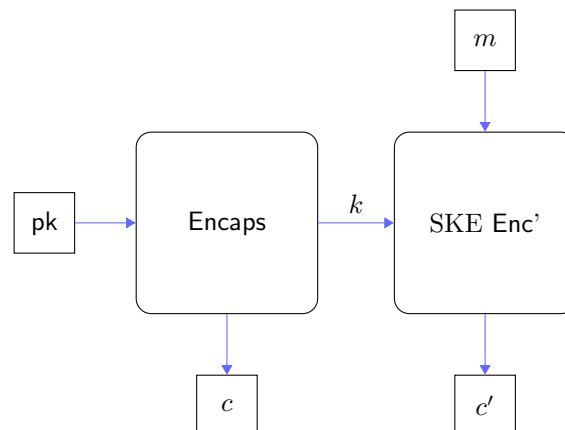
Por este motivo, aunque es técnicamente posible usar sistemas como ElGamal para mensajes arbitrariamente largos, En la practica PKE y KEX ambos utilizan una primera fase a clave publica para generar claves para un SKES. En el caso de PKE, este método se llama de cifrado híbrido (“hybrid encryption”).

Por ejemplo, sean $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES, $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ un SKES. Podemos cifrar una clave k de Π' utilizando Π , y luego cifrar un mensaje arbitrariamente largo utilizando $\text{Enc}'(k, \cdot)$. Transmitiendo ambos cifrados, es posible recuperar el mensaje.



Una técnica hoy mas común es remplazar el PKES con un mecanismo de encapsulamiento de claves (KEM, “key-encapsulation mechanism”).

Se puede pensar un KEM como un sistema de cifrado PKES de “mensajes al azar”. Esta restricción generalmente le permite a un KEM ser mas compacto y eficiente que un PKES que ofrece similar seguridad.



Un sistema híbrido que utiliza un KEM se lo llama también KEM-DEM, donde el cifrado simétrico tiene el rol de mecanismo de encapsulamiento de datos (DEM, “data-encapsulation mechanism”). Para obtener PKE que otorgue seguridad IND-CCA, vamos a utilizar una construcción KEM-DEM.

Definición 36 (KEM, mecanismo de encapsulado de claves). Sean $\mathcal{K}_p$, $\mathcal{K}_s$, $\mathcal{M}$, $\mathcal{C}$ conjuntos. Decimos que $\Pi = (\text{Gen}, \text{Encaps}, \text{Decaps})$ forman un mecanismo de encapsulamiento de claves (“key-encapsulation mechanism”, KEM) con espacio de claves publicas $\mathcal{K}_p$, de claves privadas $\mathcal{K}_s$, de claves a encapsular $\mathcal{M}$, y de cifrados o “encapsulados” $\mathcal{C}$, si

- Gen es un algoritmo aleatorio, $\text{Gen}: \emptyset \rightarrow \mathcal{K}_s \times \mathcal{K}_p$ que retorna una tupla (sk, pk) ;
- Encaps es un algoritmo aleatorio, $\text{Encaps}: \mathcal{K}_p \rightarrow \mathcal{M} \times \mathcal{C}$ que retorna una clave k y un encapsulamiento c de k ;
- Decaps es un algoritmo determinista $\text{Dec}: \mathcal{K}_s \times \mathcal{C} \rightarrow \mathcal{M}$, que decapsula c , retornando k ;

donde Gen, Encaps y Decaps son “eficientes”. Decimos que Π es δ -correcto si

$$\Pr_{\substack{(sk, pk) \sim \text{Gen}(), \\ \$ \text{ en Encaps}}}[(k, c) \leftarrow \text{Encaps}(pk, \$) : \text{Decaps}(sk, c) \neq k] \leq \delta.$$

Comentario 59. En el resto del curso, consideramos solamente KEMs (0-)correctos.

Ahora podemos definir una noción de “KEM IND-CCA”.

Definición 37 (OT-IND-CCA KEM). Sea $\Pi = (\text{Gen}, \text{Encaps}, \text{Decaps})$ un KEM.

$\text{Exp}^{\text{KEM-OT-CCA}}(\mathcal{A}, b)$	$D(c)$
1: $(\text{sk}, \text{pk}) \leftarrow \text{Gen}()$	1: if $c = c^*$:
2: $(k_0, c^*) \leftarrow \text{Encaps}(pk)$	2: return $\perp$
3: $k_1 \xleftarrow{\$} \mathcal{M}$	3: $k \leftarrow \text{Decaps}(\text{sk}, c)$
4: $b' \leftarrow \mathcal{A}^D(pk, k_b, c^*)$	4: return k
5: return b'	

Decimos que Π otorga “one-time” (ε, t, q_d) -indistinguibilidad bajo ataques de cifrado elegido (KEM IND-CCA, “indistinguishability under chosen-ciphertext attacks”), si cualquier adversario $\mathcal{A}$ que corre en tiempo $\leq t$, y realiza $\leq q_d$ queries a D , tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{KEM-OT-CCA}}, \mathcal{A}) := \left| \Pr[\text{Exp}^{\text{KEM-OT-CCA}}(\mathcal{A}, 0) \Rightarrow 1] - \Pr[\text{Exp}^{\text{KEM-OT-CCA}}(\mathcal{A}, 1) \Rightarrow 1] \right| \leq \varepsilon.$$

Utilizando un KEM que otorga OT-IND-CCA KEM y un DEM que otorga AE, es posible obtener un PKE que otorga OT-IND-CCA, y por lo tanto IND-CCA (c.f. Lemma 18).

Teorema 6 (OT-IND-CCA KEM + AE $\Rightarrow$ OT-IND-CCA PKE, informal). Sea $\Pi = (\text{Gen}, \text{Encaps}, \text{Decaps})$ un KEM, y $\Pi' = (\text{Gen}', \text{Enc}', \text{Dec}')$ un SKES, tales que el espacio $\Pi.\mathcal{M}$ de claves a encapsular de Π es igual al espacio de claves $\Pi'.\mathcal{K}$ de Π' , $\Pi.\mathcal{M} = \Pi'.\mathcal{K}$, y que $\text{Gen}'() = “k \xleftarrow{\$} \Pi'.\mathcal{K}”$. Sea $\Pi^{hy} = (\text{Gen}^{hy}, \text{Enc}^{hy}, \text{Dec}^{hy})$ el PKES obtenido de manera KEM-DEM,

$\text{Gen}^{hy}()$	$\text{Enc}^{hy}(pk, m)$	$\text{Dec}^{hy}(\text{sk}, c = (c_0, c_1))$
1: $(\text{sk}, \text{pk}) \leftarrow \text{Gen}()$	1: $k, c_0 \leftarrow \text{Encaps}(pk)$	1: $k \leftarrow \text{Decaps}(\text{sk}, c_0)$
2: return (sk, pk)	2: $c_1 \leftarrow \text{Enc}'(k, m)$	2: if $k = \perp$:
	3: $c \leftarrow (c_0, c_1)$	3: return $\perp$
	4: return c	4: $m' \leftarrow \text{Dec}'(k, c_1)$
		5: return m'

Si Π otorga KEM OT-IND-CCA y Π' otorga AE, entonces Π^{hy} otorga PKE OT-IND-CCA.

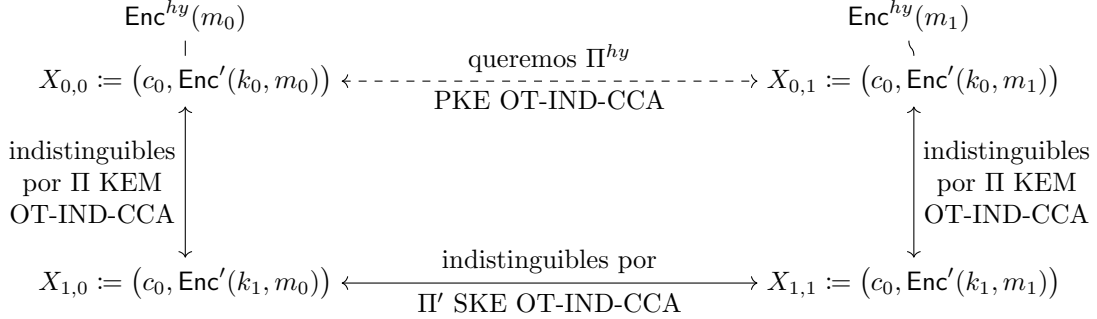
Boceto de demostración.

Para demostrar el teorema, los pasos son

1. demostrar que AE $\Rightarrow$ una noción simétrica de OT-IND-CCA.
2. demostrar que Π^{hy} otorga PKE OT-IND-CCA,.

Una demostración de Item 1 se encuentra en [BS23, Theorem 9.1]. Para demostrar Item 2, la idea es de asumir un adversario $\mathcal{A}$ que juega una variante Exp de $\text{Exp}^{\text{OT-IND-CCA}}$ contra Π^{hy} con dos parámetros, $\text{Exp}(\mathcal{A}, b, \hat{b})$. $\mathcal{A}$ tiene que adivinar b dado un cifrado

$$(c_0, \text{Enc}'(k_{\hat{b}}, m_b)), \text{ donde } (k_0, c_0) \leftarrow \text{Encaps}(\text{pk}) \text{ y } k_1 \xleftarrow{\$} \Pi.\mathcal{K} = \Pi'.\mathcal{K}.$$



Notamos que:

Si $\hat{b} = 0$, entonces $(c_0, \text{Enc}'(k_0, m_b)) = \text{Enc}^{hy}(m_b)$, por lo que para demostrar el resultado, queremos una cota superior para la distancia entre los $\text{Exp}(\mathcal{A}, 0, 0)$ y $\text{Exp}(\mathcal{A}, 1, 0)$.

$X_{b,\hat{b}}$ es una función de la variable aleatoria $Y_{\hat{b}} := (c_0, k_{\hat{b}})$ (la función es $f_{m_b}(x, y) \mapsto (x, \text{Enc}'(y, m_b))$).

Por Π ser un IND-CCA KEM, $Y_{b,0}$ y $Y_{b,1}$ son indistinguibles. Por *data processing inequality* (Theorem 2), también lo tienen que ser $X_{b,0}$ y $X_{b,1}$.

$\text{Enc}'(k_1, m_0)$ y $\text{Enc}'(k_1, m_1)$ son indistinguibles por ser Π' un OT-IND-CCA SKES.

La semejanza de $X_{0,0}$ con $X_{0,1}$, de $X_{0,1}$ con $X_{1,1}$ y $X_{1,1}$ con $X_{1,0}$ nos da el resultado. $\square$

9.3 Transformada de Fujisaki-Okamoto

Nos falta una sola componente: un KEM que otorgue OT-IND-CCA.

Para construir un KEM OT-IND-CCA, vamos a utilizar una variante de la “transformada de Fujisaki-Okamoto” (FO, “Fujisaki-Okamoto transform”). Esta nos permite combinar un PKES que otorga IND-CPA, con dos funciones de hash, obteniendo un KEM OT-IND-CCA!

Hay muchas variantes de FO [HHK17], y esta no debe considerarse la única manera o la manera “correcta”.

Queremos un KEM que resista ataques bajo queries de decapsulado. La idea es volver las queries de decapsulado inútiles. Supongamos que $\mathcal{A}$ pide una query $D(c)$ al oráculo D de decapsulado. Puede haber generado c de diferentes maneras. Llamando $c \leftarrow \text{Encaps}(\text{pk})$, que internamente llama $\text{Enc}(\text{pk}, m)$. En este caso el oráculo D es inútil, $\mathcal{A}$ conoce m ! Modificando algún $c \mapsto c + \Delta$ en tránsito “inventándose” algún $c \in \mathcal{C}$. Vamos a diseñar un PKE donde “inventarse” $c \in \mathcal{C}$ tal que $\text{Dec}(c) \neq \perp$ sea difícil, y tal que modificaciones tipo $c + \Delta$ sean detectables.

Definición 38 (PKE Rígido). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES. Decimos que Π es rígido si $\forall(\text{sk}, \text{pk}) \leftarrow \text{Gen}(), \forall c \in \mathcal{C}$

$$\text{Dec}(\text{sk}, c) = \perp \vee \text{Enc}(\text{pk}, \text{Dec}(\text{sk}, c)) = c.$$

Comentario 60. Para que un PKE Π sea rígido, necesita ser determinista! De otra manera, el valor de $\text{Enc}(\text{pk}, \text{Dec}(\text{sk}, c))$ sería una variable aleatoria.

- Podemos obtener un PKE determinista usando cifrando aleatorio $\text{Enc}(\text{pk}, m; r)$ donde seteamos las monedas $r = G(m)$, con $G: \mathcal{M} \rightarrow \mathcal{R}$ una función de hash criptográfica.
- Al ser determinista, Π no puede otorgar PKE IND-CPA. En su lugar, vamos a suponer que Π otorga una noción más débil, one-wayness: dados $\text{pk}, \text{Enc}(\text{pk}, m)$, recuperar m es difícil.

Lemma 20. Sea Π un PKE rígido y correcto. Entonces $c = \text{Enc}(\text{pk}, m) \iff \text{Dec}(\text{sk}, c) = m$.

Demostración. Supongamos que $c = \text{Enc}(\text{pk}, m)$. Por correctness, $\text{Dec}(\text{sk}, c) = m$. Supongamos $\text{Dec}(\text{sk}, c) = m$. Por rigidez, $\text{Enc}(\text{pk}, \text{Dec}(\text{sk}, c)) = \text{Enc}(\text{pk}, m) = c$. $\square$

Definición 39 (Un PKE rigido). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES. Sea $G: \mathcal{M} \rightarrow \mathcal{R}$ una función de hash criptográfica. Construimos un PKES rigido $\Pi' = T[\Pi, G] = (\text{Gen}', \text{Enc}', \text{Dec}')$ donde

$\text{Gen}'()$	$\text{Enc}'(\text{pk}, m)$	$\text{Dec}'(\text{sk} = (\text{sk}_\Pi, \text{pk}), c)$
1: $(\text{sk}_\Pi, \text{pk}) \leftarrow \text{Gen}'()$	1: $c \leftarrow \text{Enc}(\text{pk}, m; G(m))$	1: $m' \leftarrow \text{Dec}(\text{sk}_\Pi, c)$
2: $\text{sk} \leftarrow (\text{sk}_\Pi, \text{pk})$	2: return c	2: if $m' = \perp$ or $\text{Enc}(\text{pk}, m'; G(m')) \neq c$:
3: return (sk, pk)		3: return $\perp$
		4: return m'

Lemma 21. El PKES en Definición 39 es rigido.

Demostración. Por definición de Dec' , $m' = \text{Dec}(\text{sk}, c)$ es retornado si y solo si $\text{Enc}'(m') = c$ o $m' = \perp$. Por lo tanto $\text{Enc}'(\text{pk}, \text{Dec}'(\text{sk}, c)) = c \vee \text{Dec}'(\text{sk}, c) = \perp$. $\square$

Con un PKE rigido, podemos definir el KEM de Fujisaki-Okamoto.

Definición 40 (FO KEM). Sea $\mathcal{M} = \{0, 1\}^\ell$ un conjunto finito. Sean $\Pi^{\text{Rig}} = (\text{Gen}^{\text{Rig}}, \text{Enc}^{\text{Rig}}, \text{Dec}^{\text{Rig}})$ un PKES rigido. Sea $H: \Pi^{\text{Rig}} \cdot \mathcal{M} \rightarrow \mathcal{M}$ una función de hash criptográfica. Definimos el KEM $\Pi' = U_m^\perp[\Pi^{\text{Rig}}, H] = (\text{Gen}, \text{Encaps}, \text{Decaps})$ con espacio de claves a encapsular $\mathcal{M}$,

$\text{Gen}()$	$\text{Encaps}(\text{pk})$	$\text{Decaps}(\text{sk}, c)$
1: $(\text{sk}, \text{pk}) \leftarrow \text{Gen}^{\text{Rig}}()$	1: $m \xleftarrow{\$} \Pi^{\text{Rig}} \cdot \mathcal{M}$	1: $m' := \text{Dec}^{\text{Rig}}(\text{sk}, c)$
2: return (sk, pk)	2: $c := \text{Enc}^{\text{Rig}}(\text{pk}, m)$	2: if $m' = \perp$:
	3: $k := H(m)$	3: return $\perp$
	4: return (k, c)	4: $k := H(m')$
		5: return k

Comentario 61. Notamos la necesidad de usar una función de hash $H: \Pi \cdot \mathcal{M} \rightarrow \mathcal{M}$ dentro del KEM de Fujisaki-Okamoto. Esto es porque dado que Π es rigido, también es determinista. El juego KEM IND-CCA le otorga al adversario un encapsulado $c := \text{Enc}^{\text{Rig}}(\text{pk}, m)$ y una clave k que puede ser $H(m)$ o aleatoria. Si usáramos m directamente como la clave k , entonces $\mathcal{A}$ podría testear distinguir k como no-aleatoria simplemente verificando si $c = \text{Enc}^{\text{Rig}}(\text{pk}, m)$ o no.

Para que el KEM FO sea IND-CCA, vamos a necesitar construir nuestro PKE rigido a partir de un PKE que otorga “alta entropía” de cifrado, o “ γ -spreadness”.

Definición 41 (γ -spreadness). Sea $\Pi = (\text{Gen}, \text{Enc}, \text{Dec})$ un PKES con cifrado no determinista. Decimos que Π es “ γ -spread” si $\forall (\text{sk}, \text{pk}) \leftarrow \text{Gen}()$, $m \in \mathcal{M}$, $c \in \mathcal{C}$,

$$\Pr[\text{Enc}(\text{pk}, m) = c] \leq 2^{-\gamma},$$

donde la probabilidad es sobre las monedas de Enc .

Comentario 62. La mayoría de los IND-CPA PKES otorgan spreadness. Si no es así, cualquier IND-CPA PKE puede ser modificado para otorgar spreadness de manera sencilla [BS23, Exercise 12.10]. ElGamal PKE es naturalmente γ -spread con $\gamma = 1/|\mathbb{G}|$.

Teorema 7 (OW correct γ -spread PKE + ROM $\Rightarrow$ OT-IND-CCA KEM, informal). Sea $\mathcal{M} = \{0, 1\}^\ell$. Sea Π un PKES non-determinista correcto y γ -spread. Sea $G: \Pi \cdot \mathcal{M} \rightarrow \Pi \cdot \mathcal{R}$ una función de hash criptográfica. Sea $H: \Pi \cdot \mathcal{M} \rightarrow \mathcal{M}$ una función de hash criptográfica independiente de G . Sea $\Pi^{\text{Rig}} = T[\Pi, G]$ el PKE rigido definido en Definición 39. Sea $\Pi' = U_m^\perp[\Pi^{\text{Rig}}, H]$ el KEM definido en Definición 40. Si Π^{Rig} es one-way (OW), y modelamos G y H como oráculos aleatorios, entonces Π' es OT-IND-CCA.

Boceto de demostración. Una demostración formal de esta variante FO puede obtenerse combinando Teoremas 3.2 y 3.3 en [HHK17], o [HHMS25].

La idea es la siguiente. Sea $\mathcal{A}$ un adversario KEM OT-IND-CCA. Construimos un KEM donde intuitivamente un oráculo de decapsulado no le sirve, en cuanto el PKE utilizado internamente es rigido. En la demostración, vamos a construir un adversario $\mathcal{B}$ contra el OW PKE. $\mathcal{B}$ va a ejecutar $\mathcal{A}$, y cada vez que $\mathcal{A}$ pide una query al oráculo de decapsulado, $\mathcal{B}$ va a generar la respuesta *sin tener*

acceso a sk . Sea $\Pi^{\text{Rig}} = (\text{Gen}^{\text{Rig}}, \text{Enc}^{\text{Rig}}, \text{Dec}^{\text{Rig}})$ el PKE rigido. Dada una query $D(c)$, sabemos que $\text{Dec}^{\text{Rig}}(\text{sk}, c) = \perp$, o $\text{Enc}^{\text{Rig}}(\text{pk}, \text{Dec}^{\text{Rig}}(\text{sk}, c)) = c$.

Si c fue generado usando $\text{Encaps}(\text{pk})$, internamente Enc^{Rig} ha computado $G(m)$ como monedas para el cifrado. Dado que G es un oraculo aleatorio, en lugar de utilizar la funcion de hash, $\mathcal{A}$ calcula $G(m)$ a traves de una query a una “API” que implementa G . $\mathcal{B}$ va a inspeccionar cada query hecha por $\mathcal{A}$ a G , y va a guardarla en una hash-map Q_G indexada con por $c := \text{Enc}^{\text{Rig}}(\text{pk}, m)$: $Q_G[c] = m$. Si $\mathcal{A}$ pide $D(c)$, $\mathcal{B}$ entonces puede retornar $Q_G[c] = m = \text{Dec}^{\text{Rig}}(\text{sk}, c)$ correctamente, sin conocer sk . Si $\mathcal{A}$ no ha generado c a traves de $\text{Encaps}(\text{pk})$, supongamos que $\text{Dec}^{\text{Rig}}(\text{sk}, c) = m' \neq \perp$. Por γ -spreadness, $\Pr[\text{Enc}(\text{pk}, m') = c] \leq 2^{-\gamma}$. Es improbable ($p \leq 2^{-\gamma}$) generar el cifrado $c = \text{Enc}^{\text{Rig}}(\text{pk}, m')$ de m' que descifra correctamente bajo Dec^{Rig} , sin haber calculado $G(m')$. Por rigidez y correctness, $m' = \text{Dec}^{\text{Rig}}(\text{sk}, c) \Leftrightarrow c = \text{Enc}^{\text{Rig}}(\text{pk}, m')$, por lo que m' va a estar guardado bajo el correcto $Q_G[c]$, y no bajo otro cifrado. Por lo cual nuevamente $\mathcal{B}$ puede responder $D(c)$ utilizando $Q_G[c]$, fallando encontrar $Q_G[c]$ con probabilidad $\leq 2^{-\gamma}$. Si finalmente Q_G no contiene un index c , entonces $\mathcal{B}$ responde $D(c)$ con $\perp$.

Al recibir un challenge (pk, c^*) , $\mathcal{B}$ va a generar $k_1 \leftarrow U(\mathcal{M})$, y ejecutar $\mathcal{A}(\text{pk}, k_1, c^*)$. La simulacion de D es correcta con probabilidad alta ($\geq 1 - 2^{-\gamma}$), por lo tanto $\mathcal{A}$ va a lograr distinguir que $k_1 \neq H(m^* := \text{Dec}^{\text{Rig}}(\text{sk}, c^*))$. Pero al ser H un oraculo aleatorio, la distribucion de k_1 y la de $H(m^*)$ son identicas, a menos de no haber calculado $H(m^*)$ explicitamente! Por lo que la unica manera de distinguir, es de hacer en algun momento una query de m^* a la API de H . Habiendo $\mathcal{B}$ monitoreado queries a H , guardando las queries en una lista Q_H construida como Q_G , $\mathcal{B}$ va a encontrar m^* bajo $Q_H[c^*]$ cuando $\mathcal{A}$ termine de ejecutar. Por lo tanto $\mathcal{B}$ rompe OW PKE, contradiciendo nuestra suposicion de que esto es dificil. $\square$

Comentario 63. Con los cifrados del curso, Podríamos realizar un IND-CCA PKE combinando un FO KEM basado sobre ElGamal PKE, junto a un AE obtenido como EtM. Un cifrado parecido y que también otorga IND-CCA es parte del estándar ISO/IEC 18033-2 para cifrado de claves publicas, bajo el nombre de DHIES (ECIES si $\mathbb{G}$ es una curva elíptica). El estándar FIPS ML-KEM para cifrados post-cuánticos, llamado también Kyber, es un OT-IND-CCA KEM obtenido de un tipo de postulado similar a DDH, a través de una variante de FO parecida a la que acabamos de ver.

Clase 19: autenticado de clave publica

Fernando Virdia, versión: 0.1, mayo 2025

Finalmente Alicia y Bob pueden comunicar de manera confidencial, sin tener que intercambiar claves secretas de manera segura: solo requieren la clave publica del otro.

Por ejemplo, Bob podría publicar pk_B en un directorio publico, tipo la guía telefónica. Alicia bajaría pk_B , y mandaría a Bob

- $c \leftarrow \text{Enc}(pk_B, \text{"De Alicia: esta es mi clave publica: } pk_A\text{"})$, o
- $c \leftarrow \text{Enc}(pk_B, \text{"De Alicia: usemos esta clave secreta: } k\text{"})$,

para poder intercambiar mensajes.

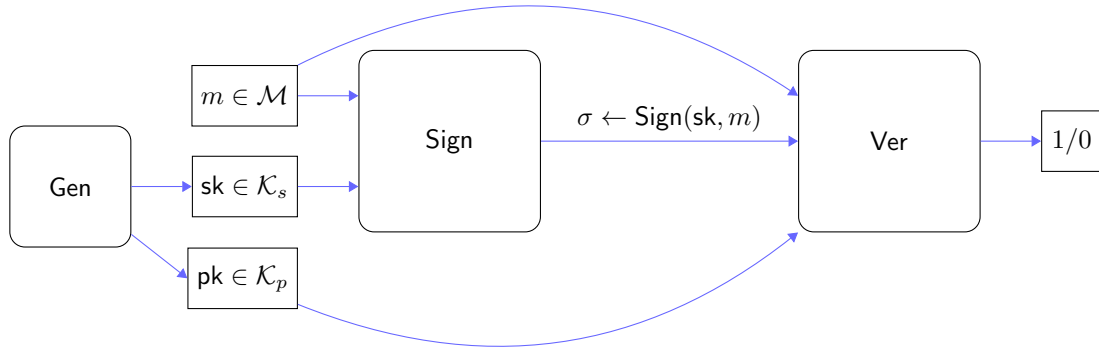
PREGUNTA: Falta algo?

- Como hace Bob para saber que fue justamente Alicia a mandarle el mensaje?
- Como hace Alicia para saber que efectivamente pk_B le pertenece a Bob?

Para resolver este problema, necesitamos dos componentes mas: *firmas digitales*, y una *infraestructura de claves publicas*.

10 Firmas digitales

Las firmas digitales sirven para comprobar la autenticidad de un mensaje m , sin requerir una clave secreta compartida. Pueden pensarlo como una versión publica de los MAC, y por eso sintaxis y noción de seguridad son parecidas.



Las firmas pueden solo ser generadas con sk , y pueden solo ser verificadas con pk .

Definición 42 (DSS). Sean $\mathcal{K}_p$, $\mathcal{K}_s$, $\mathcal{M}$, Σ conjuntos. Sea $\Pi = (\text{Gen}, \text{Sign}, \text{Ver})$ una tripla de algoritmos eficientes, tales que

- Gen es un algoritmo aleatorio, $\text{Gen}: \emptyset \rightarrow \mathcal{K}_s \times \mathcal{K}_p$ que retorna una tupla (sk, pk) ;
- Sign es un algoritmo posiblemente aleatorio, $\text{Sign}: \mathcal{K}_s \times \mathcal{M} \rightarrow \Sigma$ que retorna una firma σ ;
- Ver es un algoritmo determinista $\text{Ver}: \mathcal{K}_p \times \mathcal{M} \times \Sigma \rightarrow \{0, 1\}$, que retorna 1 si la firma es “válida” y 0 si no.

Decimos que Π es un esquema de firma digital (DSS, “digital signature scheme”) δ -completo si

$$\forall m \in \mathcal{M}, \quad \Pr_{\substack{(sk, pk) \sim \text{Gen}(), \\ \$ \text{ en Sign}}} [\text{Ver}(pk, m, \text{Sign}(sk, m; \$)) \neq 1] \leq \delta.$$

Comentario 64. Como en el caso de PKE, el espacio de mensajes “firmables” puede depender de pk , $\mathcal{M}_{pk} \subset \mathcal{M}$.

Si recuerdan, un MAC seguro se dice “fuertemente infalsificable”, y excluye que $\mathcal{A}$ pueda retornar una tupla (m^*, σ^*) validos y “nuevos”. Vamos a usar una definición parecida para los DSS.

Definición 43 (SUF-CMA). Sea $\Pi = (\text{Gen}, \text{Sign}, \text{Ver})$ un DSS. Definimos un experimento Exp^{SUF} .

$\text{Exp}^{\text{SUF}}(\mathcal{A})$	$S(m)$
1: $(\text{sk}, \text{pk}) \leftarrow \text{Gen}()$	1: $\sigma \leftarrow \text{Sign}(\text{sk}, m)$
2: $Q \leftarrow \{\}$	2: $Q \leftarrow Q \cup \{(m, \sigma)\}$
3: $(m^*, \sigma^*) \leftarrow \mathcal{A}^S(\text{pk})$	3: return σ
4: $b \leftarrow \llbracket \text{Ver}(\text{pk}, m^*, \sigma^*) = 1 \wedge (m^*, \sigma^*) \notin Q \rrbracket$	
5: return b	

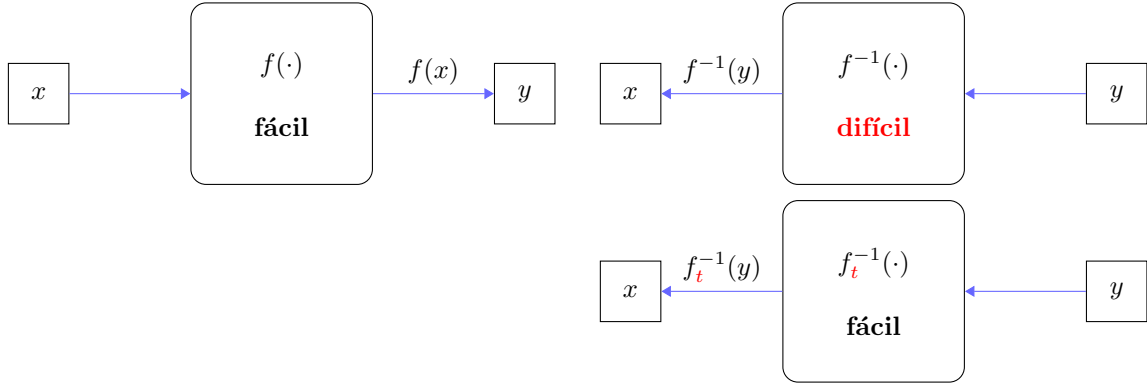
Decimos que Π es (ε, t, q) -fuertemente infalsificable bajo ataques de mensaje elegido (SUF-CMA, “strongly unforgeable under chosen-message attacks”) si cualquier $\mathcal{A}$ que corre en tiempo $\leq t$, realiza $\leq q$ queries a S , tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{SUF}}, \mathcal{A}) := \Pr[\text{Exp}^{\text{SUF}}(\mathcal{A}) \Rightarrow 1] \leq \varepsilon.$$

Comentario 65. En la definición de seguridad MAC, se podía discutir si darle a $\mathcal{A}$ acceso a un oráculo $\text{Ver}(k, \cdot)$. Esta discusión no existe con DSS, dado que $\text{Ver}(\text{pk}, \cdot)$ puede ser evaluada públicamente.

Existen principalmente cuatro maneras de realizar DSS: con protocolos de identificación, con funciones de puerta trasera, con funciones de hash y con “MPC-in-the-head”. Nosotros vamos a cubrir el segundo método, en su forma clásica a través *permutaciones de puerta trasera*, que es comúnmente utilizado en practica.

La idea es la siguiente: dado un conjunto D finito, $f: D \rightarrow D$ es una permutación (o biyección) públicamente computable, tal que $f^{-1}: D \rightarrow D$ es difícil de computar, a menos de no conocer una “puerta trasera”.



Para firmar un mensaje, podríamos usar f como clave publica, t como clave secreta, y generar firmas σ de mensajes *invirtiendo* f , y calculando $f(\sigma)$ como verificación.

Definición 44 (TDP). Sean $\mathcal{T}, \mathcal{F}$ dos conjuntos. Sea $\Pi = (\text{Gen}, \text{Eval}, \text{Invert})$ una tripla de algoritmos, tales que

- **Gen** es un algoritmo aleatorio, $\text{Gen}: \emptyset \rightarrow \mathcal{T} \times \mathcal{F}$ que retorna una tupla (t, f) , con $f: \mathcal{D}_f \rightarrow \mathcal{D}_f$ una permutación sobre un conjunto finito $\mathcal{D}_f$, que puede depender de f .
- **Eval** es un algoritmo determinista, $\text{Eval}: \mathcal{F} \times \mathcal{D}_f \rightarrow \mathcal{D}_f$ que dado (f, x) retorna $f(x)$;
- **Invert** es un algoritmo determinista $\text{Invert}: \mathcal{T} \times \mathcal{F} \times \mathcal{D}_f \rightarrow \mathcal{D}_f$, que dados (t, f, y) retorna $f^{-1}(y)$.

Decimos que Π es una permutación de puerta trasera (TDP, “trapdoor permutation”).

Opcionalmente, si $(\mathcal{D}_f, \times)$ es un grupo, $f: \mathcal{D}_f \rightarrow \mathcal{D}_f$ puede ser un isomorfismo de grupo, o sea dados $g, h \in \mathcal{D}_f$, $f(g \times h) = f(g) \times f(h)$.

La dificultad de invertir una TDP se la puede “capturar” con una definición de seguridad.

Definición 45 (OW TDP). Sean $\Pi = (\text{Gen}, \text{Eval}, \text{Invert})$ una TDP. Definimos Exp^{OW} ,

$\text{Exp}^{\text{OW}}(\mathcal{A})$
1: $(t, f) \leftarrow \text{Gen}()$
2: $y \xleftarrow{\$} \mathcal{D}_f$
3: $x \leftarrow \mathcal{A}(f, y)$
4: return $\llbracket f(x) = y \rrbracket$

Decimos que Π es (ε, t) -unidireccional (OW, “one-way”), si cualquier $\mathcal{A}$ que corre en tiempo $\leq t$ tiene ventaja

$$\text{Adv}(\text{Exp}^{\text{OW}}, \mathcal{A}) := \Pr[\text{Exp}^{\text{OW}}(\mathcal{A}) \Rightarrow 1] \leq \varepsilon.$$

Comentario 66. Notamos que el experimento requiere $y \sim U(\mathcal{D}_f)$, y requiere que a $\mathcal{A}$ no le sean dados oráculos de inversión! Esto puede volver las demostraciones mas difíciles. Esta omisión es necesaria si f es un isomorfismo de grupos: de otro modo, dados $f^{-1}(y_1), f^{-1}(y_2)$, $\mathcal{A}$ podría calcular la inversa $f^{-1}(y_1) \times f^{-1}(y_2) = f^{-1}(y_1 \times y_2)$ de $y_1 \times y_2$, sin pedirselo al oráculo.

Esto nos trae a nuestro ultimo postulado.

Postulado 4. Dado $\lambda > 0$, existen TDPs que otorgan λ -bits de seguridad OW y que son isomorfismos de grupo.

Clase 20: firmas “full-domain hash”

Fernando Virdia, versión: 0.1, mayo 2025

Vamos ahora a definir un DSS que utiliza TDPs y funciones de hash para firmar $m \in \{0,1\}^*$.

Definición 46 (TDP-FDH, o “hash-then-invert”). Sea $\Pi = (\text{Gen}, \text{Eval}, \text{Invert})$ una TDP, y sea $\{H_f\}_f$ una familia de funciones de hash $H_f := \{0,1\}^* \rightarrow \mathcal{D}_f$, donde $f: \mathcal{D}_f \rightarrow \mathcal{D}_f$. Definimos el DSS $\Pi' = (\text{Gen}', \text{Sign}', \text{Ver}')$ de hash de dominio completo (FDH, “full-domain hash” o “invert-then-sign”),

$\text{Gen}'()$	$\text{Sign}'(\text{sk} = (t, f), m)$	$\text{Ver}'(\text{pk} = f, \sigma)$
1: $(t, f) \leftarrow \text{Gen}()$	1: $h \leftarrow H_f(m)$	1: $h \leftarrow H_f(m)$
2: $\text{sk} \leftarrow (t, f)$	2: $\sigma \leftarrow \text{Invert}(t, f, h)$	2: $h' \leftarrow \text{Eval}(f, \sigma)$
3: $\text{pk} \leftarrow f$	3: return σ	3: return $\llbracket h' = h \rrbracket$
4: return (sk, pk)		

Comentario 67. Mas allá del tipo de primitiva usado, la mayoría de los DSS efectivamente firman hashes de un mensaje, no el mensaje directamente (“hash-then-sign”). “Deshacer” la estructura algebraica de m puede ser útil para obtener seguridad. Por ejemplo, esto es necesario si se usa la famosa TDP de Rivest, Shamir y Adleman (RSA).

Comentario 68. Usamos $m \in \{0,1\}^*$, pero uno podría definir TDP-FDH sobre cualquier dominio S tal que $H: S \rightarrow \mathcal{D}_f$ es una función de hash.

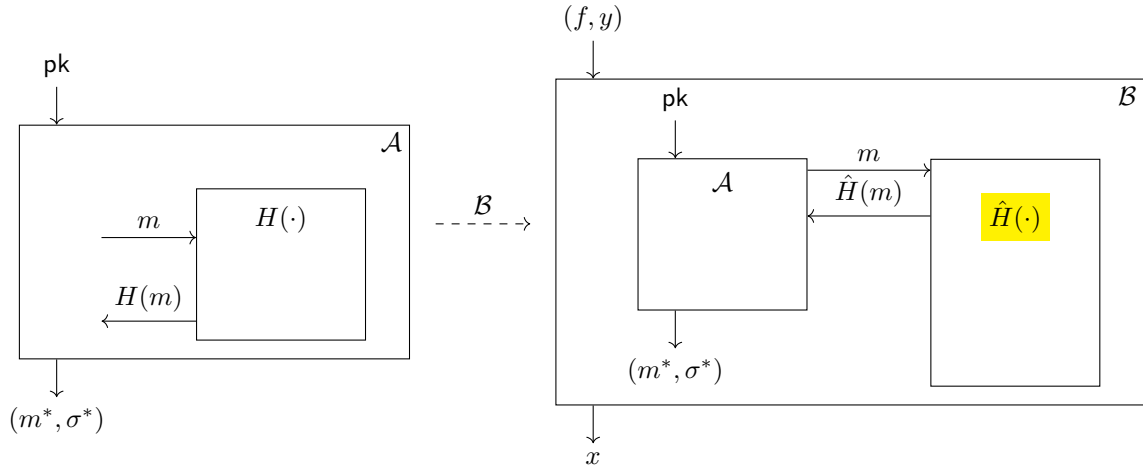
Comentario 69. Escribimos H_f , pero generalmente la función de hash es predeterminada públicamente de manera de valer para diferentes conjuntos $\mathcal{D}_f$.

Comentario 70. Como habrán notado, el esquema es muy parecido a usar una PRF para realizar un MAC, reemplazando la PRF (o PRP!) con la TDP.

Finalmente, demostramos que una TDP-FDH otorga SUF-CMA.

Teorema 8 (OW TDP + ROM $\Rightarrow$ SUF-CMA). Sea Π una TDP que es también un isomorfismo de grupos, y sea Π' el resultante DSS obtenido como TDP-FDH. Supongamos que las funciones de hash $\{H_f\}_f$ son completamente aleatorias (o sea, trabajamos con el modelo del oráculo aleatorio). Sea $q_S \geq 1$ el número de queries al oráculo de firmas $S(\cdot)$ que le permitimos a un adversario $\mathcal{A}$ contra Π' . Si Π es (ε, t) -OW entonces Π' es (ε', t', q_S) -SUF-CMA con $t' \approx t$, $\varepsilon' \lesssim e \cdot q_S \cdot \varepsilon$, donde $e = 2,71828\dots$ es la constante de Napier.

Proof. Sea $\mathcal{A}$ un adversario contra TDP-FDH. Sea $p \in (0,1)$ una constante, el cual valor determinaremos mas tarde. Vamos a definir $\mathcal{B}$, un adversario que juega Exp^{OW} contra la TDP. Sin pérdida de generalidad, escribimos H en lugar de H_f . Dado que suponemos la función H ser completamente aleatoria, vamos a suponer que cuando $\mathcal{A}$ requiere $H(m)$, calcula este valor a través de un oráculo, que nosotros vamos a reemplazar.



Esto puede parecer raro, pero puede ser pensado de la siguiente manera. Supongamos que $\mathcal{A}$ es un programa en C, y precisa calcular un hash $H(m)$. Posiblemente, $\mathcal{A}$ obtenga H a través de una librería de enlace dinámico (una DLL) contra la cual viene compilado, dado que es improbable que la estructura de la función de hash tenga un impacto sobre la dificultad de falsificar una firma (es común suponer que los hashes son “collision resistant”).

Obtenido $\mathcal{A}$, el adversario $\mathcal{B}$ decide de re-compilarla contra una diferente DLL, que implementa otra función aleatoria $\hat{H}(\cdot)$. La intuición es que esto no tendría que reducir la probabilidad de $\mathcal{A}$ de retornar una falsificación (m^*, σ^*) respecto al DSS TDP-FDH con hash $\hat{H}$, dado que $\mathcal{A}$ es un algoritmo aleatorio, y H es una función aleatoria, por lo que la distribución de la variable aleatoria $\mathcal{A}^{\hat{H}}(f, y)$ es la misma que la de $\mathcal{A}^H(f, y)$.

Decimos que este tipo de demostración se encuentra en el *modelo del oráculo aleatorio* (ROM, “random oracle model”). Mucho ha sido debatido sobre el ROM. [KM15] cubre los aspectos mas importantes del debate. A menudo este modelo nos permite demostrar la seguridad de esquemas mas eficientes [BR93], que resisten la “prueba del tiempo” una ves desplegados. Por ejemplo, la demostración de que FO es KEM OT-IND-CCA, también se encuentra en el ROM.

$\mathcal{B}(f, y)$	$\hat{H}(m)$	$S(m)$
1: $p \leftarrow //$ un valor particular $\in (0, 1)$	1: if $m \in L$:	1: $\hat{H}(m)$
2: $Q \leftarrow \{\}$	2: $(\dots, h) \leftarrow L[m]$	2: $(b, s, h) \leftarrow L[m]$
3: $L \leftarrow []$	3: return h	3: if $b = 1$:
4: $\text{pk} \leftarrow f$	4: $//$ si no, $m \notin L$:	4: ERROR
5: $(m^*, \sigma^*) \leftarrow \mathcal{A}^{S(\cdot), \hat{H}(\cdot)}(\text{pk})$	5: $r \xleftarrow{\$} (0, 1)$	5: $Q \leftarrow Q \cup \{(m, s)\}$
6: $\hat{H}(m^*) \parallel \Rightarrow m^* \in L$	6: $s \xleftarrow{\$} \mathcal{D}_f$	6: return s
7: $(b^*, s^*, h^*) \leftarrow L[m^*]$	7: if $r < p$: $//$ con pr. p	
8: $x \leftarrow \sigma^* \times (s^*)^{-1}$	8: $h \leftarrow \text{Eval}(f, s)$	
9: return x	9: $L[m] \leftarrow (0, s, h)$	
	10: else : $//$ con pr. $1 - p$	
	11: $h \leftarrow y \times \text{Eval}(f, s)$	
	12: $L[m] \leftarrow (1, s, h)$	
	13: return h	

La idea de la demostración es que para falsificar una firma, $\mathcal{A}$ tiene que invertir f sobre algun input $H(m^*)$. Dado Π' , un adversario $\mathcal{A}$ que juega Exp^{SUF} tiene acceso a un oráculo $H(\cdot)$, que vamos a remplazar con otro oráculo $\hat{H}$ de manera de “inyectar” un valor $y \in D_f$ que $\mathcal{B}$ deba invertir, en el output de $\hat{H}$.

Primero analizemos la distribución de $\hat{H}$, desde el punto de vista de $\mathcal{A}$ “dentro” de $\mathcal{B}$. Al calcular $\hat{H}(m)$, dos cosas pueden pasar:

- Si $r < p$, generamos $s \sim U(\mathcal{D}_f)$, y retornamos $h := f(s)$. Dado que f es una permutación, $h = f(s)$ también está uniformemente distribuida, $h \sim U(D_f)$.

$$\left(\forall y \in \mathcal{D}_f, \Pr_{\sigma \sim U(D)} [f(\sigma) = y] = \Pr_{\sigma \sim U(D)} [\sigma = f^{-1}(y)] = \frac{1}{|\mathcal{D}_f|} \right)$$

Por lo tanto, $\hat{H}(m) \mid r < p \sim U(D_f)$.

- Si $r \geq p$, queremos “inyectar” y , generado en Exp^{OW} , dentro de la salida $\hat{H}(m)$. Generamos $s \sim U(D_f)$, tal que $f(s) \sim U(D_f)$ también, y retornamos $y \times f(s) \sim U(D_f)$. Por lo tanto $\hat{H}(m) \mid r \geq p \sim U(D_f)$.

En ambos casos, $\hat{H}(m) \sim U(D_f)$, así como lo seria $H(m)$. En el ROM, esta técnica se llama “programación del oráculo aleatorio”.

Dentro de Exp^{SUF} , $\mathcal{A}$ tambien tiene acceso a un oráculo de firma, $S(\cdot)$. En Exp^{SUF} este es realizado usando la clave secreta, $S(m) := \text{return Sign}(\text{sk}, m)$. Al no tener sk , $\mathcal{B}$ utiliza los valores s generados internamente a $\hat{H}$ para responder $S(m)$. Recordamos que en TDP-FDH, $\text{Sign}(\text{sk} = (t, f), m) = f^{-1}(H(m))$. Al calcular $S(m)$, primero forzamos el calculo de $h \leftarrow \hat{H}(m)$. Sea $(b, s, h) \leftarrow L[m]$. Luego,

- Si h fue generado en el ramo “ $r < p$ ” ($b = 0$) de $\hat{H}$, entonces $s = f^{-1}(h)$ a $\mathcal{A}$ por construcción de $\hat{H}$, y retornamos s , que notamos ser $s \sim U(D_f) \sim f^{-1}(H(m))$.
- Si h fue generado en el ramo “ $r \geq p$ ” ($b = 1$), $f^{-1}(h) = f^{-1}(y) \times s$. Al no conocer $f^{-1}(y)$, no podemos *simular* el algoritmo de firma. $\mathcal{B}$ interrumpe su ejecución retornando un **ERROR** y fallando en el ataque. Denotamos este evento, E , y el evento que no ocurra un error, $\bar{E}$.

Finalmente, observamos que en cuanto $\mathcal{A}$ retorna (m^*, σ^*) , $\mathcal{B}$ calcula $\hat{H}(m^*)$, de manera de asegurarse que $m^* \in L$, lo que simplifica el siguiente analisis. Sean $(b^*, s^*, h^*) \leftarrow L[m^*]$. Supongamos que al calcular $\hat{H}(m^*)$, $b^* \leftarrow 1$, o sea $h^* \leftarrow y \times f(s^*)$. Si $\mathcal{A}$ logra retornar una falsificación valida, entonces $\sigma^* = f^{-1}(h^*) = f^{-1}(y \times f(s^*)) = f^{-1}(y) \times s^*$, y $\mathcal{B}$ retorna $x \leftarrow \sigma^* \times s^{*-1} = f^{-1}(y) \times s^* \times s^{*-1} = f^{-1}(y)$, ganando Exp^{OW} .

Para poder asegurarnos que $\mathcal{A}$ logre falsificar una firma, debemos asegurarnos que su entorno dentro de $\mathcal{B}$ sea indistinguible de el de Exp^{SUF} . Hemos visto que la distribucion de $\hat{H}(\cdot)$ es la correcta, y asi es la de $S(\cdot)$, siempre que no ocurra un **ERROR**. Por lo tanto, el entorno dentro de $\mathcal{B}$ será correcto si no ocurre un error, o sea $\bar{E}$. Antes de evaluar $S(m)$, viendo $\hat{H}(m)$, $\mathcal{A}$ no tiene ninguna información sobre si m va a causar un **ERROR** de manera más o menos probable respecto a algun otro $m' \neq m$. Por lo tanto del punto de vista de $\mathcal{A}$, $\Pr[S(m) \Rightarrow \text{ERROR}] = 1 - p$. Suponiendo que $\mathcal{A}$ usa sus q_S queries a S sobre al máximo q_S inputs diferentes, y siendo el valor de $S(m)$ fijo, $\Pr[\bar{E}] \geq p^{q_S}$.

Podemos entonces calcular la ventaja de $\mathcal{B}$ en base a la de $\mathcal{A}$ y a $\Pr[\bar{E}]$. Sean $(b^*, s^*, h^*) \leftarrow L[m^*]$.

$$\begin{aligned}
\text{Adv}(\text{Exp}^{\text{OW}}, \mathcal{B}) &= \Pr[\text{Exp}^{\text{OW}}(\mathcal{B}) \Rightarrow 1] = \Pr[f(\sigma^* \times s^{*-1}) = y] = \Pr[f(\sigma^*) = y \times f(s^*)] \\
&= \Pr[f(\sigma^*) = \hat{H}(m^*)] \\
&\geq \Pr[f(\sigma^*) = \hat{H}(m^*) \wedge (m^*, \sigma^*) \notin Q \wedge b^* = 1 \wedge \bar{E}] \\
&= \Pr[f(\sigma^*) = \hat{H}(m^*) \wedge (m^*, \sigma^*) \notin Q \wedge b^* = 1 \mid \bar{E}] \cdot \Pr[\bar{E}] \\
&= \Pr[\text{Exp}^{\text{SUF}}(\mathcal{A}^{\hat{H}(\cdot), \text{Sign}(\text{sk}, \cdot)}) \Rightarrow 1 \wedge b^* = 1] \cdot \Pr[\bar{E}] \\
&= \Pr[\text{Exp}^{\text{SUF}}(\mathcal{A}^{H(\cdot), \text{Sign}(\text{sk}, \cdot)}) \Rightarrow 1] \cdot \Pr[b^* = 1] \cdot \Pr[\bar{E}] \\
&\geq \text{Adv}(\text{Exp}^{\text{SUF}}, \mathcal{A}) \cdot (1 - p) \cdot p^{q_S} \\
\iff \text{Adv}(\text{Exp}^{\text{SUF}}, \mathcal{A}) &\leq \frac{\text{Adv}(\text{Exp}^{\text{OW}}, \mathcal{B})}{p^{q_S} \cdot (1 - p)}.
\end{aligned}$$

Al ser Π una (ε, t) -OW TDP, obtenemos $\varepsilon' \leq \varepsilon / (p^{q_S} \cdot (1 - p))$.

Ahora recordamos que $p \in (0, 1)$ es una constante de valor arbitrario, y notamos que $(p^{q_S} \cdot (1 - p)) \in (0, 1)$. Para obtener la mejor cota superior de $\text{Adv}(\text{Exp}^{\text{SUF}}, \mathcal{A})$ (o sea, la cota superior mas baja posible), queremos elegir el valor $p = p_0$ que maximiza el denominador, $g(p) := p^{q_S} \cdot (1 - p)$, con $p \in (0, 1)$.

1. Calculamos la derivada:

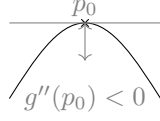
$$\begin{aligned}
g'(p) &= \frac{d}{dp} [p^{q_S}] (1 - p) + p^{q_S} \frac{d}{dp} [1 - p] \\
&= q_S \cdot p^{q_S-1} (1 - p) + p^{q_S} \cdot (-1) \\
&= p^{q_S-1} (q_S - q_S \cdot p - p) \\
g'(p) = 0 &\iff q_S - q_S \cdot p - p = q_S - p(q_S + 1) = 0 \iff p = \frac{q_S + 1 - 1}{q_S + 1} = 1 - \frac{1}{q_S + 1} =: p_0.
\end{aligned}$$

2. Verificamos que es un máximo:

$$\begin{aligned}
g''(p) &= \frac{d}{dp} [p^{q_S-1}] (q_S - q_S \cdot p - p) + p^{q_S-1} \frac{d}{dp} [q_S - q_S \cdot p - p] \\
&= (q_S - 1) \cdot p^{q_S-2} (q_S - q_S \cdot p - p) + p^{q_S-1} (-q_S - 1) \\
&= p^{q_S-2} ((q_S - 1) \cdot (q_S - q_S \cdot p - p) - p \cdot (q_S + 1)) \\
&= p^{q_S-2} (q_S \cdot (q_S - q_S \cdot p - p) - (q_S - q_S \cdot p - p) - p \cdot (q_S + 1)) \\
&= p^{q_S-2} (q_S^2 - q_S^2 \cdot p - q_S \cdot p - q_S + q_S \cdot p + p - p \cdot q_S - p) \\
&= p^{q_S-2} (q_S^2 - q_S^2 \cdot p - q_S \cdot p - q_S + \cancel{q_S \cdot p} + \cancel{p} - \cancel{p \cdot q_S} - \cancel{p})
\end{aligned}$$

$$\begin{aligned}
&= p^{q_S-2} \cdot q_S \cdot (q_S - q_S \cdot p - p - 1) \\
\implies g''(p_0) &= p_0^{q_S-2} \cdot q_S \cdot (\cancel{q_S - q_S \cdot p - p} - 1) = -q_S \cdot p_0^{q_S-2} < 0.
\end{aligned}$$

Dado que $g''(p_0)$ es negativo, $p_0 = 1 - \frac{1}{q_S+1}$ es un máximo de g (el único, dado $g'(p)$).



Ahora calculamos el denominador en p_0 ,

$$\begin{aligned}
g(p_0) &= \left(1 - \frac{1}{q_S+1}\right)^{q_S} \cdot \left(1 - \left(1 - \frac{1}{q_S+1}\right)\right) \\
&= \frac{\left(1 - \frac{1}{q_S+1}\right)^{q_S+1}}{\left(1 - \frac{1}{q_S+1}\right)} \cdot \frac{1}{q_S+1} = \frac{\left(1 - \frac{1}{q_S+1}\right)^{q_S+1}}{\left(\frac{q_S+1}{q_S+1} - \frac{1}{q_S+1}\right)} \cdot \frac{1}{q_S+1} \\
&= \left(1 - \frac{1}{q_S+1}\right)^{q_S+1} \cdot \frac{1}{q_S}
\end{aligned}$$

Dado que $\lim_{n \rightarrow \infty} \left(1 - \frac{1}{n}\right)^n = e^{-1}$ y que $q_S \gg 1$,

$$g(p_0) \approx \frac{1}{e} \cdot \frac{1}{q_S}.$$

Por lo tanto, si dentro de $\mathcal{B}$ asignamos $p \leftarrow p_0 = 1 - \frac{1}{q_S+1}$,

$$\text{Adv}(\text{Exp}^{\text{SUF}}, \mathcal{A}) \leq \frac{\text{Adv}(\text{Exp}^{\text{OW}}, \mathcal{B})}{g(p_0)} \lesssim e \cdot q_S \cdot \text{Adv}(\text{Exp}^{\text{OW}}, \mathcal{B}),$$

y $t' \approx t$, $\varepsilon' \lesssim e \cdot q_S \cdot \varepsilon$. □

Comentario 71. Notamos que las propiedades de H como función de hash (collision-resistance, etc) no juegan un rol directo en la demostración. Esto es porque ignoramos ser H un hash, y la consideramos una oráculo de función aleatoria. De manera sutil, decidimos suponer que los ataques no van a ser causados por un adversario $\mathcal{A}$ que invierta o ataque la estructura interna de H , y como esta posiblemente interactúe con la estructura de la TDP.

Clase 21: infraestructuras de claves publicas

Fernando Virdia, versión: 0.1, mayo 2025

Esta clase se encuentra en forma de diapositivas en <https://apuntes.indcpa.com>.

References

- [BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: a paradigm for designing efficient protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security, CCS '93*, page 62–73, New York, NY, USA, 1993. Association for Computing Machinery.
- [BS23] Dan Boneh and Victor Shoup. *A Graduate Course in Applied Cryptography*. Self-published, January 2023. Version 0.6, <https://toc.cryptobook.us>.
- [FT21] Pooya Farshim and Stefano Tessaro. Password hashing and preprocessing. In *Advances in Cryptology – EUROCRYPT 2021: 40th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Zagreb, Croatia, October 17–21, 2021, Proceedings, Part II*, page 64–91, Berlin, Heidelberg, 2021. Springer-Verlag.
- [HHK17] Dennis Hofheinz, Kathrin Hövelmanns, and Eike Kiltz. A modular analysis of the fujisaki-okamoto transformation. In *Theory of Cryptography Conference*, pages 341–371. Springer, 2017.
- [HHMS25] Kathrin Hövelmanns, Andreas Hülsing, Christian Majenz, and Fabrizio Sisinni. (un)breakable curses - re-encryption in the fujisaki-okamoto transform. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025*, pages 245–274, Cham, 2025. Springer Nature Switzerland.
- [KL20] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography*. Chapman and Hall/CRC, December 2020.
- [KM15] Neal Koblitz and Alfred J Menezes. The random oracle model: a twenty-year retrospective. *Designs, Codes and Cryptography*, 77:587–610, 2015.
- [Ros21] Mike Rosulek. *The Joy of Cryptography*. Self-published, January 2021. <https://joyofcryptography.com>.