

IIC3253

Funciones de hash

Funciones de hash criptográficas

Función de un espacio de posibles mensajes a un espacio de mensajes de largo fijo:

$$h : \mathcal{M} \rightarrow \mathcal{H}$$

\mathcal{M} es el espacio de mensajes y \mathcal{H} es el espacio de posibles valores de la función de hash

- Por ejemplo, $\mathcal{M} = \{0, 1\}^*$ y $\mathcal{H} = \{0, 1\}^{128}$
- Decimos que $h(m)$ es el hash de un mensaje m

Una propiedad básica de las funciones de hash

Debe existir un algoritmo eficiente que, dado $m \in \mathcal{M}$, calcula $h(m)$

Una primera propiedad fundamental de las funciones de hash

No debe existir un algoritmo eficiente que, dado $x \in \mathcal{H}$, encuentra $m \in \mathcal{M}$ tal que $h(m) = x$

Esta propiedad se denota como **ser resistente a preimagen**

¿Por qué insistimos en el adjetivo "criptográficas"?

Considere la siguiente función de hash:

$$h(m) = (A \cdot m + B) \bmod C$$

Suponemos que los mensajes son números naturales

- A , B y C son constantes, C es un número primo

¿Es esta función resistente a preimagen?

La función anterior no es resistente a preimagen

Considere $h(m) = (13 \cdot m + 97) \bmod 641$

Suponga que tiene el valor de hash 200. ¿Puede encontrar un mensaje m tal que $h(m) = 200$?

Una combinación de herramientas de aritmética modular nos pueden dar una respuesta rápida: 501

$$h(15244) = (13 \cdot 501 + 97) \bmod 641 = 200$$

Una segunda propiedad fundamental de las funciones de hash

No debe existir un algoritmo eficiente que pueda encontrar $m_1, m_2 \in \mathcal{M}$ tales que $m_1 \neq m_2$ y $h(m_1) = h(m_2)$

Esta propiedad se denota como **ser resistente a colisiones**

Las funciones de hash en la práctica

```
1 import hashlib
2
3 if __name__ == "__main__":
4     h1 = hashlib.md5()
5     h2 = hashlib.md5()
6     h3 = hashlib.md5()
7
8     h1.update(b"este es mi primer mensaje")
9     h2.update(b"El objetivo del curso es introducir al alumno a "
10              b"los conceptos fundamentales de criptografia y "
11              b"seguridad computacional, poniendo enfasis tanto en "
12              b"los aspectos formales necesarios para definir")
13     h3.update(b"este es mi prXmer mensaje")
14
15     print(h1.hexdigest())
16     print(h2.hexdigest())
17     print(h3.hexdigest())
```

Output: 30635f74755bfb8c9faeac3ab106c2ab

 c0e1fe34f764e458463c4fd8a91355d0

 2c5956c357577eed8e76608cf40e79ee

Las funciones de hash en la práctica

```
1 import hashlib
2
3 if __name__ == "__main__":
4     h1 = hashlib.sha256()
5     h2 = hashlib.sha256()
6     h3 = hashlib.sha256()
7
8     h1.update(b"este es mi primer mensaje")
9     h2.update(b"El objetivo del curso es introducir al alumno a "
10              b"los conceptos fundamentales de criptografia y "
11              b"seguridad computacional, poniendo enfasis tanto en "
12              b"los aspectos formales necesarios para definir")
13     h3.update(b"este es mi prXmer mensaje")
14
15     print(h1.hexdigest())
16     print(h2.hexdigest())
17     print(h3.hexdigest())
```

Output: 105f0a373501caffc828ce3da6b0b9c7569c68194f1db1057831fa8b3844cc8c
 5a022e6e371bf654c33025642eb147a432f26dc3c3206ec992fc7725799c3868
 5d94508d4fb9a1daeade09995904a281ccbdd165d2dc7a24798fcff9a80c96e7

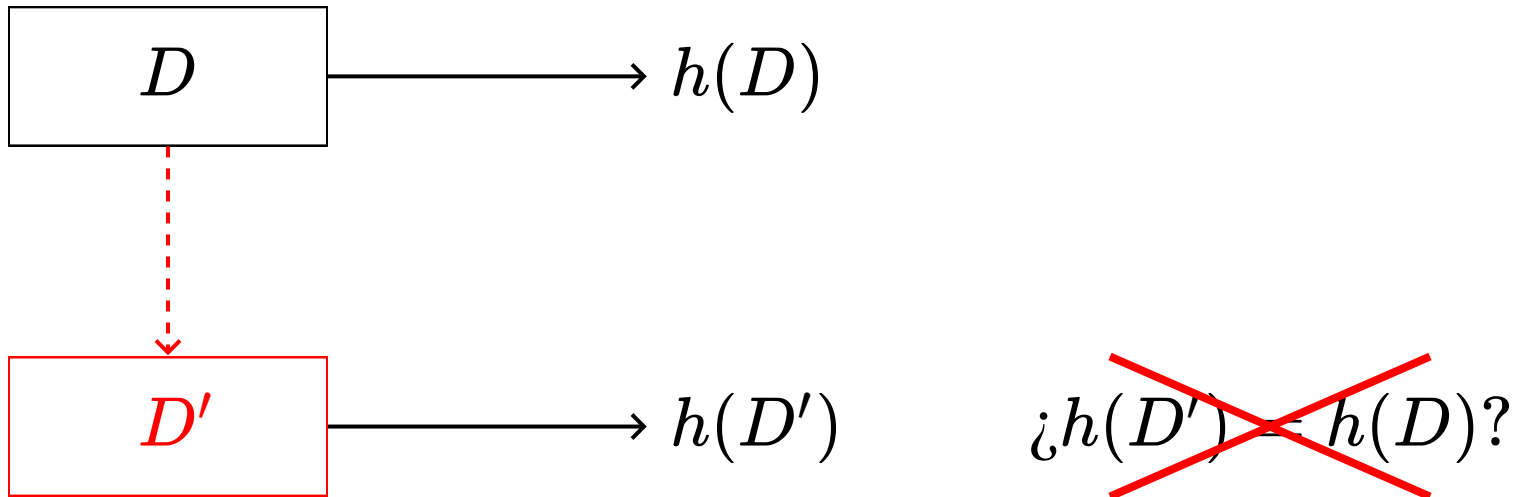
**¿Para qué son usadas
las funciones de hash
criptográficas?**

Aplicaciones de las funciones de hash

Las funciones de hash (criptográficas) tienen muchas aplicaciones prácticas, y nos van a acompañar durante todo el curso

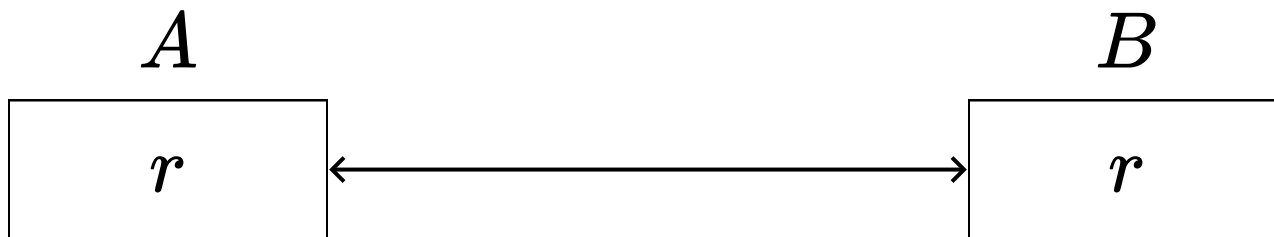
En las siguientes láminas vamos a ver algunas aplicaciones que dan una idea de su utilidad

Una primera aplicación: integridad de un documento

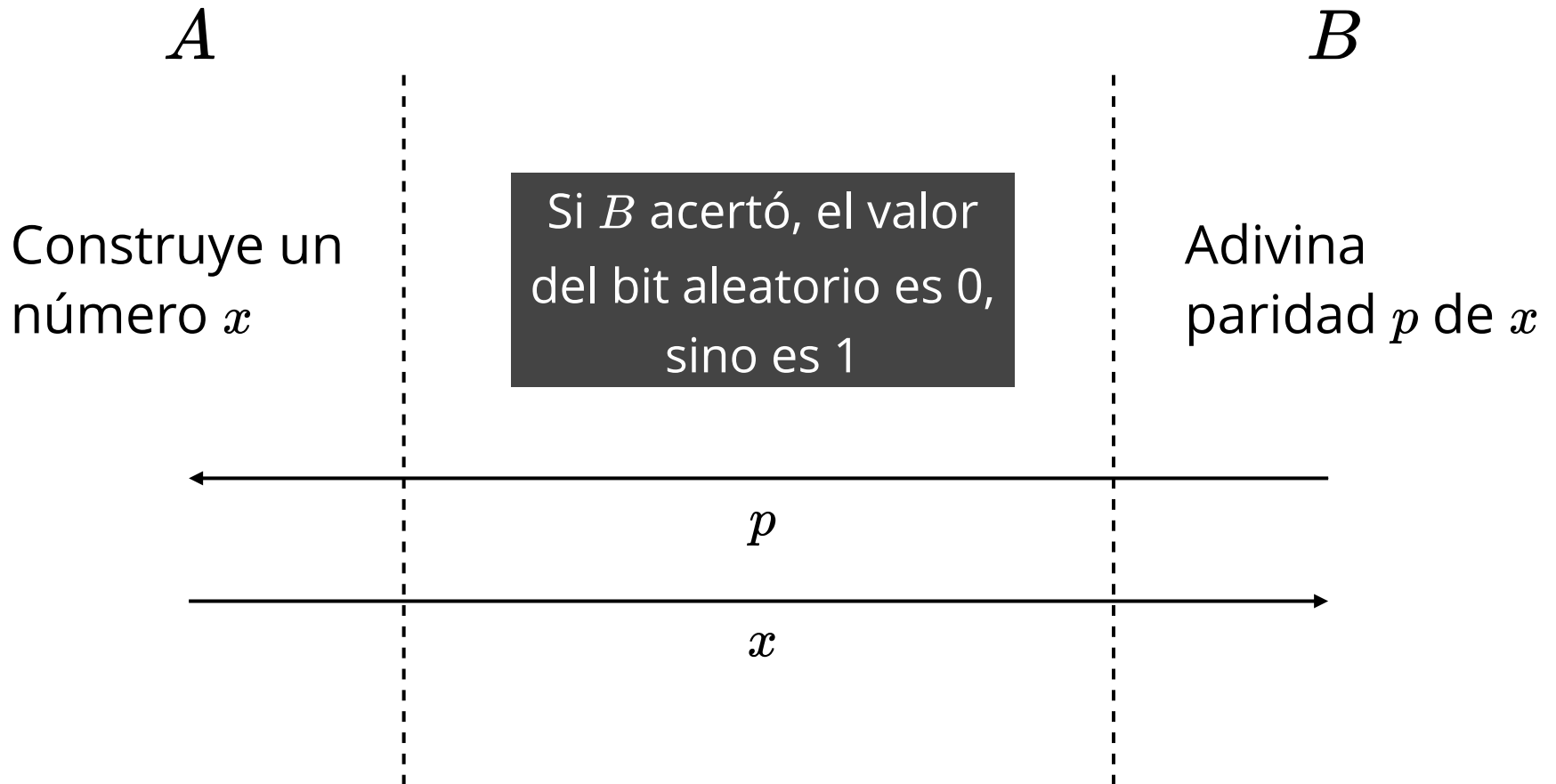


Una segunda aplicación: lanzar una moneda por teléfono

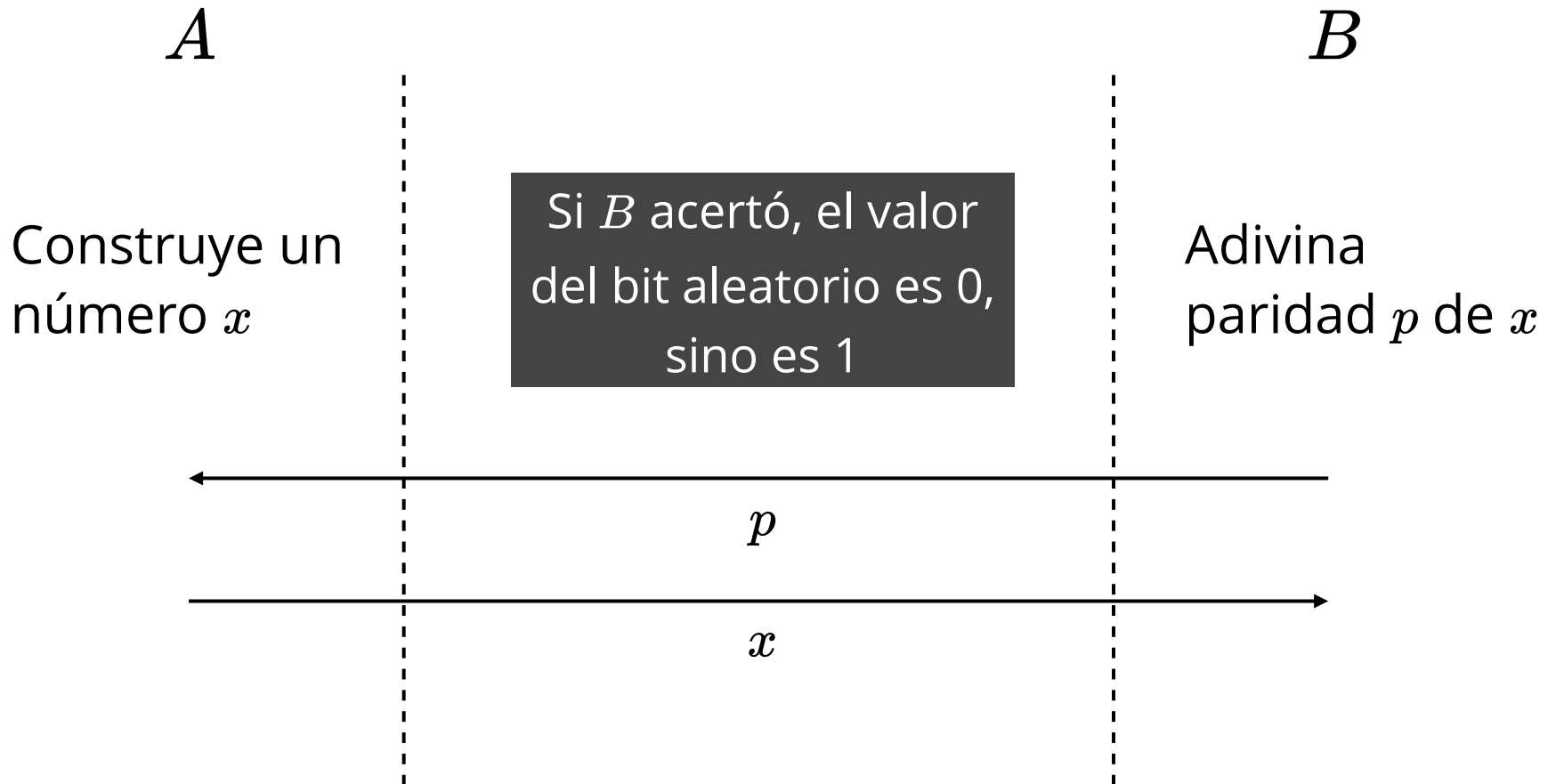
¿Pueden dos usuarios ponerse de acuerdo en
un número aleatorio de manera remota?



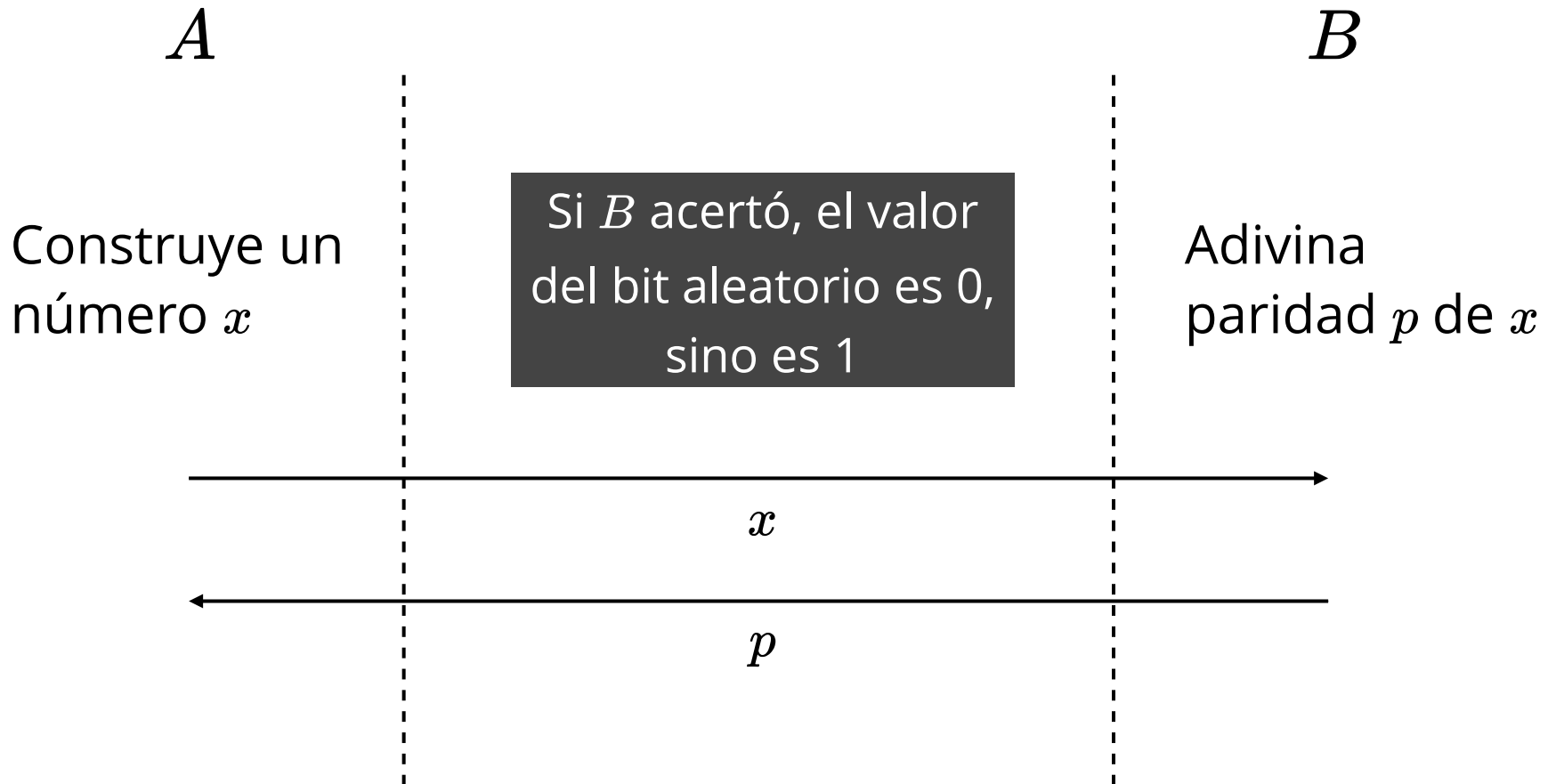
Un primer protocolo



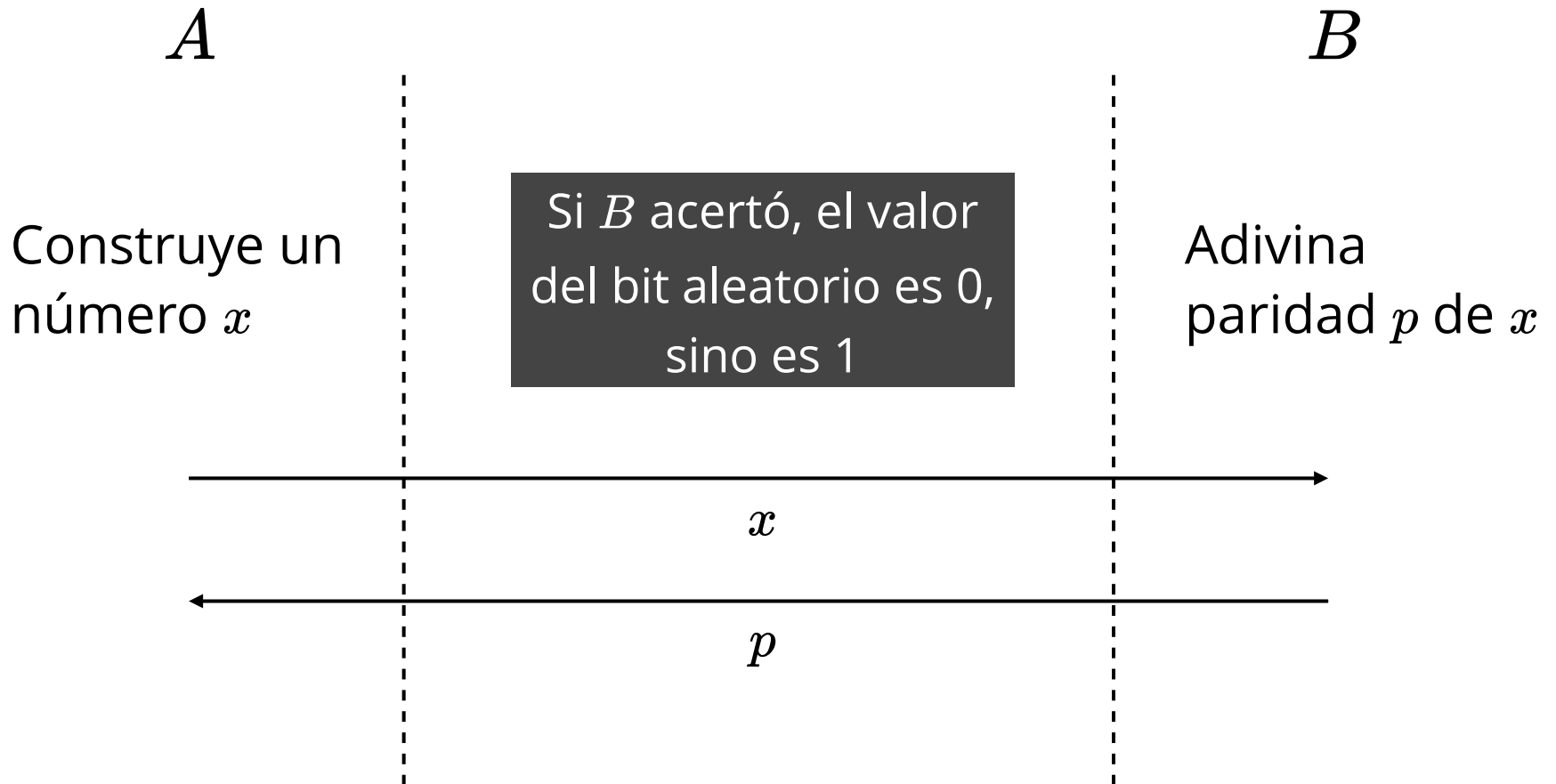
~~Un primer protocolo~~



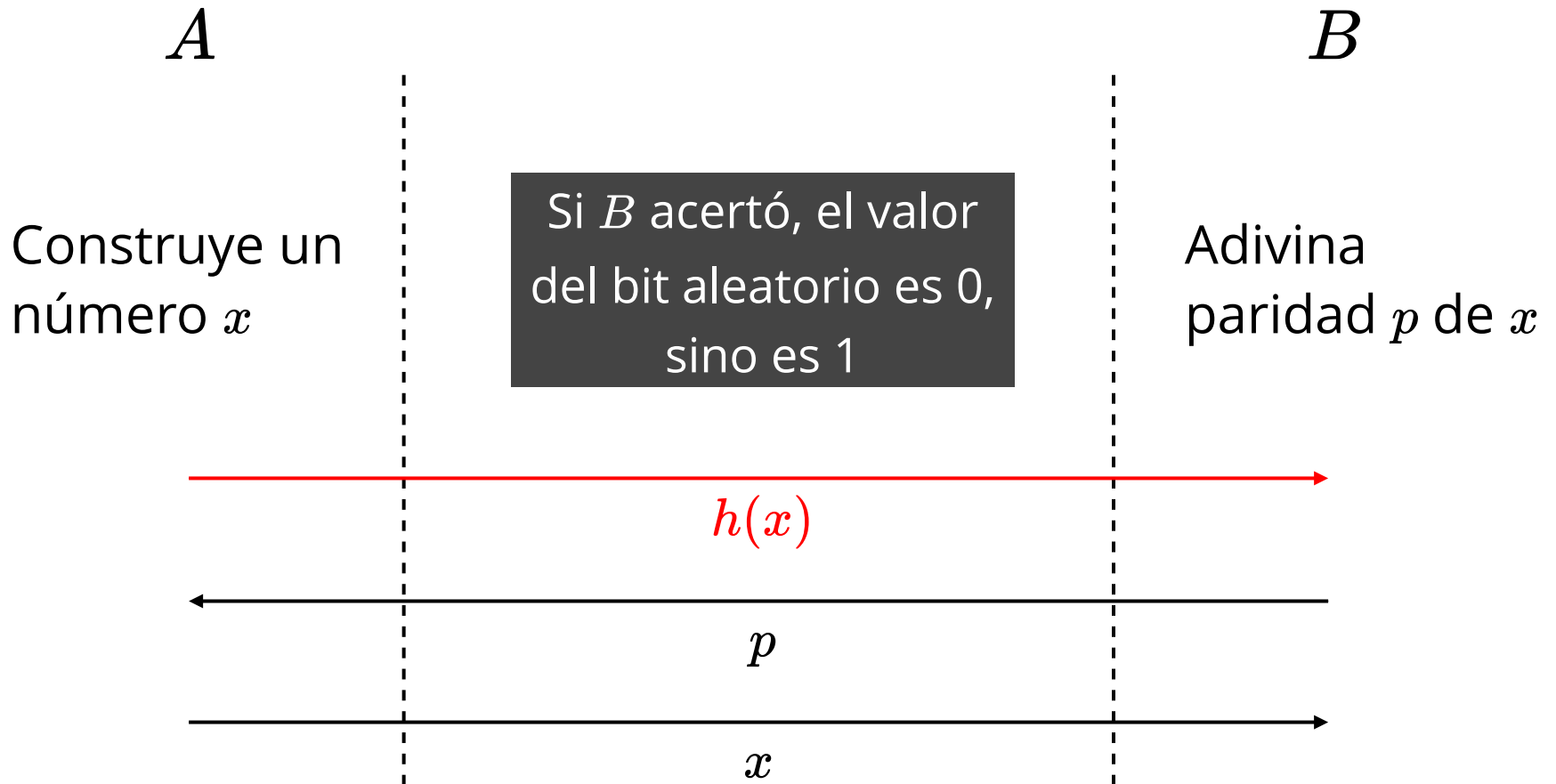
Un segundo protocolo



~~Un segundo protocolo~~



¡Un protocolo correcto!



**¿Cómo se formaliza la noción
de función de hash?**

**¿Por qué es necesario
formalizar esta noción?**

Una definición formal de función de hash

Una función de hash es un par (Gen, h) de algoritmos de tiempo polinomial tales que dado un parámetro de seguridad 1^n :

- Gen toma como entrada 1^n y retorna una llave s
- h toma como entrada s y un mensaje $m \in \{0, 1\}^*$, y retorna un hash $h^s(m) \in \{0, 1\}^{\ell(n)}$, donde ℓ es un polinomio fijo

Una definición formal de función de hash

En una definición más general pedimos que (Gen, h) sea un par de algoritmos aleatorizados de tiempo polinomial

Si $m \in \{0, 1\}^{\ell'(n)}$ para un polinomio fijo ℓ' tal que $\ell'(n) > \ell(n)$, entonces (Gen, h) es una función de hash de largo fijo

Una noción necesaria

Una función $f : \mathbb{N} \rightarrow \mathbb{N}$ es despreciable si:

$$(\forall \text{ polinomio } p : \mathbb{N} \rightarrow \mathbb{N})(\exists n_0 \in \mathbb{N})(\forall n \geq n_0) \left(f(n) < \frac{1}{p(n)} \right)$$

1. Muestre que 2^{-n} y $n^{-\log(n)}$ son funciones despreciables
2. Demuestre que si f y g son funciones despreciables y p es un polinomio, entonces $f + g$ y $f \cdot p$ son funciones despreciables

Formalizando la noción de resistencia a colisiones

Considere una función de hash (Gen, h)

Definimos el juego $Hash-Col(n)$:

1. El verificador genera $s = Gen(1^n)$, y se lo entrega al adversario
2. El adversario elige mensajes m_1 y m_2 con $m_1 \neq m_2$
3. El adversario gana el juego si $h^s(m_1) = h^s(m_2)$, y en caso contrario pierde

Formalizando la noción de resistencia a colisiones

Una función de hash (Gen, h) se dice resistente a colisiones si **para todo adversario que funciona como un algoritmo aleatorizado de tiempo polinomial**, existe una función despreciable $f(n)$ tal que:

$$\Pr(\text{Adversario gane } Hash-Col(n)) \leq f(n)$$

¿Cómo se formaliza la noción de ser resistente a preimagen usando las ideas anteriores?

Dejamos esta definición como ejercicio

Además, dejamos como ejercicio demostrar que ser resistente a colisiones implica ser resistente a preimagen

¿Dónde estamos?

- Estudiamos dos conceptos fundamentales en criptografía: cifrado simétrico y funciones de hash
 - Vimos algunas propiedades teóricas de estos conceptos
- Vamos a estudiar un tercer concepto fundamental: autenticación de mensajes
 - También vamos a ver algunas de sus propiedades teóricas
- Después de esto vamos a ver cómo se pueden implementar estos conceptos en la práctica