

CRIPTO · SERIE 1 · PARTE 02 / 04

SHA-256 POR DENTRO

Construcción, operaciones y propiedades de la función hash de Bitcoin, con un ejemplo trabajado ronda a ronda, medidas propias del efecto avalancha y el estado actual de su criptoanálisis.

Complemento técnico de la guía «Un hash por dentro»

<https://amstech.dev/guias/bitcoin-s1/02-un-hash-por-dentro/>

@ams_tech

ÍNDICE

1	Resumen
2	Qué se le exige a una función hash
3	Notación y operaciones
4	La construcción de Merkle–Damgård
5	Las constantes: «nada en la manga»
6	Expansión del mensaje
7	La función de compresión
8	Ejemplo trabajado: SHA-256("hola")
9	El efecto avalancha, medido
10	Por qué no se puede invertir
11	SHA-256 en Bitcoin
12	Lo que el vídeo simplifica
13	Referencias

1. RESUMEN

SHA-256 es la función hash criptográfica estandarizada por el NIST en FIPS 180-4 [1] y la pieza sobre la que se apoya casi todo Bitcoin: la huella de cada bloque, los identificadores de las transacciones, el árbol de Merkle y la prueba de trabajo [10]. Transforma un mensaje de cualquier longitud (hasta $2^{64} - 1$ bits) en un resumen de **256 bits**.

Este dossier describe su funcionamiento completo: la construcción de Merkle-Damgård que la convierte en una función de longitud arbitraria (§4), las constantes y su origen (§5), la expansión del mensaje (§6) y la función de compresión de 64 rondas (§7). Se sigue paso a paso el cálculo de `SHA-256("hola")` (§8) y se cuantifica el efecto avalancha con dos experimentos propios: el ejemplo del vídeo (un bit de diferencia: 117 bits distintos tras 6 rondas) y una muestra de 20.000 pares, cuya distancia media en la salida es de 127,90 bits frente a los 128 teóricos (§9). Por último se resume por qué no se conoce forma de invertirla y qué consiguen los mejores ataques publicados (§10), y cómo la usa Bitcoin (§11).

2. QUÉ SE LE EXIGE A UNA FUNCIÓN HASH

Una función hash H asigna a cada mensaje M un resumen de longitud fija, $n = 256$ bits en SHA-256. Calcularla tiene que ser rápido y, además, tiene que resistir tres tipos de ataque [3]:

PROPIEDAD	EL ATACANTE NO DEBE PODER...	COSTE GENÉRICO
Resistencia a preimagen	dado un resumen h , encontrar M con $H(M) = h$	$\approx 2^{256}$
Resistencia a segunda preimagen	dado M_1 , encontrar $M_2 \neq M_1$ con $H(M_2) = H(M_1)$	$\approx 2^{256}$
Resistencia a colisiones	encontrar dos mensajes cualesquiera con el mismo resumen	$\approx 2^{128}$

«Coste genérico» es lo que cuesta el ataque por fuerza bruta contra una función ideal; una función se considera rota si alguien encuentra un método sustancialmente más barato.

La cota de las colisiones es la mitad en el exponente por la **paradoja del cumpleaños**: entre q resúmenes aleatorios hay $\binom{q}{2} \approx q^2/2$ parejas, y cada una coincide con probabilidad 2^{-n} . El número esperado de intentos hasta la primera colisión es

$$E[q] \approx \sqrt{\frac{\pi}{2}} 2^n = 2^{128,33} \quad (n = 256). \quad (1)$$

Por eso el NIST asigna a SHA-256 una fortaleza de **128 bits** frente a colisiones y de **256 bits** frente a preimágenes [2].

Idea clave. Que haya muchos mensajes con el mismo resumen es inevitable: los mensajes de un solo bloque ya son del orden de 2^{448} , y solo hay 2^{256} resúmenes, así que a cada resumen le corresponden de media unos 2^{192} mensajes. La seguridad no consiste en que las preimágenes no existan, sino en que **no se pueda encontrar ninguna** con menos trabajo que probar a ciegas.

3. NOTACIÓN Y OPERACIONES

SHA-256 trabaja con **palabras de 32 bits**, que se escriben en hexadecimal (8 cifras) y se interpretan como enteros sin signo, con el bit más significativo a la izquierda (*big-endian*). Todas las operaciones son baratas en cualquier procesador:

SÍMBOLO	OPERACIÓN	DEFINICIÓN
$x \oplus y$	XOR bit a bit	1 donde los bits difieren
$x \wedge y, \neg x$	AND y NOT bit a bit	
$x \boxplus y$	suma modular	$(x + y) \bmod 2^{32}$
$\text{SHR}^n(x)$	desplazamiento a la derecha	$x \gg n$ (entran ceros)
$\text{ROTR}^n(x)$	rotación a la derecha	$(x \gg n) \vee (x \ll (32 - n))$

La diferencia entre las dos últimas importa: el desplazamiento **pierde** los n bits que salen; la rotación los reintroduce por la izquierda y es, por tanto, una permutación de los bits.

Las operaciones bit a bit ($\oplus, \wedge$, rotaciones) son lineales o casi lineales sobre $\mathbb{F}_2$; la suma modular no lo es, porque el acarreo propaga información de los bits bajos a los altos. Mezclar ambos mundos (lo que se conoce como diseño **ARX**: *add-rotate-xor*) es lo que impide modelar el algoritmo con un sistema de ecuaciones manejable.

4. LA CONSTRUCCIÓN DE MERKLE-DAMGÅRD

SHA-256 no procesa el mensaje de golpe: lo trocea en bloques de 512 bits y los encadena con una **función de compresión** C que recibe un estado de 256 bits y un bloque de 512 y devuelve un estado nuevo de 256 bits. Es la construcción propuesta de forma independiente por Merkle y por Damgård en 1989 [4].

4.1 RELLENO

Sea ℓ la longitud del mensaje en bits. Se añade un bit **1**, después k ceros, con k el menor entero no negativo tal que

$$\ell + 1 + k \equiv 448 \pmod{512}, \quad (2)$$

y por último ℓ escrito como entero de 64 bits. El resultado tiene una longitud múltiplo de 512. Incluir la longitud (el llamado *refuerzo de Merkle-Damgård*) es lo que permite demostrar que una colisión del hash completo implica una colisión de la función de compresión [4].

Para "hola", $\ell = 32$, así que $k = 415$ y cabe en un único bloque:

```
686f6c61 80000000 00000000 00000000 00000000 00000000 00000000 00000000
00000000 00000000 00000000 00000000 00000000 00000000 00000000 00000020
```

Se reconocen los códigos ASCII de `h o l a` (`68 6f 6c 61`), el bit `1` del relleno (el byte `80`) y, al final, la longitud: `0x20` = 32.

4.2 ITERACIÓN

El mensaje relleno se divide en bloques $M^{(1)}, \dots, M^{(N)}$ y se calcula

$$H^{(0)} = IV, \quad H^{(i)} = H^{(i-1)} \boxplus C(H^{(i-1)}, M^{(i)}), \quad \text{resumen} = H^{(N)}. \quad (3)$$

La suma final del estado de entrada (la estructura de **Davies-Meyer**) hace que, aunque las 64 rondas fueran invertibles para quien conoce el bloque, el paso completo no lo sea: para deshacerla habría que conocer a la vez la entrada y la salida.

5. LAS CONSTANTES: «NADA EN LA MANGA»

SHA-256 usa dos tablas de constantes, y el estándar fija su origen para demostrar que no esconden una puerta trasera [1]:

- **Estado inicial** $H^{(0)}$: los 32 primeros bits de la parte fraccionaria de las raíces cuadradas de los 8 primeros primos.
- **Constantes de ronda** $K_0, \dots, K_{63}$: lo mismo con las raíces cúbicas de los 64 primeros primos.

$$H_j^{(0)} = \lfloor 2^{32} (\sqrt{p_j} - \lfloor \sqrt{p_j} \rfloor) \rfloor, \quad K_t = \lfloor 2^{32} (\sqrt[3]{p_t} - \lfloor \sqrt[3]{p_t} \rfloor) \rfloor. \quad (4)$$

Por ejemplo, $\sqrt{2} = 1,41421356 \dots$ y $0,41421356 \dots \times 2^{32} = 1.779.033.703,95 \dots$, cuya parte entera en hexadecimal es `6a09e667`: la primera palabra del estado inicial. Se han recalculado las 72 constantes con aritmética entera exacta y coinciden con las del estándar.

VALORES

$H^{(0)}$ 6a09e667 bb67ae85 3c6ef372 a54ff53a 510e527f 9b05688c 1f83d9ab 5be0cd19

$K_0 \dots K_3$ 428a2f98 71374491 b5c0fbcf e9b5dba5

6. EXPANSIÓN DEL MENSAJE

Cada bloque de 512 bits se lee como 16 palabras $W_0, \dots, W_{15}$ y se extiende a 64 palabras, una por ronda:

$$W_t = \sigma_1(W_{t-2}) \boxplus W_{t-7} \boxplus \sigma_0(W_{t-15}) \boxplus W_{t-16}, \quad 16 \leq t \leq 63, \quad (5)$$

con

$$\begin{aligned} \sigma_0(x) &= \text{ROTR}^7(x) \oplus \text{ROTR}^{18}(x) \oplus \text{SHR}^3(x) \\ \sigma_1(x) &= \text{ROTR}^{17}(x) \oplus \text{ROTR}^{19}(x) \oplus \text{SHR}^{10}(x) \end{aligned} \quad (6)$$

El objetivo es que cada bit del mensaje intervenga en muchas rondas y en posiciones distintas. Aunque el bloque de "hola" es casi todo ceros, la expansión llena enseguida las palabras: $W_{16} = 796f8c61$, $W_{17} = 80140000$, $W_{18} = 37a2c879$, $W_{19} = 80205508$.

7. LA FUNCIÓN DE COMPRESIÓN

El estado son ocho registros de 32 bits, $a, b, \dots, h$, que se inicializan con $H^{(i-1)}$. Cada ronda usa cuatro funciones:

$$\begin{aligned} \Sigma_0(a) &= \text{ROTR}^2(a) \oplus \text{ROTR}^{13}(a) \oplus \text{ROTR}^{22}(a) \\ \Sigma_1(e) &= \text{ROTR}^6(e) \oplus \text{ROTR}^{11}(e) \oplus \text{ROTR}^{25}(e) \\ \text{Ch}(e, f, g) &= (e \wedge f) \oplus (\neg e \wedge g) \\ \text{Maj}(a, b, c) &= (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c) \end{aligned} \quad (7)$$

- Σ_0 y Σ_1 son **difusión** lineal: cada bit del resultado depende de tres bits separados de la entrada, así que un cambio en un bit se convierte en tres.
- Ch (*choose*) es un multiplexor: donde e vale 1 toma el bit de f , y donde vale 0, el de g .
- Maj (*majority*) devuelve en cada posición el valor mayoritario entre a , b y c .
- Ch y Maj son funciones de grado 2 sobre $\mathbb{F}_2$: junto con los acarros de $\boxplus$, son la fuente de **no linealidad**.

En cada ronda $t = 0, \dots, 63$:

$$\begin{aligned}
T_1 &= h \boxplus \Sigma_1(e) \boxplus \text{Ch}(e, f, g) \boxplus K_t \boxplus W_t \\
T_2 &= \Sigma_0(a) \boxplus \text{Maj}(a, b, c) \\
(a, b, c, d, e, f, g, h) &\leftarrow (T_1 \boxplus T_2, a, b, c, d \boxplus T_1, e, f, g)
\end{aligned} \tag{8}$$

Solo dos registros reciben un valor nuevo (a y e); los otros seis se desplazan una posición, como una cinta transportadora. Por eso un cambio tarda unas pocas rondas en alcanzar todo el estado: es exactamente lo que se mide en §9.

8. EJEMPLO TRABAJADO: SHA-256("hola")

Con el bloque de §4 y el estado inicial de §5, la ronda $t = 0$ usa $W_0 = 686f6c61$ y $K_0 = 428a2f98$. Tras ella, el estado vale:

A	B	C	D	E	F	G	H
6477f4ae	6a09e667	bb67ae85	3c6ef372	01374f03	510e527f	9b05688c	1f83d9ab

Se ve la cinta transportadora: b, c, d son los antiguos a, b, c (las tres primeras palabras de $H^{(0)}$), y f, g, h son los antiguos e, f, g . Solo a y e son nuevos.

Tras las 64 rondas y la suma final de la ecuación (3):

```

SHA-256("hola") = b221d9dbb083a7f33428d7c2a3c3198ae925614d70210e28716ccaa7cd4ddb79
SHA-256("hola") = e028fa3463f8bbcb76607c0819bd85103de648004fcabb0648ce3662459956f4

```

Los valores se han obtenido con una implementación propia de FIPS 180-4, verificada contra la de Node.js (`node:crypto`) con 2.000 mensajes aleatorios de 0 a 199 bytes.

9. EL EFECTO AVALANCHA, MEDIDO

9.1 EL CRITERIO

Webster y Tavares formalizaron en 1985 el **criterio estricto de avalancha** (SAC) [5]: al invertir un solo bit de la entrada, cada bit de la salida debe cambiar con probabilidad exactamente $1/2$. Si además los bits cambian de forma independiente, la distancia de Hamming D entre las dos salidas (el número de bits distintos) sigue una binomial:

$$D \sim \text{Bin}(256, \frac{1}{2}), \quad E[D] = 128, \quad \sigma_D = \sqrt{256 \cdot \frac{1}{2} \cdot \frac{1}{2}} = 8. \tag{9}$$

9.2 RONDA A RONDA: "ho!a" FRENTE A "ho!A"

La `a` (`0x61`) y la `A` (`0x41`) difieren en un único bit. Se ha medido cuántos de los 256 bits del estado interno difieren tras cada ronda:

RONDA	0	1	2	3	4	5	6	8	16	32	64
Bits distintos	0	5	30	57	89	111	117	119	135	126	131

En la ronda 1 la diferencia solo ha llegado a los registros `a` y `e`; cada ronda la propaga a los siguientes y la amplifica a través de Σ_0 , Σ_1 y los acarrees. Hacia la ronda 6 el estado está ya en la zona de 128 ± 8 que predice (9), y a partir de ahí solo fluctúa. Las 58 rondas restantes son el **margen de seguridad**: los ataques conocidos explotan precisamente las rondas en las que la difusión todavía es incompleta (§10). En la salida final, tras la suma de (3), las huellas de "ho!a" y "ho!A" difieren en 131 bits.

9.3 MUESTRA DE 20.000 PARES

Para comprobar (9) estadísticamente se generaron 20.000 mensajes aleatorios de 64 bytes (con semilla fija, para que el experimento sea reproducible), se invirtió en cada uno un bit elegido al azar y se comparó la salida:

MEDIDA	OBSERVADO	TEÓRICO
Distancia media	127,90 bits	128
Desviación típica	8,06 bits	8
Mínimo / máximo	96 / 159	
Probabilidad de cambio de cada bit de salida	entre 0,490 y 0,509	0,5

El error estándar de la media es $8/\sqrt{20.000} \approx 0,057$, así que la diferencia observada (0,10 bits) está dentro de unas dos desviaciones: compatible con una función ideal. Lo mismo ocurre con cada bit por separado: con 20.000 muestras, la desviación típica de una proporción de 0,5 es $\approx 0,0035$, y los valores observados caen dentro de ± 3 desviaciones.

10. POR QUÉ NO SE PUEDE INVERTIR

No existe una demostración de que SHA-256 sea irreversible: su seguridad es **empírica**, apoyada en más de veinte años de criptoanálisis público sin que nadie haya encontrado un atajo. Los argumentos estructurales son:

1. **Pérdida de información.** La función de compresión reduce 768 bits de entrada (512 del bloque y 256 del estado) a 256. Cada salida tiene una cantidad astronómica de entradas posibles y nada en el resultado indica cuál fue.

2. **No linealidad.** Ch, Maj y los acarreos de $\boxplus$ hacen que las ecuaciones que describen 64 rondas tengan un grado altísimo. Los métodos algebraicos que funcionan con unas pocas rondas se vuelven intratables mucho antes de llegar a 64.
3. **Davies–Meyer.** La suma final de (3) impide recorrer las rondas hacia atrás aunque se conozca el bloque.

10.1 ESTADO DEL ARTE

Los mejores ataques publicados atacan versiones **recortadas** del algoritmo:

- Mendel, Nad y Schl  ffer [6] obtuvieron en 2013 colisiones para SHA-256 reducido a 31 de sus 64 pasos, con un coste te  rico de $2^{65,5}$, y colisiones de *inicio semilibre* (en las que el atacante elige tambi  n el estado inicial) para 38 pasos, con un coste pr  ctico de 2^{37} .
- Li, Liu y Wang [7] presentaron en 2024 el primer par de mensajes que colisiona en la pr  ctica para 31 pasos, y colisiones de inicio semilibre pr  cticas para 39 pasos.

Tras m  s de una d  cada de mejoras, el margen sigue siendo amplio: ning  n ataque se acerca a las 64 rondas completas.

Ordenadores cu  nticos. El algoritmo de Grover [8] reduce la b  squeda de preim  genes de 2^{256} a unas 2^{128} evaluaciones, que siguen estando muy fuera de alcance. Por eso SHA-256 se considera adecuado tambi  n a largo plazo.

11. SHA-256 EN BITCOIN

11.1 DOBLE SHA-256

Bitcoin aplica la funci  n dos veces:

$$\text{SHA256d}(M) = \text{SHA256}(\text{SHA256}(M)). \quad (10)$$

Con ella calcula la huella de las cabeceras de bloque, los identificadores de transacci  n y los nodos del   rbol de Merkle [10]. Satoshi Nakamoto no explic   el motivo. La raz  n que se cita habitualmente es la protecci  n frente a los **ataques de extensi  n de longitud**: en Merkle–Damg  rd, conociendo $H(M)$ y la longitud de M se puede calcular $H(M \parallel \text{relleno} \parallel X)$ sin conocer M , porque el resumen es el estado interno. Aplicar el hash dos veces, como propusieron Ferguson y Schneier [9], corta esa posibilidad.

11.2 MINAR ES BUSCAR PREIM  GENES PARCIALES

La prueba de trabajo exige encontrar una cabecera B con

$$\text{SHA256d}(B) < T, \quad (11)$$

donde T es el objetivo de la red. Es una **preimagen parcial**: no se busca un resumen concreto, sino uno de los T resúmenes que están por debajo del objetivo. Si SHA256d se comporta como una función aleatoria (y el §9 indica que sí), cada intento acierta con probabilidad $T/2^{256}$, de forma independiente, y no existe mejor estrategia que probar.

La cabecera ocupa 80 bytes, así que tras el relleno son **dos bloques** de 512 bits. El primero (versión, huella anterior y los primeros 28 bytes de la raíz de Merkle) no cambia mientras el minero recorre los valores del nonce, que están en el segundo. Los mineros calculan una sola vez el estado intermedio del primer bloque (el **midstate**) y por cada intento solo evalúan **dos** compresiones en lugar de tres: el segundo bloque y el SHA-256 exterior. Es un ejemplo de cómo la construcción de §4 se aprovecha en hardware real.

12. LO QUE EL VÍDEO SIMPLIFICA

EN EL VÍDEO	CON PRECISIÓN
«Solo hace una cosa: mezclar bits»	Hay tres fases: relleno, expansión del mensaje y 64 rondas de compresión, más la suma final de Davies–Meyer
«Se giran, se cruzan con XOR y se suman»	Son las operaciones base; se combinan en Σ_0 , Σ_1 , σ_0 , σ_1 , Ch y Maj
«Se rellena hasta llegar a 512»	Hasta un múltiplo de 512, con un bit 1, ceros y la longitud en 64 bits
«En 6 rondas, casi la mitad»	117 de 256 bits en el ejemplo "ho1a" / "ho1A"; el valor exacto depende de qué bit cambie
«No se conoce forma de darle la vuelta»	Correcto: es una conjetura respaldada por el criptoanálisis, no un teorema

13. REFERENCIAS

- [1] National Institute of Standards and Technology. *FIPS PUB 180-4: Secure Hash Standard (SHS)*. 2015. <https://doi.org/10.6028/NIST.FIPS.180-4>
- [2] Q. Dang. *NIST SP 800-107 Rev. 1: Recommendation for Applications Using Approved Hash Algorithms*. NIST, 2012. <https://doi.org/10.6028/NIST.SP.800-107r1>
- [3] P. Rogaway, T. Shrimpton. «Cryptographic Hash-Function Basics: Definitions, Implications, and Separations for Preimage Resistance, Second-Preimage Resistance, and Collision Resistance». *FSE 2004*, LNCS 3017, Springer.
- [4] R. C. Merkle. «One Way Hash Functions and DES». *CRYPTO '89*, LNCS 435, Springer. I. B. Damgård. «A Design Principle for Hash Functions». *CRYPTO '89*, LNCS 435, Springer.
- [5] A. F. Webster, S. E. Tavares. «On the Design of S-Boxes». *CRYPTO '85*, LNCS 218, Springer.
- [6] F. Mendel, T. Nad, M. Schläffer. «Improving Local Collisions: New Attacks on Reduced SHA-256». *EUROCRYPT 2013*, LNCS 7881, Springer.

- [7] Y. Li, F. Liu, G. Wang. «New Records in Collision Attacks on SHA-2». *EUROCRYPT 2024*, Springer.
- [8] L. K. Grover. «A Fast Quantum Mechanical Algorithm for Database Search». *STOC '96*, ACM.
- [9] N. Ferguson, B. Schneier. *Practical Cryptography*. Wiley, 2003.
- 10] S. Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System*. 2008. <https://bitcoin.org/bitcoin.pdf>

Las cifras de §5, §6, §8 y §9 se han calculado con un script propio (SHA-256 según [1], verificado contra `node:crypto`). El experimento de §9 usa una semilla fija y es reproducible.