Las Matemáticas que sostienen el Blockchain: el lenguaje secreto de la confianza digital

Leonardo Cerchiara

Resumen— La tecnología blockchain constituye un ejemplo relevante de aplicación de la matemática discreta, la probabilidad y la criptografía a sistemas distribuidos. En este artículo se analizan algunos de los conceptos matemáticos que permiten su funcionamiento, incluyendo funciones hash criptográficas, estructura temporal de la cadena, árboles de Merkle y el mecanismo probabilístico de consenso Proof of Work. También se introducen los protocolos tolerantes a fallos bizantinos y la teoría de juegos como marco para analizar la estabilidad del sistema.

Palabras Claves— Blockchain, Criptografía, Probabilidad, Consenso, Matemática Discreta.

1. Introducción

Blockchain es un registro digital descentralizado de transacciones, compartido entre los nodos de una red, que almacena datos de forma segura, transparente e inmutable, impidiendo su modificación una vez registrados. La tecnología blockchain ha despertado un gran interés en los últimos años debido a sus aplicaciones en criptomonedas y sistemas distribuidos. Sin embargo, más allá de su implementación práctica, su funcionamiento se apoya en una base matemática bien definida. La seguridad del sistema no depende de una autoridad central, sino de propiedades probabilísticas y de la dificultad computacional de ciertos problemas [2]. Desde un punto de vista matemático, la blockchain combina herramientas de distintas disciplinas. Las funciones hash permiten garantizar la integridad de la información, la teoría de grafos describe la estructura de los datos [3], la probabilidad modeliza los mecanismos de consenso [1],[5] y la teoría de juegos analiza el comportamiento estratégico de los participantes [4]. La interacción de estos elementos permite construir un sistema descentralizado robusto. En las siguientes secciones se presentarán las herramientas matemáticas esenciales que han permitido el desarrollo de la tecnología blockchain.

2. Funciones Hash Criptográficas

Las funciones hash constituyen uno de los elementos fundamentales de la blockchain [1],[2]. Formalmente, una función hash es una aplicación

$$H: \{0,1\}^* \to \{0,1\}^n$$

que asigna a cada entrada de longitud arbitraria una cadena binaria de longitud fija. En criptografía, estas funciones se modelan como funciones pseudoaleatorias uniformes [2]. Una propiedad esencial es la resistencia a la preimagen. Dado un valor y, encontrar x tal que:

$$H(x) = y$$

requeriría, en el peor de los casos, hasta 2^n intentos mediante fuerza bruta. Esto convierte el problema en computacionalmente intratable, incluso para valores relativamente pequeños de n. Por ejemplo, la función hash SHA-1 produce salidas de 160 bits, lo que implica un total de 2^{160} posibles resultados, aproximadamente 1.4×10^{48}. Otra propiedad importante es la resistencia a colisiones. Una colisión ocurre cuando existen $x_1 \neq x_2$ tales que

$$H(x_1) = H(x_2).$$

El número de valores hash que deben generarse para que exista una probabilidad de al menos 0.5 de obtener una colisión es del orden de $2^{n/2}$ [2]. Este resultado se explica mediante la paradoja del cumpleaños, un principio probabilístico que muestra que, en un conjunto relativamente pequeño de elementos elegidos al azar, la probabilidad de que dos coincidan es mucho mayor de lo que sugiere la intuición. Aplicado a las funciones hash, esto significa que no es necesario generar 2^n valores para encontrar una colisión, sino aproximadamente $2^{n/2}$.

Leonardo Cerchiara, Master en Ingegneria Informatica, Universidad de Huelva, leonardocerchiara16@gmail.com

3. Estructura Matemática De La Blockchain

Una blockchain puede interpretarse como una sucesión ordenada de bloques enlazados criptográficamente [1]. Denotando por

$$B_1,\ B_2, \dots, B_n$$

los bloques de la cadena, cada uno de ellos contiene un conjunto de datos D_i, que almacena la información que se quiere guardar (por ejemplo, los datos de una transacción bancaria), una marca temporal t_i y el hash del bloque anterior. Por tanto, cada bloque puede representarse como

$$B_i = \big(D_i, t_i, H(B_{i-1})\big).$$

La inclusión de la marca temporal introduce una estructura temporal que permite ordenar cronológicamente los bloques. Si t_i representa la marca temporal del bloque B_i, se obtiene la relación

$$t_1 \le t_2 \le \cdots \le t_n$$

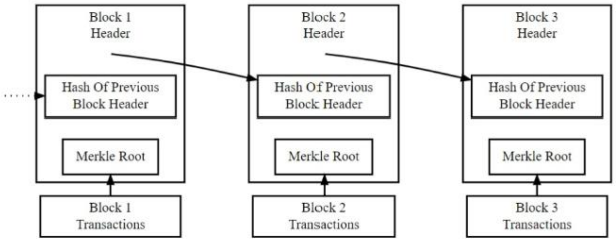

Ilustración 1-Estructura de la blockchain donde cada bloque contiene el hash del bloque anterior y la raíz de Merkle.

Esta ordenación impide reorganizaciones arbitrarias de la cadena y permite verificar la coherencia temporal del sistema distribuido. Además, el hecho de que cada bloque contenga el hash del bloque anterior implica que cualquier modificación en un bloque altera todos los bloques posteriores [1],[2]. Esta dependencia recursiva es la que proporciona la inmutabilidad práctica de la blockchain. Desde el punto de vista matemático, la estructura puede interpretarse como una lista enlazada criptográficamente donde cada nodo depende del anterior mediante una función hash [1].

4. Árboles De Merkle

Dentro de cada bloque, la información contenida en los conjuntos de datos D_i (anteriormente referida como transacciones) no se almacena de forma lineal sino que se organiza mediante árboles de Merkle [1],[3]. Esta estructura se basa en árboles binarios, que en matemática discreta se definen como grafos conexos sin ciclos con estructura jerárquica. El proceso comienza calculando el hash de cada dato individual. Posteriormente, los hashes se agrupan de dos en dos y se vuelve a calcular el hash del resultado. Este procedimiento se repite recursivamente hasta obtener un único valor denominado Merkle root [1],[3]. Si H_L y H_R representan dos nodos hijos, el nodo padre se define como

$$H_P = H(H_L \,||\, H_R),$$

donde || significa concatenación. La principal ventaja de esta estructura es que permite verificar la pertenencia de una transacción utilizando únicamente una parte del árbol. Como consecuencia, la complejidad de verificación se reduce de orden lineal a orden logarítmico,

$$O(\log n)$$

lo que resulta especialmente eficiente en sistemas distribuidos [3].

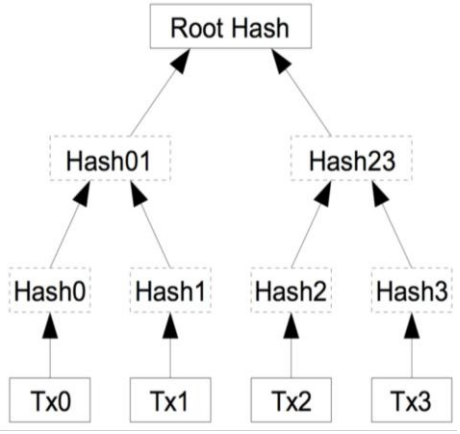

Ilustración 2-Árbol de Merkle donde cada nodo interno es el hash de sus nodos hijos.

5. Proof Of Work y Modelos Probabilísticos

Antes de analizar el Proof of Work, es necesario definir qué es un mecanismo de consenso: se trata de un protocolo que permite a todos los nodos de la red alcanzar un acuerdo común sobre la validez de los datos, asegurando la integridad del registro sin una autoridad central [1],[4]. El mecanismo de consenso más conocido es el Proof of Work. En este modelo, los nodos deben encontrar un valor denominado nonce que satisfaga la condición

$$H(datos \,||\, nonce) < T$$

donde T es un umbral fijado por la red. Debido al comportamiento pseudoaleatorio de la función hash, la única estrategia posible consiste en probar valores de forma repetida. Si la salida de la función hash se

distribuye uniformemente, la probabilidad de éxito en un intento es

$$p = T/2^n$$

[2],[5]. El número de intentos necesarios sigue una distribución geométrica cuya esperanza es

$$E[N] = \frac{1}{p}.$$

En una situación práctica, se requieren trillones de intentos (hashes por segundo) para que la red encuentre colectivamente una solución válida. A nivel global, la creación de bloques puede modelarse mediante un proceso de Poisson [1],[5], lo que permite ajustar dinámicamente la dificultad del sistema y mantener aproximadamente constante el tiempo medio entre bloques. Este coste computacional es el que proporciona seguridad al sistema, ya que modificar un bloque implicaría rehacer el trabajo asociado a todos los bloques posteriores.

6. Consenso Bizantino Y Teoría De Juegos

El problema del consenso en blockchain está relacionado con el problema de los generales bizantinos, en el que varios participantes deben alcanzar una decisión común sobre si un dato o información es verdadera o falsa, aunque algunos puedan actuar de forma maliciosa [6]. Actuar de forma maliciosa se refiere a tomar una decisión incorrecta a sabiendas de que lo es, o enviar información contradictoria para evitar el consenso. Este problema pone de manifiesto la dificultad de lograr acuerdo en sistemas distribuidos. Los protocolos tolerantes a fallos bizantinos demuestran que el consenso puede alcanzarse siempre que el número de nodos maliciosos sea menor que un tercio del total [4],[6], esto es,

$$f < n/3.$$

Estos mecanismos se combinan con incentivos económicos diseñados mediante teoría de juegos [4]. Los participantes reciben recompensas por actuar correctamente y penalizaciones en caso contrario, lo que conduce a un equilibrio de Nash en el que seguir el protocolo es la estrategia óptima.

7. Conclusiones

La tecnología blockchain constituye un ejemplo significativo de aplicación de la matemática a sistemas distribuidos. Funciones hash, estructuras de grafos, modelos probabilísticos y teoría de juegos trabajan conjuntamente para construir un sistema descentralizado seguro. Desde el punto de vista matemático, la seguridad no se basa en certezas absolutas, sino en probabilidades extremadamente pequeñas y problemas computacionalmente difíciles. Este enfoque permite diseñar mecanismos de consenso robustos y abre nuevas líneas de investigación en criptografía y computación distribuida.

Referencias

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System", 2008. https://bitcoin.org/bitcoin.pdf

[2] J. Katz and Y. Lindell, *Introduction to Modern Cryptography*, Chapman & Hall/CRC, pp. 231-290, 2014.

[3] R. C. Merkle, “Protocols for Public Key Cryptosystems,” in *Proceedings of the IEEE Symposium on Security and Privacy*, 1980.

[4] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” *Proceedings of the Third Symposium on Operating Systems Design and Implementation (OSDI)*, 1999.

[5] C. Dwork and M. Naor, "Pricing via Processing or Combatting Junk Mail", Journal of Cryptology, vol. 5, no. 2, pp. 139-147, 1993.

[6] L. Lamport, R. Shostak and M. Pease, “The Byzantine Generals Problem,” Technical Report, SRI International, California, USA, 1982.

Leonardo Cerchiara, Máster en Ingeniería Informatica (Estudiante Erasmus), 1er curso.