Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional

Departamento de Matemáticas

Bifurcaciones en cadenas de bloques bajo las reglas de la cadena más larga y el subárbol observado más pesado

T E S I S

QUE PARA OBTENER EL GRADO DE:

Maestro en Ciencias
en la Especialidad de Matemáticas

PRESENTA:

Daniel Gregorio Longino

TUTOR

Dr. Ruy Fabila Monroy

Ciudad de México, 2019

Índice general

Introducción

"El comercio en el internet ha venido a depender exclusivamente de instituciones financieras las cuales sirven como terceros confiables para el procesamiento de pagos electrónicos. Mientras que el sistema funciona lo suficientemente bien para la mayoría de las transacciones, aun sufren de las debilidades inherentes del modelo basado en la confianza. Transacciones completamente no reversibles no son realmente posibles, dado que las instituciones financieras no pueden evitar mediar disputas. El costo de la mediación incrementa costos de transacción, limitando el tamaño mínimo práctico por transacción y eliminando la posibilidad de pequeñas transacciones casuales, y hay un costo más amplio en la pérdida de la habilidad de hacer pagos no-reversibles por servicios no-reversibles. Con la posibilidad de revertir, la necesidad de confianza se expande. Los comerciantes deben tener cuidado de sus clientes, molestándolos pidiendo más información de la que se necesitaría de otro modo. Un cierto porcentaje de fraude es aceptado como inevitable"(Nakamoto, 2008, p.1).

Ante esta situación, lo que se necesitaba era un sistema de pagos electrónico basado en pruebas criptográficas en vez de confianza, pemitiendole a dos partes interesadas realizar transacciones directamente sin la necesidad de un tercero confiable.

En 2009 aparece Bitcoin una moneda electrónica descentralizada concebida por Satoshi Nakamoto. La idea central detrás del protocolo de Bitcoin es reemplazar el control centralizado de la transmisión del dinero generalmente asumido por las grandes organizaciones tales como bancos, compañias de tarjetas de crédito y otros transmisores de dinero, por una gran red usuario-a-usuario. Los nodos de esta red verifican el trabajo de cada uno y por lo tanto se aseguran de que ninguna entidad sea capaz de comportarse mal. Bitcoin logra esto manteniendo un registro público y completo de todas sus

transacciones en cada nodo de la red. Este libro principal el cual es conocido como la cadena de bloques esta compuesta por una secuencia creciente de bloques, cada una contiene un conjunto de la transacciones aprobadas. El principal desafío que Bitcoin supera es la sincronización del libro principal entre varios de los nodos. Terceros maliciosos quizás intenten interferir con esta sincronización con el objetivo del doble gasto para redirigir pagos previamente procesados que les permitirán a ellos usar el mismo dinero dos veces.

Para ayudar a resolver el problema de doble gasto se requiere que los bloques contengan una prueba de trabajo, el cual es computacionalmente difícil de hacer. La dificultad de esta labor se configura de manera adaptativa de manera que un bloque sea creado aproximadamente una vez cada 10 minutos en la red entera. El intervalo de 10 minutos permite a los bloques propagarse a la mayoría de los nodos antes de que otro bloque sea creado. Si un nodo recibe dos bloques conflictivos, los cuales fueron creados por distintos nodos que desconocen el trabajo de cada uno (o quizás por un atacante malicioso), este resuelve el conflicto seleccionando el bloque correspondiente a la cadena de bloques más larga y adoptándola. El análisis original de Satoshi Nakamoto muestra que mientras el atacante tenga menos del 50 % del poder computacional de la red entera, la probabilidad de que el ataque de doble gasto sea exitoso decrece exponencialmente con el tiempo, lo cual esencialmente permite a los pagos ser considerados aceptados e irreversibles después de algún tiempo. El análisis, sin embargo, asume que los bloques son enviados a través de la red más rápido de lo que estos son creados, y por lo tanto no se ajusta a un escenario en el que muchas transacciones son procesadas por la red (el cual requiere la creación de bloques más grandes, que requieren más tiempo para transmitirse).

"Mientras varios obstáculos tales como incertidumbre regulatoria y una infraestructura subdesarollada aún necesitan ser superados, los principales retos que deben ser enfrentados desde una perspectiva de ciencias de la computación están relacionados con la capacidad de Bitcoin para escalar a mayores tasas de transacción y su capacidad para procesar rápidamente transacciones individuales " (Sompolinsky y Zohar, 2015).

En efecto, la capacidad para el procesamiento de transacciones adicionales en Bitcoin es muy necesaria. A partir de dicimebre de 2014, la red de Bitcoin procesa alrededor de 90 mil transacciones por día [1], un número el cual

ha ido lentamente creciendo, y equivale en promedio a una transacción por segundo (TPS). En contraste, el sistema de pago global de Visa manejó un informe de 150 millones de transacciones por día (aproximadamente 2000 TPS) y ha ido creciendo desde entonces. Si Bitcoin no es capaz de escalar a tasas apropiadas que coincidan con la demanda, las tarifas de transacciones aumentarán y los usuarios serán conducidos a usar otras formas de pago.

El bajo número actual de transacciones de Bitcoin se debe principalmente a su pequeña base de usuarios. Una vez que la adopción aumente, el sistema necesitará escalar para procesar transacciones a una tasa más alta, y las garantías anteriores de seguridad ya no se pueden mantenerse. Teniendo en cuenta esto, en el artículo *Secure High-Rate Transaction Proccesing in Bitcoin* [8] los investigadores Yonatan Sompolinsky y Aviv Zohar estudian que tan susceptible es el protocolo ante el doble gasto cuando más transacciones son procesadas por segundo. Ellos hacen notar que bloques de mayor tamaño y los eventos de creación de bloques más frecuentes dan como resultado más conflictos entre bloques, lo cual reduce drásticamente el nivel de seguridad ante ataques. Para resolver estos problemas de seguridad, los investigadores antes mencionados proponen una alternativa a la regla de la cadena más larga llamada GHOST, el cual cambia el procedimiento para la resolución de conflictos para la cadena de bloques. GHOST selecciona en cada bifurcación de la cadena, el subárbol más pesado enraizado en la bifurcación. Ellos muestran que esta modificación del protocolo alivia los problemas de seguridad antes mencionados.

Teniendo en cuenta lo antes descrito, los objetivos de esta tesis son: primero describir el funcionamiento de Bitcoin, y segundo, estudiar la regla de selección de la cadena principal mediante GHOST.

En el primero capítulo se presentan algunos temas de la Teoría de la Probabilidad, así como ciertos resultados que utilizaremos.

El segundo capítulo está dedicado a estudiar la parte criptográfica detrás de Bitcoin: funciones hash, punteros hash y estructura de datos, y firmas digitales.

En el tercer capítulo explicamos cómo funciona Bitcoin y también realizamos un análisis ante el doble gasto.

En el cuarto capítulo nos enfocamos en el estudio de GHOST. Se muestran algunas propiedades básicas de GHOST. Se realiza un análisis de la tasa de

crecimiento de la cadena principal cuando se sigue GHOST y cuando se sigue la regla de la cadena más larga. Ya en la parte final se retoma el tema de doble gasto en redes con retrasos.

Capítulo 1

Preliminares

En este capítulo presentamos algunos temas de la Teoría de la Probabilidad, así como ciertos resultados, con la finalidad de sentar los fundamentos teóricos necesarios para el desarrollo de esta tesis.

1.1. El problema de la ruina del jugador

Una variable aleatoria de Bernoulli de parámetro $p \in (0,1)$ toma el valor 1 con probabilidad p y el valor -1 con probabilidad $1-p$. Es, entre otras cosas, un modelo adecuado para describir un volado (águila es 1 y sol es -1). Estas variables aleatorias se llaman así en honor al famoso matemático Jakob Bernoulli, quién en su célebre libro Ars Conjectandi probó el primer teorema de los grandes números precisamente para estas variables llamadas binarias porque sólo pueden tomar dos valores. Este teorema ha tenido una trascendencia enorme en todo el desarrollo posterior de la teoría de la probabilidad [5].

Las caminatas aleatorias simples pueden verse como un modelo para una sucesión de juegos de volados, que se suponen independientes entre sí; hay dos jugadores que llamaremos A y B; cada jugador apuesta en un volado un peso. Gana ese peso el jugador A con probabilidad $p \in (0,1)$ (sale águila) y lo pierde con probabilidad $q = 1-p$ (sale sol). La independencia significa que lo que sucede en un volado no influye en los demás volados. El modelo matemático se construye con una sucesión de variables aleatorias independientes $(X_n)_{n\in\mathbb{N}}$

(que describen a los resultados de los volados sucesivos), y cada una de ellas es una variable Bernoulli de parámetro $p \in (0,1)$. Si el capital inicial de A es $a \in \mathbb{N}$ pesos y el de B es $c-a \in \mathbb{N}$ pesos, el capital de A después de n volados es $S_n = a + \sum_{i=1}^{n} X_i$, y el capital de B será $T_n = c - a - \sum_{i=1}^{n} X_i$. A la sucesión $\{S_n : n \in \mathbb{N}\}$ se le llama caminata aleatoria simple que empieza en a.

El problema que nos interesa es saber cuándo alguno de los dos jugadores se arruina, es decir, cuando alguno de los dos pierde todo su dinero y entonces el juego termina y el jugador que ganó todo el capital c es quién gana el juego. ¿Cuál es la probabilidad de que gane todo el capital el jugador A? ¿Cuál la de que gane B? Claramente esto va a depender del valor de p y del monto del capital inicial de cada uno. Una forma de resolver el problema es usando probabilidad condicional. Para ello, se observa que pasa en el primer juego y se define la probabilidad buscada en términos del capital inicial del jugador A y se mantiene constante el capital total c. Sea $h(u) := \mathbb{P}(R_u)$ la probabilidad de que A pierda habiendo empezado con $u \in \{1, 2, \ldots, c-1\}$ pesos. Por el teorema de probabilidad total se tiene lo siguiente:

$$\begin{aligned}\mathbb{P}(R_u) &= \mathbb{P}(R_u \mid S_1 = u+1)\mathbb{P}(S_1 = u+1) + \mathbb{P}(R_u \mid S_1 = u-1)\mathbb{P}(S_1 = u-1) \\ &= p\mathbb{P}(R_{u+1}) + q\mathbb{P}(R_{u-1}) \\ &= ph(u+1) + qh(u-1).\end{aligned}$$

Como además se sabe que $h(0) = 1$ y $h(c) = 1$, el problema consiste en resolver la ecuación en diferencias finitas

$$h(u) = ph(u+1) + qh(u-1),\ h(0) = 1,\ h(c) = 1.$$

En el caso $p = 1/2 = q$, se puede obtener rápidamente que

$$kh(c-1) = h(c-k) \text{ para } k = 0, 1, \ldots, c.$$

Luego tomando $u = c - k$ obtenemos

$$h(u) = \frac{c-u}{c}. \tag{1.1.1}$$

Para resolver el sistema en el caso $p \neq q$, procedemos de la siguiente manera

$$(p+q)h(u) = ph(u+1) + qh(u-1)$$

lo que es equivalente a

$$p(h(u) - h(u+1)) = q(h(u-1) - h(u)),\ u = 1, 2, \ldots, c-1. \tag{1.1.2}$$

Sean $d_k = h(k) - h(k+1),\ u = 1, 2, \ldots, c-1$, y $r = q/p$.
La igualdad (1.1.2) escrita con esta notación es

$$d_u = r d_{u-1}$$

y por iteración se llega a

$$d_u = r^u d_0$$

en consecuencia

$$1 = h(0) - h(c) = \sum_{j=0}^{c-1} d_j = d_0 \sum_{j=0}^{c-1} r^j = d_0 \left(\frac{1 - r^c}{1 - r} \right),$$
$$h(u) = h(u) - h(c) = \sum_{i=u}^{c-1} d_i = d_0 \sum_{i=u}^{c-1} r^i = d_0 \left(\frac{r^u - r^c}{1 - r} \right).$$

De la primera ecuación se obtiene el valor de d_0 que se sustituye en la segunda y se obtiene

$$h(u) = \frac{r^u - r^c}{1 - r^c} \tag{1.1.3}$$

De esta manera hemos obtenido la probabilidad de que el jugador A pierda el juego si empieza con u pesos.

El caso límite en que $c = \infty$ corresponde a un juego contra un adversario infinitamente rico. Si tomamos $c \to \infty$ en (1.1.1) y (1.1.3), obtenemos

$$h(u) = \begin{cases} 1 & \text{si } p \leq q \\ (q/p)^u & \text{si } p > q. \end{cases} \tag{1.1.4}$$

Interpretamos a $h(u)$ como la probabilidad de ruina a largo plazo de un jugador con capital inicial u, que juega contra un adversario infinitamente rico.

1.2. Las funciones de riesgo y supervivencia

Sea T una variable aleatoria no negativa que representa el tiempo de espera hasta la ocurrencia de un cierto evento. Por simplicidad adoptamos la terminología de análisis de supervivencia, refiriendonos al evento de interés como "muerte", y el tiempo de espera como el "tiempo de supervivencia".

Supongamos que T es una variable aleatoria continua con función de densidad $f(t)$ y función de distribución acumulada $F(t) = \mathbb{P}(T < t)$, el cual nos da la probabilidad de que el evento se haya producido antes de t unidades de tiempo. A menudo es conveniente trabajar con el complemento de la función de distribución acumulada, la *función de supervivencia*

$$S(t) = \mathbb{P}(T \geq t) = 1 - F(t) = \int_t^\infty f(x)dx, \tag{1.2.1}$$

el cual nos da la probabilidad de que el evento de interés no se haya producido antes de t unidades de tiempo.

Una caracterización alternativa de la distribución de T está dada por la *función de riesgo* o *tasa instantánea de ocurrencia del evento*, definido como

$$\lambda(t) = \lim_{dt \to 0} \frac{\mathbb{P}(t \leq T < t + dt \mid T \geq t)}{dt}. \tag{1.2.2}$$

El numerador de esta expresión es la probabilidad condicional de que el evento ocurra en el intervalo $[t, t + dt]$ dado que este no ha ocurrido antes del tiempo t, y en el denominador tenemos la longitud del intervalo. Dividiendo uno por el otro obtenemos una tasa de ocurrencia del evento por unidad de tiempo. Tomando el límite cuando la longitud del intervalo tiende a cero, obtenemos una tasa de ocurrencia instantánea.

Ahora notemos que

$$\begin{aligned}
\lim_{dt \to 0} \frac{\mathbb{P}(t \leq T < t + dt \mid T \geq t)}{dt} &= \lim_{dt \to 0} \frac{\mathbb{P}(t \leq T < t + dt, T \leq t)}{\mathbb{P}(T \leq t)dt}. \\
&= \lim_{dt \to 0} \frac{\mathbb{P}(t \leq T < t + dt)}{\mathbb{P}(T \geq t)dt} \\
&= \frac{f(t)}{S(t)}
\end{aligned}$$

en consecuencia

$$\lambda(t) = \frac{f(t)}{S(t)}. \tag{1.2.3}$$

Observemos que de la ecuación 1.2.1 se sigue que $-f(t)$ es la derivada de $S(t)$. Entonces la igualdad anterior puede ser reescrita como

$$\lambda(t) = -\frac{d}{dt}\left(\log(S(t)\right). \tag{1.2.4}$$

Si ahora integramos de 0 a t e introducimos la condición $S(0) = 1$ (ya que el evento no ha ocurrido antes del tiempo cero) entonces podemos resolver la expresión de arriba para obtener una fórmula para la función de supervivencia.

$$\begin{aligned}\int_0^t \lambda(x)dx &= -\int_0^t \frac{d}{dx}\left(\log(S(x)\right)dx \\ &= -(\log(S(x)) - \log(S(0))) \\ &= -\log(S(x)).\end{aligned}$$

Por lo tanto

$$S(t) = \exp\{-\int_0^t \lambda(x)dx\}. \tag{1.2.5}$$

La integral entre llaves en la igualdad anterior se llama la *función de riesgo acumulada* y se denota por

$$\Lambda(t) = \int_0^t \lambda(x)dx. \tag{1.2.6}$$

1.3. El proceso Poisson

Suponga que un mismo evento ocurre repetidas veces de manera aleatoria a lo largo del tiempo, como se muestra en la figura 1.1. Tal evento puede ser, por ejemplo, la llegada de una reclamación a una compañía aseguradora o la recepción de una llamada a un conmutador, la llegada de un cliente a una ventanilla para solicitar algún servicio o los momentos en los que una cierta

máquina requiere reparación, etcétera. Suponga que las variables aleatorias $T_1, T_2, \ldots$ representan los tiempos que transcurren entre una ocurrencia del evento y la siguiente ocurrencia. Suponga que estos tiempos son independientes uno del otro y que cada uno tiene distribución $exp(\lambda)$. Se define el proceso de Poisson al tiempo t como el número de ocurrencias del evento que se han observado hasta ese instante t [4]. Esta es una definición constructiva de este proceso, la cual formalizamos a continuación.

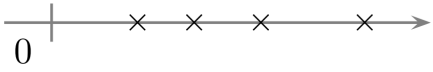

Figura 1.1.

Definición 1. Sea $T_1, T_2, \ldots$ una sucesión de variables aleatorias independientes cada una con distribución $exp(\lambda)$. El **proceso de Poisson** de parámetro λ es el proceso a tiempo continuo $\{X_t : t \geq 0\}$ definido de la siguiente manera:

$$X_t = \text{máx}\{n \geq 1 : T_1 + \ldots + T_n \leq t\}.$$

Se postula que el proceso inicia en cero y para ello se define $\text{máx}\, \varnothing = 0$. En palabras, la variable X_t es el entero n máximo tal que $T_1 + \ldots + T_n$ es menor o igual a t, y ello equivale a contar el número de eventos ocurridos hasta el tiempo t. A este proceso se llama proceso de Poisson homogéneo, tal adjetivo se refiere a que el parámetro λ no cambia con el tiempo, es decir, es homogéneo en el tiempo. Una trayectoria típica de este proceso puede observarse en la figura 1.2, la cual es no decreciente, constante por partes, continua por la derecha y con límite por la izquierda. A los tiempos $T_1, T_2, \ldots$ se les llama tiempos de estancia o tiempos de interarribo, y corresponden a los tiempos que transcurren entre un salto del proceso y el siguiente salto. Hemos supuesto que estos tiempos son independientes y que todos tiene distribución $\exp(\lambda)$. En consecuencia, la variable $W_n = T_1 + \ldots + T_n$ tiene distribución $\text{gamma}(n, \lambda)$. Esta variable representa el tiempo real en el que se observa la ocurrencia del n-ésimo evento. Observe la igualdad de eventos $(X_t \geq n) = (W_n \leq t)$, esto equivale a decir que al tiempo t han ocurrido por lo menos n eventos si y sólo si el n-ésimo evento ocurrió antes de t.

Una de las características sobresalientes de este proceso es que puede encontrase explícitamente la distribución de probabilidad de la variable X_t para cualquier $t > 0$. La respuesta es la distribución Poisson, y de allí el proceso adquiere su nombre.

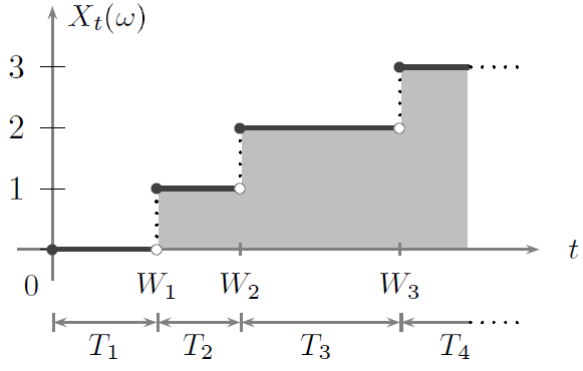

Figura 1.2. El proceso de Poisson y los tiempos de ocurrencia de eventos.

Teorema 2. *La variable aleatoria X_t tiene distribución Poisson(λt), es decir, para cualquier $t > 0$, y para $n = 0, 1, \ldots$*

$$\mathbb{P}(X_t = n) = e^{-\lambda t}\frac{(\lambda t)^n}{n!}.$$

Demostración. Como W_n tiene distribución gama(n, λ), su función de distribución es, para $t > 0$,

$$\mathbb{P}(W_n \leq t) = 1 - e^{-\lambda t}\sum_{k=0}^{n-1}\frac{(\lambda t)^n}{n!}.$$

Entonces para cualquier $t > 0$ y para cada $n = 0, 1, \ldots$

$$\begin{aligned}\mathbb{P}(X_t = n) &= \mathbb{P}(X_t \geq n) - \mathbb{P}(X_t \geq n+1)\\ &= \mathbb{P}(W_n \leq t) - \mathbb{P}(W_{n+1} \leq t)\\ &= e^{-\lambda t}\frac{(\lambda t)^n}{n!}.\end{aligned}$$

■

1.4. Martingalas

Sea $(\Omega, \mathscr{F}, \mathbb{P})$ un espacio de probabilidad para modelar un cierto experimento aleatorio. La $\sigma-$álgebra $\mathscr{F}$ es la estructura que agrupa a los eventos del experimento aleatorio a los cuales se les puede calcular su probabilidad. Suponga ahora que se consideran dos sub $\sigma-$álgebras $\mathscr{F}_1$ y $\mathscr{F}_2$ tales que $\mathscr{F}_1 \subseteq \mathscr{F}_2$. Entonces $\mathscr{F}_2$ contiene más información que $\mathscr{F}_1$ en el sentido de que, en general, un mayor número de subconjuntos son considerados como eventos. Más generalmente, puede considerarse una sucesión no decreciente de sub $\sigma-$álgebras $\mathscr{F}_1 \subseteq \mathscr{F}_2 \subseteq \ldots$. Este tipo de estructuras surgen de manera natural, por ejemplo, para un proceso estocástico a tiempo discreto $\{X_n : n \geq 1\}$, pues puede construirse la sucesión $\{\mathscr{F}_n\}_{n\geq 1}$ de la siguiente forma:

$$\mathscr{F}_n = \sigma\{X_1, \ldots, X_n\},$$

en donde efectivamente se cumple que $\mathscr{F}_1 \subseteq \mathscr{F}_2 \subseteq \ldots$. En este caso la $\sigma-$álgebra $\mathscr{F}_n$ contiene los eventos para los cuales puede determinarse su ocurrencia o no ocurrencia, sólo con la información o historia del proceso hasta el tiempo n.

Ejemplo 1. Si las variables $\{X_n : n \geq 1\}$ representan los resultados de lanzar sucesivamente un dado, y se define el evento A como "La suma de los dos primeros resultados es mayor a 3", entonces es claro que $A \notin \mathscr{F}_1$, sin embargo $A \in \mathscr{F}_2$. Por supuesto, si sucede por ejemplo que $X_1 = 4$, entonces sabemos que el evento A ocurre, pero sin esta información adicional la ocurrencia o no ocurrencia del evento A no puede determinarse sino hasta el segundo lanzamiento del dado.

Estas consideraciones sobre sucesiones no decrecientes de $\sigma-$álgebras llevan a la definición de filtración.

Definición 3. Una **filtración** es una colección de $\sigma-$álgebras $\{\mathscr{F}_n\}_{n\geq 1}$ tal que $\mathscr{F}_n \subseteq \mathscr{F}_m$ cuando $n \leq m$. En particular, la filtración natural o canónica de un proceso $\{X_n : n \geq 1\}$ es aquella sucesión de $\sigma-$álgebras definida por

$$\mathscr{F}_n = \sigma\{X_1, \ldots, X_n\}.$$

A tiempo continuo las definiciones de estos conceptos son análogas: una filtración es una colección no numerable de sub $\sigma-$álgebras $\{\mathscr{F}_t\}_{t\geq 0}$ tal que

$\mathscr{F}_s \subseteq \mathscr{F}_t$ cuando $0 \leq s \leq t$. La filtración natural o canónica de un proceso a tiempo continuo $\{X_t : t \geq 0\}$ es la colección de $\sigma-$álgebras $\{\mathscr{F}_t\}_{t\geq 0}$ dadas por $\mathscr{F}_t = \sigma\{X_s : 0 \leq s \leq t\}$, esto es, $\mathscr{F}_t$ es la mínima sub $\sigma-$álgebra que hace que cada una de las variables X_s, para valores de s en el intervalo $[0, t]$, sean medibles. A la $\sigma-$álgebra $\mathscr{F}_t$ se le denomina la historia del proceso al tiempo t. Regresemos al caso de tiempo discreto.

Definición 4. Se dice que un proceso estocástico $\{X_n : n \geq 1\}$ **es adaptado** a una filtración $\{\mathscr{F}_n\}_{n\geq 1}$ si la variable aleatoria X_n es $\mathscr{F}_n-$medible, para cada $n \geq 1$.

Inicialmente sabemos que cada variable aleatoria X_n del proceso estocástico es $\mathscr{F}-$medible. La condición de adaptabilidad requiere que X_n sea también variable aleatoria respecto a la sub $\sigma-$álgebra $\mathscr{F}_n$. Esta condición técnica de adaptabilidad facilita el cálculo de probabilidades de los eventos de un proceso en ciertas situaciones. Naturalmente todo proceso es adaptado a su filtración natural.

Sea $\{X_n : n \geq 1\}$ un proceso adaptado a una filtración $\{\mathscr{F}_n\}_{n\geq 1}$. Un tiempo de paro es una variable aleatoria con valores en el espacio parametral de este proceso, que registra el tiempo en el que ocurre un cierto evento del proceso de tal forma que puede determinarse si ha ocurrido o no ha ocurrido tal evento al tiempo n con la información de la $\sigma-$álgebra $\mathscr{F}_n$. Esta variable aleatoria puede tomar el valor infinito cuando el evento de interés nunca ocurre.

Definición 5. Una variable aleatoria τ con valores en $\{1, 2, \ldots\} \cup \{\infty\}$ es un **tiempo de paro** con respecto de una filtración $\{\mathscr{F}_n\}_{n\geq 1}$ si para cada $n \geq 1$ se cumple que $(\tau = n) \in \mathscr{F}_n$.

Bajo la interpretación de que τ es un tiempo aleatorio en el cual ocurre un cierto evento de interés, la condición $(\tau = n) \in \mathscr{F}_n$. puede interpretarse del siguiente modo: la pregunta de si el evento de interés ha ocurrido al tiempo n, deber ser respondida con la información dada por la filtración al tiempo n.

Ejemplo 2 (**Tiempo de primer arribo**)**.** Sea $X_1, X_2, \ldots$ una sucesión de variables aleatorias adaptadas a la filtración $\{\mathscr{F}_n\}_{n\geq 1}$. Sea A un conjunto de

Borel de $\mathbb{R}$, y defina

$$\tau = \text{mín}\{n \geq 1 : X_n \in A\},$$

en donde conviene definir mín $\varnothing = \infty$. Es decir, τ es el primer momento en el que la sucesión toma un valor dentro del conjunto A, si acaso ello sucede. La variable aleatoria τ es un tiempo de paro, pues para cualquier valor entero de n,

$$(\tau = n) = (X_1 \notin A) \cap \ldots \cap (X_{n-1} \notin A) \cap (X_n \in A) \in \mathscr{F}_n$$

Definición 6. Se dice que un proceso a tiempo discreto $\{X_n : n \geq 1\}$ es una **martingala** respecto de una filtración $\{\mathscr{F}_n\}_{n\geq 1}$ si se cumplen las siguientes tres condiciones:

(a) Es integrable.

(b) Es adaptado a la filtración.

(c) Para cualesquiera $n \leq m$,

$$\mathbb{E}(X_m \mid \mathscr{F}_n) = X_n, \ c.s. \tag{1.4.1}$$

Cuando en lugar de 1.4.1 se cumple la desigualdad $\mathbb{E}(X_m \mid \mathscr{F}_n) \geq X_n$, entonces el proceso es una submartingala, y si $\mathbb{E}(X_m \mid \mathscr{F}_n) \leq X_n$, entonces es una supermartingala.

Además, cuando la filtración es la natural, es decir, cuando $\mathscr{F}_n$ es igual a $\sigma\{X_1, \ldots, X_n\}$, la condición de martingala puede escribirse en la forma

$$\mathbb{E}(X_{n+1} \mid X_n, \ldots, X_1) = X_n.$$

1.5. Dos teoremas importantes

El siguiente resultado establece que, bajo ciertas condiciones, el promedio de variables aleatorias converge casi seguramente a una constante cuando el número de sumandos crece a infinito.

Teorema 7 (Ley fuerte de los grandes números)**.** *Sean $X_1, X_2, \ldots$ variables aleatorias independientes e idénticamente distribuidas con media μ. Entonces*

$$\frac{1}{n}\sum_{i=1}^{n} X_i \xrightarrow{c.s.} \mu$$

Teorema 8 (Desigualdad de Markov)**.** *Sea $X \geq 0$ una variable aleatoria con esperanza finita. Para cualquier $\epsilon > 0$,*

$$\mathbb{P}(X \geq \epsilon) \leq \frac{\mathbb{E}(X)}{\epsilon}.$$

Capítulo 2

Criptografía

"Todas las monedas necesitan alguna forma de controlar el suministro y hacer cumplir varias propiedades de seguridad para evitar trampas. En las monedas fiduciarias, organizaciones como los bancos centrales controlan la oferta monetaria y agregan características antifalsificación de la moneda física, sin embargo, estas características no hacen que el dinero sea imposible de falsificar. En última instancia, la aplicación de la ley es necesaria para evitar que las personas infrinjan las reglas del sistema. Las criptomonedas también deben tener medidas de seguridad que eviten que las personas alteren el estado del sistema y que se equivoquen, es decir, que hagan declaraciones mutuamente inconsistentes a diferentes personas. Si Alice convence a Bob de que le pagó una moneda digital, por ejemplo, no debería poder convencer a Carol de que le pagó esa misma moneda. Pero a diferencia de las monedas fiduciarias, las reglas de seguridad de las criptomonedas deben aplicarse estrictamente tecnológicamente y sin contar con una autoridad central. Como sugiere la palabra, las criptomonedas hacen un uso intensivo de la criptografía. La criptografía proporciona un mecanismo para codificar de forma segura las reglas de un sistema de criptomonedas en el propio sistema. Podemos usarlo para evitar alteraciones y equívocos, así como para codificar las reglas para la creación de nuevas unidades de la moneda en un protocolo matemático" (Narayanan et al., 2016, p. 23).

La criptografía es un campo de investigación académica profunda que utiliza muchas técnicas matemáticas avanzadas que son notoriamente sutiles y complicados de entender. Afortunadamente, Bitcoin sólo se basa en un

puñado de construcciones criptográficas relativamente simples y bien conocidas. En este capítulo, vamos a estudiar específicamente las funciones hash, punteros hash y estructura de datos, así como firmas digitales, estos temas nos ayudarán a comprender de mejor manera la parte criptográfica detrás de Bitcoin.

2.1. Funciones hash

Una ***función hash*** es una función matemática con las siguientes tres propiedades:

1. Su entrada puede ser cualquier cadena de cualquier tamaño.
2. La función produce una salida de tamaño fijo. Aquí vamos a suponer que las salidas son de 256-bits.
3. Es eficientemente computable. Intuitivamente esto significa que dada una cadena de entrada, uno puede encontrar cuál es la salida de la función en un tiempo razonable. Formalmente significa que computar el hash de una cadena de n-bits toma un tiempo que es $O(n)$.

Estas propiedades definen de manera general una función hash, nosotros vamos a enfocarnos exclusivamente en *funciones hash criptográficas*. Para que una función hash sea criptográficamente segura, se requiere que esta tenga las siguientes tres propiedades adicionales: (1) resistencia a colisiones, (2) hiding, (3) puzzle-friendliness.

Resistencia a colisiones. Formalmente, una función hash H se dice que es resistente a colisiones si no es factible computacionalmente encontrar dos valores distintos x, y tales que $H(x) = H(y)$. Note que se dijo que no es factible computacionalmente encontrar una colisión, pero no se dijo que no existan colisiones. En realidad, las colisiones existen, y uno puede probar esto mediante un argumento de conteo. El espacio de entradas contiene todas las cadenas de todas las longitudes, y el espacio de salida contiene sólo cadenas de longitud fija. Ya que el espacio de entrada es más grande que el espacio de salida, deben haber cadenas de entrada que tienen la misma cadena de salida.

Para más información y aplicaciones de las propiedades (1) y (2) véase por ejemplo [2], p. 2-8.

2.2. Punteros hash y estructura de datos

Un **puntero hash** es simplemente un puntero donde se almacena cierta información junto con un hash criptográfico de la información. Mientras que un puntero regular brinda una forma de recuperar la información, un puntero hash también da una forma de verificar que la información no ha sido cambiada.

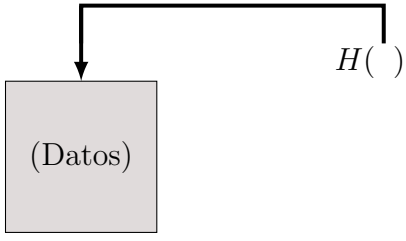

Figura 2.1. Un puntero hash es simplemente un puntero donde se almacena cierta información junto con un hash criptográfico de la información.

Podemos usar punteros hash para construir toda clase de estructura de datos. Intuitivamente, podemos tomar una estructura de datos que use punteros como una lista vinculada o un árbol de búsqueda binario e implementarla con punteros hash, en vez de usar punteros como normalmente lo haríamos.

En la figura 2.2, se muestra una lista vinculada usando punteros hash. Vamos a llamar a esta estructura de datos una **cadena de bloques**. Mientras que en una lista vinculada regular donde se tiene una serie de bloques, cada bloque tiene información, así como un puntero al bloque previo de la lista, en una cadena de bloques el puntero del bloque anterior será remplazado por un puntero hash. Así, cada bloque no sólo nos dice dónde estaba el valor del bloque anterior, sino que también contiene un resumen de ese valor que nos permite verificar que el valor no ha sido cambiado.

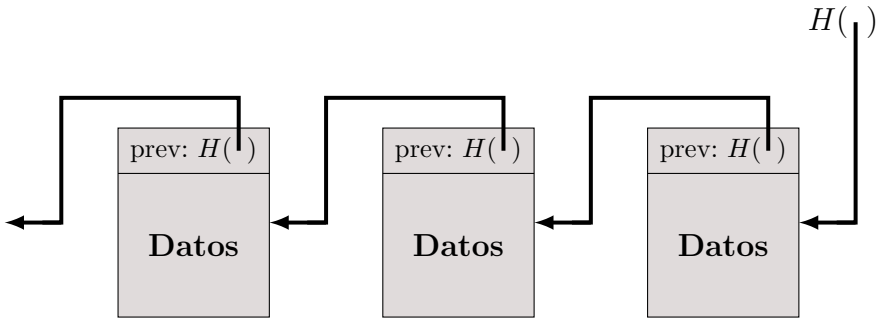

Figura 2.2. Una cadena de bloques es una lista vinculada y que está construida con hash pointers.

Un uso para una cadena de bloques es un registro a prueba de manipulaciones. Es decir, queremos construir una estructura de datos de registro que almacene una gran cantidad de datos, y nos permita agregar datos al final del registro, pero además si alguien altera los datos que están antes en el registro, deseamos poder detectarlo.

Para comprender por qué una cadena de bloques logra esta prueba de manipulaciones, veamos que pasa si un adversario quiere alterar los datos que están en medio de la cadena. Específicamente, el objetivo del adversario es hacerlo de tal manera que alguien que sólo recuerde el puntero hash en la cabeza de la cadena de bloques no pueda detectar la alteración. Para lograr este objetivo, el adversario cambia los datos de algún bloque k. Dado que los datos han sido cambiados, el hash en el bloque $k+1$, el cual es el hash del bloque k original no va a coincidir. Recordemos que estamos garantizando estadisticamente que el nuevo hash no coincidirá con el hash que está en el bloque $k+1$, ya que la función es resistente a colisiones, de esta manera detectaremos la inconsistencia entre los nuevos datos en el bloque k y el puntero hash en el bloque $k+1$. Es claro que el adversario puede continuar cambiando lo datos de los siguientes bloques, pero esta estrategia fallará cuando él alcanze la cabeza de la lista. Específicamente, siempre y cuando almacenemos el puntero hash en la cabeza de la lista en un lugar donde el adversario no pueda cambiarlo, el adversario no podrá cambiar ningún bloque sin ser detectado (ver figura 2.3).

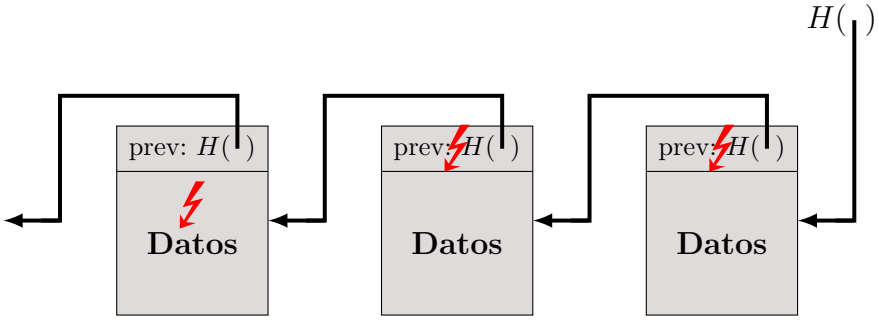

Figura 2.3. Si un adversario modifica los datos en algún bloque de la cadena de bloques, resultará que el hash en el siguiente bloque no coincidirá con el hash del bloque que se ha modificado. Si guardamos el puntero hash de la cabeza de la lista, aún si el adversario modifica todos los siguientes punteros hash para ser consistente con los datos modificados, el puntero hash de la cabeza de la lista será incorrecto, y entonces detectaremos las alteraciones.

2.3. Firmas digitales

Una firma digital supone ser el análogo digital de una firma manuscrita en papel. Deseamos dos propiedades de las firmas digitales que se correspondan bien con la analogía de las firmas manuscritas. En primer lugar, sólo Usted puede hacer su firma, pero cualquiera que la vea puede verificar que sea válida. En segundo lugar, queremos que la firma este vinculada a un documento particular para que la firma no pueda usarse para indicar su consentimiento o respaldo de un documento diferente. Para las firmas manuscritas, esta última propiedad es análoga a asegurar que ninguna persona pueda tomar su firma, cortarla de un documento y pegarla en la parte inferior de otro.

¿Cómo podemos construir esto en forma digital usando criptografía? Hagamos que la discusión intuitiva anterior se un poco más concreta, esto nos permitirá razonar mejor sobre los esquemas de firma digital.

Esquema de firma digital. Un esquema de firma digital consta de los siguientes tres algoritmos:

1. (s_k, p_k) := **generateKeys**(***keysize***). El método **generateKeys** toma un tamaño de clave y genera un par de claves s_k y p_k. La clave

privada s_k se mantiene en privado y se usa para firmar mensajes. p_k es la clave de verificación pública que le das a todos, cualquiera con esta clave puede verificar tu firma.

2. **sig := sign(s_k, *message*)**. El método **sign** toma un mensaje y una clave privada s_k como entrada y emite una firma para el mensaje bajo s_k.

3. **isValid := verify(p_k, *message, sig*)**. El método **verify** toma una clave pública, un mensaje, y una firma como entrada. Devuelve un valor booleano IsValid, que será verdadero si sig es una firma válida para el mensaje bajo la clave pública p_k, y es falso en otro caso.

Requerimos que se cumplan las siguientes dos propiedades. Primero las firmas válidas deben verificar

$$\textbf{verify}(p_k,\ message,\ \textbf{sign}(s_k,\ message)) == \text{verdadero}.$$

Es decir, si firmo un mensaje con s_k mi clave privada, y luego alguien intenta validar esa firma sobre ese mismo mensaje usando mi clave pública p_k, la firma debe validarse correctamente. Esta propiedad es un requisito básico para que las firmas digitales sean útiles. El segundo requisito es que es computacionalmente inviable falsificar firmas. Es decir, un adversario que conoce tu clave pública y puede ver tus firmas en otros mensajes no puede falsificar tu firma en algún mensaje en el cual no ha visto tu firma.

Los métodos criptográficos garantizan que esa pareja de claves s_k y p_k sólo se puedan generar una vez, de modo que se puede asumir que no es posible que dos personas hayan obtenido casualmente la misma pareja de claves. Si el propietario del par de claves usa su clave privada para cifrar un mensaje, cualquiera puede descifrarlo utilizando la clave pública del primero. En este caso se consigue la identificación y autentificación del remitente, ya que se sabe que sólo pudo haber sido él quien empleó su clave privada (salvo que un tercero la haya obtenido). Esta idea es el fundamento de la firma digital, donde jurídicamente existe la presunción de que el firmante es efectivamente el dueño de la clave privada.

En el caso de Bitcoin se utiliza un esquema de firma digital particular llamado Elliptic Curve Digital Signature Algorithm (ECDSA). Este algoritmo ha recibo un considerable análisis criptográfico, y en general, se cree que es seguro.

2.4. Claves públicas como identidades

Veamos un buen truco que acompaña a las firmas digitales. La idea es tomar una clave pública, una de esas claves de verificación pública de un esquema de firma digital y equipararlo a la identidad de una persona o actor en un sistema. Si se ve un mensaje con una firma que se verifica bajo una clave pública p_k, entonces podemos pensar esto como que p_k está diciendo el mensaje. Literalmente podemos pensar a una clave pública como un actor o individuo del sistema quien puede hacer declaraciones firmando esas declaraciones. Desde este punto de vista, la clave pública es una identidad. Así, para que alguien hable bajo la identidad p_k debe conocer la clave privada correspondiente s_k.

Una consecuencia de tratar a las claves públicas como identidades es que nosotros podemos crear una nueva identidad cuando lo deseemos, simplemente creando un nuevo par de claves s_k y p_k, a través de la operación **generateKeys** en nuestro esquema de firma digital, p_k es la nueva identidad pública que podemos usar, y s_k es la clave privada correspondiente que sólo nosotros conocemos, y nos permite hablar bajo la identidad p_k. En la practica, podemos usar el hash de p_k como identidad ya que las claves públicas son largas. Si hacemos eso, entonces para verificar que un mensaje proviene de nuestra identidad, uno tendría que verificar, primero que el hash de p_k es en efecto nuestra identidad, y segundo, que el mensaje se verifica bajo la clave pública p_k.

Esto nos lleva a la **gestión descentralizada de identidad**. En lugar de tener una autoridad central a la que tiene que acudir para registrarse como usuario en un sistema, usted puede registrarse como usuario por si mismo. No necesita que se le emita un nombre de usuario, ni informar a alguien que va a usar un nombre particular. Si desea una nueva identidad, puede generar una en cualquier momento, y puede crear tantas como desee. Si quiere ser conocido por cinco nombre diferentes, ¡no hay problema!, solo debe crear cinco identidades. Si desea permanecer en el anonimato por un tiempo, puede crear una nueva identidad, usarla por un tiempo y luego desecharla. Todas esas cosas son posibles con la gestión descentralizada de identidad, y esta es la forma en que Bitcoin, de hecho, hace la identidad. Estas identidades se llaman ***direcciones*** en el contexto de Bitcoin y criptomonedas.

A primera vista, puede parecer que la gestión descentralizada de identidad

conduce a un gran anonimato y privacidad. Después de todo, uno puede crear una nueva identidad de aspecto aleatorio por si mismo y sin decirle a nadie su identidad del mundo real. Pero no es tan simple, con el tiempo la identidad que se crea hace una serie de declaraciones. Las personas ven estas declaraciones, entonce pueden empezar a conectar los puntos utilizando esta serie de declaraciones. Un observador puede vincular estas cosas a lo largo del tiempo y hacer inferencias que lo lleven a cierto tipo de conclusiones.

Capítulo 3

Bitcoin

"El comercio en el internet ha venido a depender exclusivamente de instituciones financieras las cuales sirven como terceros confiables para el procesamiento de pagos electrónicos. Mientras que el sistema funciona lo suficientemente bien para la mayoría de las transacciones, aun sufren de las debilidades inherentes del modelo basado en la confianza. Transacciones completamente no reversibles no son realmente posibles, dado que las instituciones financieras no pueden evitar mediar disputas. El costo de la mediación incrementa costos de transacción, limitando el tamaño mínimo práctico por transacción y eliminando la posibilidad de pequeñas transacciones casuales, y hay un costo más amplio en la pérdida de la habilidad de hacer pagos no-reversibles por servicios no-reversibles. Con la posibilidad de revertir, la necesidad de confianza se expande. Los comerciantes deben tener cuidado de sus clientes, molestándolos pidiendo más información de la que se necesitaría de otro modo. Un cierto porcentaje de fraude es aceptado como inevitable" (S. Nakamoto, 2008).

Ante esta situación, lo que se necesitaba era un sistema de pagos electrónico basado en pruebas criptográficas en vez de confianza, pemitiendole a dos partes interesadas realizar transacciones directamente sin la necesidad de un tercero confiable.

En 2009 aparece Bitcoin una moneda electrónica descentralizada concebida por Satoshi Nakamoto. La idea central detrás del protocolo de Bitcoin es reemplazar el control centralizado de la transmisión del dinero generalmente asumido por las grandes organizaciones tales como bancos, compañias de

tarjetas de crédito y otros transmisores de dinero, por una gran red usuario-a-usuario.

En este capítulo vamos a explicar el funcionamiento de Bitcoin, veremos que el mecanismo a través del cual Bitcoin logra la descentralización es una combinación de métodos técnicos e inteligentes incentivos de ingeniería. Ya en la parte final realizaremos un análisis del doble gasto.

3.1. Centralización vs Descentralización

Descentralización es un importante concepto que no es exclusivo de Bitcoin. La noción de paradigmas competitivos de centralización versus descentralización surge en una variedad de diferentes tecnologías digitales. Para entender de mejor manera como este se desarrolla en Bitcoin, es útil entender el conflicto central; la tensión entre estos dos paradigmas en una variedad de otros contextos.

Por un lado tenemos Internet, un sistema famosamente descentralizado que ha históricamente competido y prevalecido contra servicios de información como AOL's y CompuServe's. Después está el correo electrónico (email) que en esencia es un sistema descentralizado basado en el protocolo para transferencia simple de correo (Simple Mail Transfer Protocol o SMTP), un modelo abierto. Si bien tiene competencia de sistemas de mensajería patentados como Facebook o Linkdln Mail, email ha logrado seguir siendo el predeterminado para la comunicación persona a persona en linea. En el caso de mensajería instantánea y mensajes de texto, tenemos un modelo híbrido que no puede ser categóricamente descrito como centralizado o descentralizado. Finalmente están las redes sociales, a pesar de numerosos esfuerzos realizados por los aficionados, desarrolladores y empresarios para crear alternativas al dominante modelo centralizado, sistemas centralizados como Facebook y Linkdln aún dominan este espacio.

La descentralización no es todo o nada, casi ningún sistema es puramente descentralizado o puramente centralizado. Por ejemplo, email es fundamentalmente un sistema descentralizado basado en un protocolo estandarizado, SMTP, y cualquiera que lo desee puede operar un servidor de correo por su cuenta. Sin embargo lo que ha sucedido en el mercado es que un pequeño número de proveedores de correo web centralizados se han vuelto dominantes.

Similarmente, mientras el protocolo de Bitcoin es descentralizado, servicios como intercambio de Bitcoin, donde uno puede convertir Bitcoin a otras monedas, y sofware de billetera, o sofware que permite administrar tus bitcoins pueden estar centralizados o descentralizados en diversos grados.

Con esto en mente, analicemos la pregunta de cómo el protocolo de Bitcoin logra la descentralización en tres preguntas más específicas; ¿Quién mantiene el libro de transacciones?, ¿Quién tiene autoridad sobre qué transacciones son validas?, ¿Quién crea nuevos bitcoins?.

Las tres preguntas anteriores reflejan los detalles técnicos del protocolo de Bitcoin, y son en estas preguntas en las que nos enfocaremos.

Diferentes aspectos de Bitcoin caen en diferentes grados de centralización/descentralización. La red usuario-a-usuario está cerca de ser puramente descentralizada ya que cualquiera puede ejecutar un nodo de bitcoin en su laptop o PC. Actualmente hay varios miles de tales nodos. El *minado* de Bitcoin, el cual será presentado más adelante, está técnicamente abierto a cualquiera, pero este requiere un costo de capital muy alto. Debido a esto, ha habido un alto grado de centralización o concentración de poder en el ecosistema de minería de Bitcoin. Muchos en la comunidad de Bitcoin ven esto como algo indeseable. Un tercer aspecto son las actualizaciones del sofware que ejecutan los nodos de Bitcoin, y esto tiene relación con cómo y cuándo cambian las reglas del sistema.

3.2. Consenso sin identidad

En esta sección estudiaremos los detalles técnicos del algoritmo de consenso de Bitcoin. Recordemos que los nodos de Bitcoin no tienen identidades persistente a largo plazo. Esta es otra diferencia con los algoritmos de consenso tradicionales. Una razón para esta falta de identidades es que en un sistema usuario-a-usuario no hay una autoridad central que asigne identidades a los participantes y verifique que no están creando nuevos nodos a voluntad. El término técnico para esto es un Sybil attack. Sybils son sólo copias de un nodo que un adversario malicioso puede crear para que parezca que hay muchos participantes, cuando en realidad todos esos pseudo-participantes están controlados por el mismo adversario. Otra razón es que el anonimato es inherentemente un objetivo de Bitcoin. Incluso si fuera fácil establecer identidades

para todos los nodos o todos los participantes no necesariamente querriamos hacer eso. Aunque Bitcoin no ofrece garantías sólidas de anonimato en el sentido de que las diferentes transacciones que uno realiza, a menudo se pueden vincular, tiene la propiedad de que nadie se ve obligado a revelar su identidad real o su dirección IP. Y esa es una propiedad importante y una característica central del diseño de Bitcoin.

Si los nodos tuvieran identidades el diseño sería más fácil. Si los nodos estuvieran identificados y no fuera trivial crear nuevos nodos, entonces podríamos hacer deducciones sobre el número de nodos maliciosos y podríamos derivar propiedades de seguridad. Es por ello que la falta de identidades introduce dificultades para el protocolo de consenso de Bitcoin.

Podemos compensar la fata de identidades haciendo una suposición. Asumamos que de alguna manera existe la posibilidad de elegir un nodo de forma aleatoria en el sistema. Una buena analogía para esto es una lotería o un sorteo, o cualquier sistema de la vida real donde es difícil rastrear a las personas, darles identidades y verificar esas identidades. Lo que hacemos en esos contextos es dar fichas o boletos, o algo similar. Eso luego nos permite elegir una ficha de manera aleatoria y llamar al propietario de esa ficha. Así que por el momento demos un salto de fe y asumamos que es posible elegir un nodo de forma aleatoria en el sistema de Bitcoin. Supongamos además por el momento que este algoritmo de generación y distribución de fichas es lo suficientemente inteligente como para que si el adversario intenta crear muchos nodos Sybil, esos Sybils juntos solo conseguirán una ficha. Eso significa que el adversario no es capaz de multiplicar su poder creando nuevos nodos. Podríamos pensar que estamos suponiendo demasiado, pero más adelante mostraremos con detalle cómo ciertas propiedades similares a estas se llevan a cabo en Bitcoin.

Esta suposición de seleccionar un nodo de forma aleatoria hace posible algo llamado **consenso implícito**. Hay múltiples rondas en nuestro protocolo, cada una corresponde a diferentes bloques en la cadena de bloques. En cada ronda se selecciona de alguna manera un nodo aleatorio, y este nodo elegido propone unilateralmente cuál será el próximo bloque en la cadena de bloques. ¿Pero qué pasa si el nodo es malicioso? Bueno, hay un proceso para manejar eso. Otros nodos aceptarán o rechazarán implícitamente ese bloque eligiendo si construyen sobre dicho bloque o no. Si ellos aceptan ese bloque, entonces ellos señalarán su aceptación extendiendo la cadena de bloques que

incluye a dicho bloque. Por el contrario, si rechazan ese bloque, entonces ignorarán ese bloque y construirán sobre el bloque anterior que aceptaron. Recordemos que cada bloque contiene el hash del bloque que este extiende. Este mecanismo técnico permite a los nodos señalar cuál bloque es el que están extendiendo.

Algoritmo de consenso de Bitcoin (simplificado)

Este algoritmo es simplificado porque asume la capacidad de seleccionar un nodo aleatorio de manera que no sea vulnerable a Sybil attacks.

1. Nuevas transacciones son compartidas a todos los nodos.
2. Cada nodo reúne nuevas transacciones en un bloque.
3. En cada ronda un nodo aleatorio consigue compartir su bloque.
4. Otros nodos aceptan el bloque sólo si todas las transacciones en esta son válidas (monedas no gastadas, firmas válidas).
5. Los nodos expresan su aceptación del bloque incluyendo su hash en el siguiente bloque que ellos crean.

Tratemos de entender cómo trabaja este algoritmo de consenso. Para esto consideremos cómo un adversario, a quien llamaremos Alice, puede ser capaz de subvertir este proceso.

Robar bitcoins. ¿Puede Alice simplemente robar bitcoins que le pertenecen a otro usuario de una dirección que ella no controla?. No. Incluso si es el turno de Alice para proponer el siguiente bloque en la cadena, ella no puede robar los bitcoins de otro usuario. Hacer eso requeriría crear una transacción válida que gaste esas monedas. Esto requeriría que Alice falsificara la firma del propietario, lo que no puede hacer si se usa un esquema de firma digital seguro. Entonces, siempre que la criptografía subyacente sea sólida, ella no será capaz de robar bitcoins.

Ataque de denegación de servicio. Consideremos otro ataque. Digamos que Alice realmente no le agrada otro usuario llamado Bob, entonces

Alice puede elegir no incluir ninguna transacción que se origine de la dirección de Bob, en ningún bloque que ella proponga ingresar a la cadena de bloques. En otras palabras, ella le está denegando el servicio a Bob. Si bien este es un ataque que Alice puede intentar montar, afortunadamente no es más que una molestia menor. Si la transacción no está en el siguiente bloque que Alice propone, Bob sólo tiene que esperar hasta que un nodo honesto tenga la oportunidad de proponer un bloque con dicha transacción. Así que esto tampoco es un buen ataque.

Ataque de doble gasto. Alice puede intentar hacer un ataque de doble gasto. Para comprender cómo funciona esto, supongamos que Alice es cliente de algún comerciante en linea o sitio web administrado por Bob, quién proporciona algunos servicios en linea a cambio de un pago en bitcoins, digamos que el servicio de Bob es permitir la descarga de algún software. Así es como podría funcionar un ataque de doble gasto. Alice agrega un articulo a su carrito de compras en el sitio web de Bob y el servidor solicita el pago. Luego Alice crea una transacción de Bitcoin desde su dirección a la de Bob, y la transmite a la red. Digamos que algún nodo honesto crea el siguiente bloque e incluye esa transacción en dicho bloque. Entonces ahora hay un bloque creado por un nodo honesto que contiene una transacción que representa el pago de Alice al comerciante Bob.

Recordemos que una transacción es una estructura de datos que contiene la firma de Alice, una instrucción para pagar a la clave pública de Bob y un hash. Este hash representa un puntero a una salida de transacción anterior que Alice recibió y ahora está gastando. Ese puntero debe hacer referencia a una transacción que se incluyó en algún bloque anterior en la cadena de consenso.

Es importante tener en cuenta que hay dos tipos diferentes de punteros hash aquí que pueden confundirse fácilmente. Los bloques incluyen un puntero hash al bloque anterior que están extendiendo. Las transacciones incluyen uno o varios punteros hash a la salida de transacciones anteriores que se están canjeando.

Volvamos a cómo Alice puede realizar un ataque de doble gasto. El último bloque fue generado por un nodo honesto e incluye la transacción en la que Alice paga a Bob por la descarga del software. Al ver esta transacción incluida en la cadena de bloques, Bob concluye que Alice le ha pagado y le permite descargar el software. Supongamos que el siguiente nodo aleatorio que se

selecciona en la próxima ronda pasa a estar controlado por Alice. Ahora dado que Alice puede proponer el siguiente bloque, podría proponer un bloque que ignore el bloque que contiene el pago a Bob, y en su lugar contiene un puntero al bloque anterior. Además en el bloque que propone, Alice incluye una transacción en la que transfiere las monedas que estaba enviando a Bob a una dirección que ella misma controla (ver figura 3.1). Este es un patrón clásico de doble gasto. Como las dos transacciones gastan las misma monedas, sólo una de ellas puede incluirse en la cadena de bloques. Así, si Alice logra incluir el pago a su propia dirección en la cadena de bloques, entonces la transacción en la que paga a Bob es inútil, ya que nunca puede incluirse más tarde en la cadena de bloques.

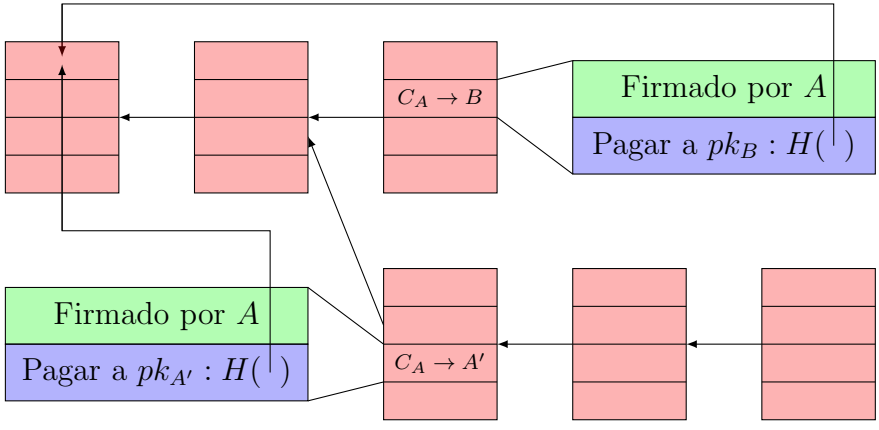

Figura 3.1. **Un intento de doble gasto**. Alice crea dos transacciones. Una en la que envía bitcoins a Bob ($C_A \to B$), y una segunda en la que gasta dos veces esas mismas monedas enviandolos a una dirección que ella controla ($C_A \to A'$). Como gastan los mismos bitcoins, sólo una de esas dos transacciones puede incluirse en la cadena de bloques. Las flechas son punteros de un bloque al bloque anterior que extiende, incluyendo el hash del bloque anterior dentro de su propio contenido.

Y ¿Cómo sabemos si este intento de doble gasto va a tener éxito o no? Bueno, eso depende de que bloque finalmente terminará en la cadena de consenso a largo plazo ¿Qué determina que bloque se incluirá? Los nodos honestos siguen la politica de extender la rama válida de mayor longitud, así

que ¿Cuál rama extenderán ellos? No hay respuesta correcta, en este punto las dos ramas tiene la misma longitud, sólo difieren en el último bloque y ambos bloques son válidos. El nodo que proponga el siguiente bloque luego pude decidir construir sobre cualquiera de los dos, y esta elección puede determinar en gran medida si el doble gasto tiene éxito o no.

Desde un punto de vista moral, existe una clara diferencia entre el bloque que contiene la transacción en el que Alice le paga a Bob y el bloque que contiene la transacción en el que Alice gasta esas monedas dos veces. Pero esta distición solo se basa en nuestro conocimiento de la historia en la cual Alice primero le pago a Bob y luego intentó hacer un doble gasto. Sin embargo, desde un punto de vista tecnológico estas dos transacciones son completamente idénticas y ambos bloques son válidos. Los nodos que están viendo esto, realmente no tiene forma de saber cuál es la transacción moralmente legítima.

En la práctica, los nodos a menudo sigue la heuristica de extender el bloque sobre el que escucharon primero, pero esto no es una regla sólida debido a la latencia de la red. Podría pasar que el bloque del que un nodo se entero primero sea en realidad el creado en segundo lugar. Por lo tanto existe al menos una posibilidad de que el siguiente nodo que proponga un bloque extienda el bloque que contiene el doble gasto. Alice podría aumentar la probabilidad de que suceda sobornando al siguiente nodo. Si el siguiente nodo construye sobre el bloque que contiene el doble gato por alguna razón, entonces ahora esta cadena será la cadena más larga que la que incluye la transacción a Bob. En este punto es mucho más probable que el siguiente nodo honesto continue desarrollandose en esta cadena, ya que es más larga. Este proceso continuará y será cada vez más probable que el bloque que contiene el doble gasto sea parte de la cadena de consenso a largo plazo. Por otro lado el bloque que contiene la transacción a Bob está completamente abandonado, y ahora dicho bloque es llamado un **bloque huérfano**.

Ahora consideremos toda esta situación desde el punto de vista de Bob. Comprender cómo Bob puede protegerse de este ataque de doble gasto es una parte clave para entender la seguridad de Bitcoin. Cuando Alice comparte la transacción que representa su pago a Bob, él está al pendiente en la red y se entera incluso antes de que un bloque con esta transacción sea creado. Si Bob fuera más insensato de lo que describimos anteriormente permitiría que Alice descargara el software en ese momento. Recordemos que en este caso no se ha

creado un bloque que contenga la transacción en la que Alice le paga a Bob, se dice entonces que esta transacción tiene **cero confirmaciones**. Esto lleva a un ataque de doble gasto aún mas básico que el descrito anteriormente. Antes, para que ocurriera el ataque de doble gasto, teníamos que suponer que un actor malicioso controla el nodo que propone el siguiente bloque. Pero si Bob permite que Alice descargue el software antes de que la transacción reciba una sola confirmación, entonces Alice puede compartir inmediatamente una transacción de doble gasto, y un nodo honesto podría incluirla en el siguiente bloque en vez de la transacción en la que se paga a Bob.

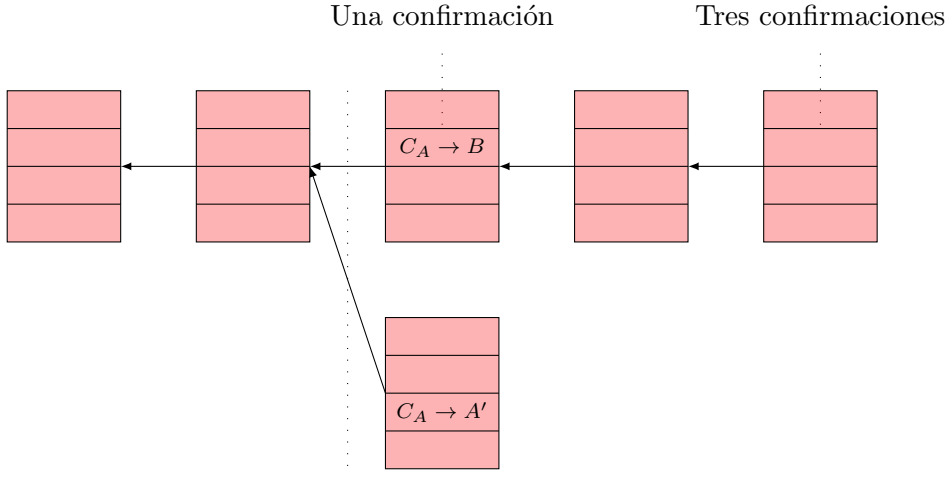

Figura 3.2. Así es como se ve el intento de doble gasto de Alice desde el punto de vista de Bob. Para protegerse de este ataque, Bob debe esperar hasta que la transacción este incluida en la cadena de bloques y tenga varias confirmaciones.

Por otro lado, un comerciante cauteloso no le permitiría descargar el software a Alice incluso después de que la transacción estuviese contenida en un bloque, y continuaría esperando. Si Bob ve que el ataque de doble gasto ha sido exitoso, él se da cuenta que el bloque que contenía la transacción para él ha quedado huérfano, y entonces no le permite a Alice descargar el software. En cambio, si a pesar del intento de doble gasto, varios nodos construyen sobre el bloque que contiene la transacción Alice $\to$ Bob y dicha

transacción recibe varias confirmaciones, entonces Bob ganan confianza de que esta transacción estará a largo plazo en la cadena de consenso, y entonces le permite descargar el software (ver figura 3.2).

En general cuantas más confirmaciones obtenga una transacción mayor será la probabilidad de que termine en la cadena de consenso a largo plazo. Recordemos que el comportamiento de los nodos honestos es siempre extender la cadena válida más larga. De hecho, la probabilidad del doble gasto disminuye exponencialmente con el número de confirmaciones. Entonces si la transacción que interesa ha recibo k confirmaciones,, entonces la probabilidad de que una transacción de doble gasto termine en la cadena de consenso a largo plazo disminuye exponencialmente en función de k. La heurística más común que se usa en Bitcoin es esperar seis confirmaciones.

En resumen, la protección contra transacciones inválidas es completamente criptográfica. Pero se aplica por consenso, lo que significa que si un nodo intenta incluir una transacción criptográficamente inválida, entonces la única razón para que la transacción no termine en la cadena de consenso a largo plazo es porque la mayoría de los nodos son honestos. Por otro lado, la protección contra el doble gasto es puramente por consenso, la criptografía no tiene nada que decir al respecto, ya que la transacción legítima y la transacción que incluye el doble gasto, ambas son válidas desde un punto de vista critográfico, pero el consenso es el que determina cuál bloque terminará en la cadena de consenso a largo plazo. Y finalmente, nunca estás cien por ciento seguro de que una transacción en la que estas interesado terminará en la cadena de consenso, pero esta garantía de probabilidad exponencial es bastante buena. Después de aproximadamente seis confirmaciones, prácticamente no hay posibilidades de que las cosas salgan mal.

3.3. Incentivos y prueba de trabajo

Hay dos mecanismos de incentivos separados en Bitcoin. El primero es la **recompensa por bloque**. De a cuerdo con las reglas de Bitcoin, el nodo que crea un bloque puede incluir una transacción especial en ese bloque. Esta transacción es una transacción de creación de moneda, y el nodo también puede elegir la dirección del destinatario de esta transacción. Claro esta que el nodo generalmente elegirá una dirección que le pertenezca a él mismo. Uno

puede pensar esto como una pago al nodo a cambio del servicio de crear un bloque en en el tope de la cadena de consenso. Es importante resaltar que esta es la única manera en la que se pueden crear bitcoins.

Bitcoin es una moneda deflacionaria, eso quiere decir que al igual que el oro su sumistro es cada vez más escaso y difícil de extraer. Satoshi Nakamoto decidió regular el suministro reduciendo a la mitad el número de bitcoins que los mineros podrían adquirir cada 210,000 bloques, que es cada cuatro años con las condiciones actuales. Cuando se lanzó Bitcoin a comienzos del 2009, la recompensa por minar un bloque era de 50 bitcoins, misma que se redujo a la mitad (25 BTC) el 28 de noviembre de 2012 después de que se extrajera el bloque 210,000, y así se ha continuado hasta la recompensa actual de 12.5 BTC por bloque. Este ciclo seguirá hasta que se hayan extraído los 21 millones de bitcoins disponibles.

Quizás se este preguntando por qué la recompensa por bloque incentiva el comportamiento honesto. Puede parecer con base a lo que hemos dicho hasta ahora que el nodo obtiene la recompensa independientemente si propone un bloque válido o si se comporta de manera maliciosa. Pero esto no es verdad, recordemos cómo tal nodo obtiene su recompensa, eso sólo pasará si el bloque en cuestión está al final de la cadena de consenso (la cadena más larga), ya que como cualquier otra transacción, la transacción de creación de moneda sólo será aceptada por otros nodos si esta está en el último bloque de la cadena más larga. Esta es la idea clave detrás de este mecanismo de incentivo. Es un truco sutil pero muy poderoso. Esto incentiva a los nodos a comportarse de la manera que creen que hará que otros nodos extiendan sus bloques. Así que si la mayoría de la red está siguiendo la regla de la cadena más larga, esto incentiva a todos los nodos a seguir esa regla.

El segundo mecanismo de incentivo es llamado **tarifa de transacción**. El creador de cualquier transacción puede escoger hacer el valor total de la salida menor que el valor de su entrada. Así, quién crea primero el bloque coloca esa transacción en la cadena de bloques y cobra la diferencia, lo cual representa una tarifa de transacción. De esta manera, si un nodo creó un bloque que contenía digamos 200 transacciones, entonces la suma de todas las tarifas de esas transacciones se paga a la dirección que el nodo ingresó en el bloque. La tarifa de transacción es puramente voluntaria, pero se espera, dada nuestra comprensión del sistema, que a medida que la recompensa por bloque empiece a agotarse, será cada vez más importante para los usuarios

incluir tarifas de transacción, con el objetivo de conseguir una calidad más rasonable de servicio. Hasta cierto punto, esto ya está empezando a suceder, pero aún no está claro como se desarrollará el sistema.

Todavía quedan algunos problemas con el mecanismo de consenso tal como lo describimos. El primero es el salto de fe acerca de que es posible de alguna manera elegir un nodo aleatorio. En segundo lugar, hemos creado un nuevo problema al dar a los nodos estos incentivos para participar, ya que el sistema podría volverse inestable al momento de que todos quieran ejecutar un nodo de Bitcoin con la esperanza de capturar alguna de estas recompensas. Y tercero, es que un adversario podría tratar de crear varios nodos Sybil para tratar de subvertir este proceso.

Minería y prueba de trabajo. Resulta que los problemas anteriores están relacionados y todos tienen la misma solución que se llama **prueba de trabajo**. La idea detrás de prueba de trabajo es que aproximamos la selección de un nodo aleatorio seleccionando nodos en proporción de un recurso que esperamos que nadie pueda monopolizar. Si por ejemplo este recurso es poder computacional, entonces es un sistema de prueba de trabajo. Alternativamente, podría ser proporcional a la propiedad de moneda, y eso se llama prueba de participación.

Pero volviendo a prueba de trabajo. Tratemos de tener una mejor idea de lo que significa seleccionar un nodo en proporción a su poder computacional. Otra forma de entender esto es que estamos permitiendo que los nodos compitan entre sí mediante el uso de su poder de computo, y eso dará como resultado que los nodos se seleccionen automáticamente en esa proporción. Otra visión más de prueba de trabajo es que estamos haciendo moderadamente más difícil crear nuevas identidades. Es una especie de impuesto sobre la creación de identidad y por lo tanto sobre el ataque Sybil. Todo esto puede parecer un poco vago, así que veamos los detalles de prueba de trabajo que es usado en Bitcoin, lo cual debería de aclarar las cosas mucho más.

Bitcoin logra prueba de trabajo utilizando **hash puzzles**. Para crear un bloque se requiere que el nodo que propone ese bloque encuentre un número, o **nonce**, de modo que cuando concatene el nonce, el hash del bloque que está extendiendo, y la lista de transacciones que componen ese bloque, y luego tome el hash de toda esa cadena, la salida del hash sea un número que está en un espacio objetivo que es bastante pequeño en relación con el espacio de salida de esa función hash. Podemos definir dicho espacio objetivo como el

conjunto de valores que son menores o iguales a un **valor objetivo**, digamos m. En este caso, el nonce deberá satisfacer la siguiente desigualdad:

$$H(nonce \mid\mid prev\_hash \mid\mid t_1 \mid\mid t_2 \mid\mid, \ldots, \mid\mid t_k) < m. \tag{3.3.1}$$

Como vimos anteriormente un bloque contiene una serie de transacciones. Además, un bloque contiene un puntero hash al bloque anterior. Y ahora estamos también requiriendo que cada bloque contenga un nonce. La idea es que queremos que sea moderadamente difícil encontrar un nonce que satisfaga la propiedad requerida. Si la función hash satisface la propiedad Puzzle friendliness, entonces la única forma de tener éxito es probar suficientes nonces uno por uno hasta que tengamos suerte.

Esta noción de hash puzzles y prueba de trabajo elimina por completo el requisito de elegir mágicamente un nodo aleatorio. En lugar de esto, los nodos simplemente compiten independientemente para resolver estos hash puzzles todo el tiempo. De vez en cuando uno de ellos tendrá suerte y encontrará un nonce aleatorio que satisfaga la propiedad antes descrita, y entonces tal nodo propondrá el siguiente bloque. Así es como el sistema está completamente descentralizado. Nadie decide qué nodo es el que propone el siguiente bloque.

Hay dos propiedades importantes de los hash puzzles. La primera es que deben ser difíciles de calcular. Debido a que crear un bloque requiere una gran cantidad de computo, sólo algunos nodos se molestan en competir en este proceso de creación de bloques. Este proceso de intentar resolver estos hash puzzles se conoce como **minería** de Bitcoin, y los nodos participantes son llamados **mineros**. Aunque técnicamente cualquiera puede ser un minero, ha habido mucha concentración de poder en el ecosistema de minería debido a su alto costo.

La segunda propiedad es que el costo sea parametrizable, no un costo fijo todo el tiempo. La manera en que se logra esto es que todos los nodos en la red de Bitcoin automáticamente vuelven a calcular el valor objetivo cada 2016 bloques. Ellos recalculan el valor objetivo de tal manera que el tiempo de creación entre dos bloques consecutivos producidos en la red de Bitcoin sea de unos 10 minutos en promedio. Esto implica que se recalcula el valor objetivo aproximadamente cada dos semanas. Pensemos que significa esto, si nosotros somos mineros y hemos invertido una cierta cantidad fija de hardware en la minería de Bitcoin, pero el sistema minero está creciendo, entonces más mineros se están uniendo a la red de Bitcoin, o se está implementando

hardware más potente. En consecuencia, durante un periodo de dos semanas se encontrarán un poco más de bloques de lo esperado. Entonces los nodos reajustarán automáticamente el valor objetivo, y la cantidad de trabajo que tenemos que hacer para encontrar un bloque va a incrementar. Así, si nosotros invertimos una cantidad fija de hardware la tasa a la cual encontraremos bloques depende de lo que están haciendo los otro nodos. Específicamente, si el poder computacional total de la red de Bitcoin es H, y nosotros tenemos un poder computacional de pH $(0 < p \leq 1)$, entonces la probabilidad de que encontremos el próximo bloque es $\frac{pH}{H} = p$. Eso significa que si tenemos el 0.1 por ciento del poder computacional global, entonces encontraremos uno de cada 100 bloques en promedio.

Resolver hash puzzles es algo probabilistico porque nadie puede predecir qué nonce resolverá el problema. La única forma de hacerlo es probar nonces uno por uno y esperar que uno tenga éxito. Este proceso de ir probando nonces es un proceso de ensayos Bernoulli, aquí los dos posibles resultados son si el hash cae o no en el conjunto objetivo. Por lo general los nodos prueban tantos nonces que los ensayos Bernoulli, un proceso de probabilidad discreta, pueden aproximarse bien mediante un proceso Poisson [2]. Es por esta razón que el capítulo 4 se modela la generación de bloques de un nodo en particular mediante un proceso Poisson.

3.4. Análisis del doble gasto

Las transacciones de Bitcoin son agrupadas en bloques. Cada bloque referencia un bloque anterior incluyendo el hash de dicho bloque en su cabecera. La única excepción es el primer bloque creado, conocido como el bloque génesis. Por lo tanto los bloques forman un árbol de bloques, con el bloque génesis como su raíz, y cada bloque siendo un hijo del bloque que referencia. Una rama en este árbol es un camino de una hoja al bloque génesis, de tal manera que cada rama representa una versión de la historia de las transacciones de Bitcoin. Cada rama debe ser consistente y nunca puede incluir dos transacciones conflictivas, sin embargo, las ramas no necesitan ser consistentes una con la otra, es decir, una rama puede incluir una transacción la cual contradice una transacción en la otra rama [6].

Ya que se desea una única historia coherente de las transacciones, los nodos siempre considerarán la cadena más larga como la correcta y empiezan a trabajar en extenderla. Si dos nodos emiten versiones diferentes del próximo bloque simultáneamente, algunos nodos puede que reciban uno o el otro primero. En ese caso trabajan en el que reciban primero, pero guardan la otra rama en caso de que esta se vuelva más larga. El empate se rompe cuando la próxima prueba de trabajo es encontrada y una rama se vuelve más larga, los nodos que estaban trabajando en la otra rama luego se cambian a la más larga (ver figura 3.3).

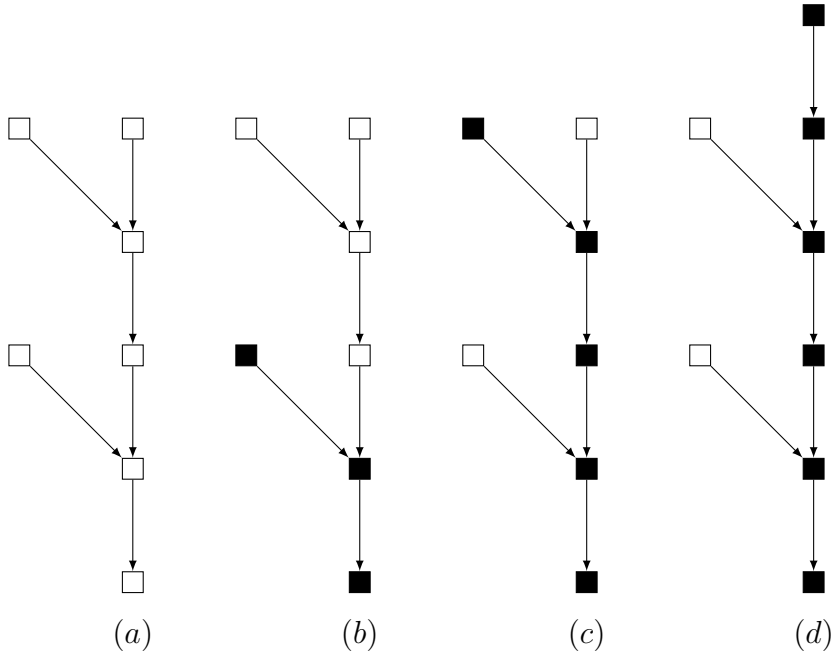

Figura 3.3. (*a*) Una posible estructura de árbol de bloques en algún punto del tiempo. (b) La rama marcada es inválida porque hay otras ramas más largas. (c) Una rama la cual está empatada en cuanto a longitud máxima con otra, y puede ser considerada como válida por algunos nodos. (d) Si un nuevo bloque es encontrado referenciando a la hoja de la otra rama, esa llega a ser la rama más larga, y es acordado por los nodos como la cadena válida.

Una transacción se dice que tiene n confirmaciones, si esta esta incluida en un bloque el cual es parte de la cadena principal, y hay n bloques en el camino entre dicho bloque y la hoja de la cadena.

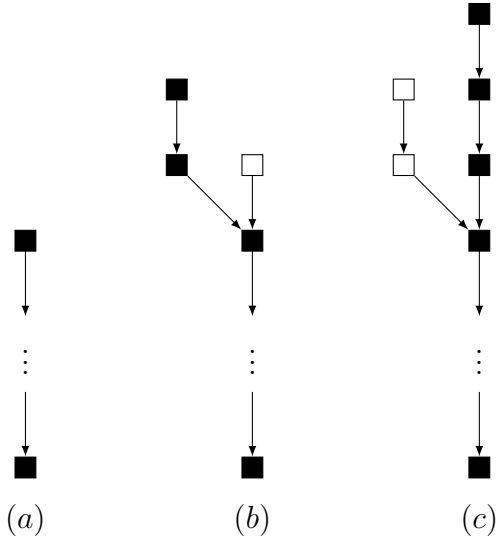

Figura 3.4. (a) El estado de la cadena de bloques cuando el ataque empieza. (b) La rama de la izquierda es conocida por la red, e incluye la transacción del pago al comerciante con dos confirmaciones. El comerciante envía el producto. Mientras tanto el atacante ha encontrado un bloque en una rama privada alterna, en la que se hace el pago a él mismo. (c) Si el atacante consigue que su rama sea más larga que la conocida por la red, él la comparte, y el pago a él mismo es aceptado por la red.

Un ataque exitoso de doble gasto consiste de los siguientes pasos (ver figura 3.4):

1. Compartir a la red una transacción en la cual es comerciante atacado es pagado.

2. Minar una rama secretamente, la cual se construye sobre el bloque anterior al bloque que contiene la transacción que quiere ser remplazada.

3. Esperar hasta que la transacción al comerciante reciba suficientes confirmaciones, y el comerciante confiado de su pago envía el producto.

4. Si es necesario continuar extendiendo la rama secreta (la cual contradice la transacción) hasta que esta sea más larga que la cadena pública (la cual incluye la transacción), después compartirla con la red. Dado que la nueva rama es más larga que la conocida por la red, esta será considerada válida, y el pago al comerciante será remplazada por el pago al propio atacante.

Al intentar un doble gasto, el atacante se encontrará él mismo típicamente en la siguiente situación. La red conoce una rama, la cual tiene n bloques enlazados de donde empezó la bifurcación a la hoja de la rama. El atacante tiene otra rama con m bloques enlazados de donde empezó la bifurcación a la hoja de dicha rama, y ambos están intentando extender sus respectivas ramas. ¿Podrá el atacante crear una cadena más larga? o ¿La brecha irá incrementando, dejando al atacante con una rama la cual nunca será válida?

Para modelar esto consideremos lo siguiente:

1. El poder computacional total entre la red honesta y el atacante es constante. La red honesta tiene una fracción $p \geq 0$ del poder computacional de la red entera, y el atacante tiene una fracción q de la misma, donde $p + q = 1$.

2. La dificultad de minado es constante, y es tal que con el poder computacional total de la red de Bitcoin, el tiempo esperado para encontrar un bloque es T_0.

Denotemos por $z = n - m$ el número de bloques de los que la red honesta tiene ventaja sobre el atacante. Siempre que un nuevo bloque es encontrado el valor de z cambia. Si el bloque fue encontrado por la red honesta entonces z incrementa en 1, y si el bloque fue encontrado por el atacante entonces z decrece en 1. Formalmente, esto es una caminata aleatoria que empieza en z y se mueve hacia la derecha con probabilidad p, y hacia la izquierda con probabilidad q. Si z alcanza algunas vez el valor -1, entonces la cadena del atacante llega a ser la más larga y su ataque tiene éxito. Si esto nunca pasa, entonces el ataque falla.

Note que cuando un bloque es encontrado, este tiene probabilidad p de haber sido encontrado por la red honesta y probabilidad q de haber sido encontrado por el atacante, y entonces este proceso puede ser resumido como

$$z_{i+1} = \begin{cases} z_i + 1 & \text{con probabilidad } p \\ z_i - 1 & \text{con probabilidad } q. \end{cases}$$

Denotemos por a_z la probabilidad de que el atacante tenga éxito estando z bloques atrás. Claramente, si $z < 0$ entonces $a_z = 1$ ya que el atacante ya tiene una cadena más larga.

Ahora note que para el caso $z \geq 0$, la probabilidad de que el atacante llegue a tener éxito es análogo al problema de la ruina del jugador en el caso infinito. Y en consecuencia $a_z = h(z+1)$. Por lo tanto

$$a_z = \begin{cases} 1 & \text{si } p \leq q \text{ o } z < 0 \\ (q/p)^{z+1} & \text{si } p > q \text{ y } z \geq 0. \end{cases} \tag{3.4.1}$$

3.5. Esperando por confirmaciones

En la practica el comerciante espera n confirmaciones del pago de su transacción, y después envía el producto. Mientras la red está creando esas n confirmaciones, el atacante está construyendo su propia rama la cual contradice dicha transacción, ¿Cuál es la probabilidad de un doble gasto exitoso?

Antes de que n confirmaciones sean obtenidas, el atacante no puede compartir su rama, aunque esta sea más larga, ya que esto disuadiría al comerciante de completar la orden. Él debe esperar n confirmaciones, y después compartir su rama si es que tiene ventaja.

La probabilidad de éxito depende de su desventaja que es el valor de z al momento en el que n confirmaciones son alcanzadas. Consideremos X el número de bloques encontrados por el atacante antes de que la red honesta encuentre su n-ésimo bloque, entonces $X \sim \text{bin neg}(n, p)$, y en consecuencia

$$\mathbb{P}(X = m) = \binom{n+m-1}{m} p^n q^m.$$

Una vez que n bloques son encontrados por la cadena honesta; en un periodo en el cual $m+1$ bloques han sido encontrados por el atacante (estamos

suponiendo que un bloque fue pre-minado por el atacante antes de comenzar el ataque), la carrera empieza con $z = n - m - 1$ bloques. Luego se sigue que la probabilidad de que se lleve a cabo un doble gasto exitoso, cuando el comerciante espera n confirmaciones es

$$\begin{aligned} r &= \sum_{m=0}^{\infty} \mathbb{P}(X = m) a_{n-m-1} \\ &= \sum_{m=0}^{n-1} \binom{n+m-1}{m} p^n q^m a_{n-m-1} + \sum_{m=n}^{\infty} \binom{n+m-1}{m} p^n q^m a_{n-m-1} \\ &= \sum_{m=0}^{n-1} \binom{n+m-1}{m} p^n q^m a_{n-m-1} + \sum_{m=n}^{\infty} \binom{n+m-1}{m} p^n q^m. \end{aligned}$$

Luego, cuando $p > q$,

$$\begin{aligned} r &= \sum_{m=0}^{n-1} \binom{n+m-1}{m} p^n q^m \left(\frac{q}{p}\right)^{n-m} + \sum_{m=n}^{\infty} \binom{n+m-1}{m} p^n q^m \\ &= \sum_{m=0}^{n-1} \binom{n+m-1}{m} p^m q^n + \sum_{m=n}^{\infty} \binom{n+m-1}{m} p^n q^m \\ &= 1 - \sum_{m=0}^{n-1} \binom{n+m-1}{m} (p^n q^m - p^m q^n) \end{aligned}$$

y cuando $q \geq p$ se tiene que $a_{n-m-1} = 1$, por lo cual $r = 1$. Por lo tanto

$$r = \begin{cases} 1 - \displaystyle\sum_{m=0}^{n-1} \binom{n+m-1}{m} (p^n q^m - p^m q^n) & \text{si } q < p \\ 1 & \text{si } q \geq p. \end{cases} \tag{3.5.1}$$

Capítulo 4

GHOST

"Bitcoin es un protocolo disruptivo para la moneda digital, la cual se basa en elementos criptográficos para asegurar su operación. Desde su lanzamiento inicial en 2009 por su misterioso creador Satoshi Nakamoto, el interés general en la moneda ha ido aumentando lentamente, y sus usos se han ido expandiendo lentamente. Mientras varios obstáculos tales como incertidumbre regulatoria y una infraestructura subdesarollada aún necesitan ser superados, los principales retos que deben ser enfrentados desde una perspectiva de ciencias de la computación están relacionados con la capacidad de Bitcoin para escalar a mayores tasas de transacción y su capacidad para procesar rápidamente transacciones individuales" (Sompolinsky y Zohar, 2015, p. 1).

La idea central detrás del protocolo de Bitcoin es reemplazar el control centralizado de la transmisión del dinero generalmente asumido por las grandes organizaciones tales como bancos, compañias de tarjetas de crédito y otros transmisores de dinero, por una gran red usuario-a-usuario. Los nodos de esta red verifican el trabajo de cada uno y por lo tanto se aseguran de que ninguna entidad sea capaz de comportarse mal. Bitcoin logra esto manteniendo un registro público y completo de todas sus transacciones en cada nodo de la red. Este libro principal el cual es conocido como la cadena de bloques esta compuesta por una secuencia creciente de bloques, cada una contiene un conjunto de la transacciones aprobadas. El principal desafío que Bitcoin supera es la sincronización del libro principal entre varios de los nodos. Terceros maliciosos quizás intenten interferir con esta sincronización con el objetivo del doble gasto para redirigir pagos previamente procesados que

les permitirán a ellos usar el mismo dinero dos veces.

Para ayudar a resolver el problema de doble gasto se requiere que los bloques contengan una prueba de trabajo, el cual es computacionalmente difícil de hacer. La dificultad de esta labor se configura de manera adaptativa de manera que un bloque sea creado aproximadamente una vez cada 10 minutos en la red entera. El intervalo de 10 minutos permite a los bloques propagarse a la mayoría de los nodos antes de que otro bloque sea creado. Si un nodo recibe dos bloques conflictivos, los cuales fueron creados por distintos nodos que desconocen el trabajo de cada uno (o quizás por un atacante malicioso), este resuelve el conflicto seleccionando el bloque correspondiente a la cadena de bloques más larga y adoptándola. El análisis original de Satoshi Nakamoto muestra que mientras el atacante tenga menos del 50 % del poder computacional de la red entera, la probabilidad de que el ataque de doble gasto sea exitoso decrece exponencialmente con el tiempo, lo cual esencialmente permite a los pagos ser considerados aceptados e irreversibles después de algún tiempo. El análisis, sin embargo, asume que los bloques son enviados a través de la red más rápido de lo que estos son creados, y por lo tanto no se ajusta a un escenario en el que muchas transacciones son procesadas por la red (el cual requiere la creación de bloques más grandes, que requieren más tiempo para transmitirse).

En efecto, la capacidad para el procesamiento de transacciones adicionales en Bitcoin es muy necesaria. A partir de dicimebre de 2014, la red de Bitcoin procesa alrededor de 90 mil transacciones por día [1], un número el cual ha ido lentamente creciendo, y equivale en promedio a una transacción por segundo (TPS). En contraste, el sistema de pago global de Visa manejó un informe de 150 millones de transacciones por día (aproximadamente 2000 TPS) y ha ido creciendo desde entonces. Si Bitcoin no es capaz de escalar a tasas apropiadas que coincidan con la demanda, las tarifas de transacciones aumentarán y los usuarios serán conducidos a usar otras formas de pago.

El bajo número actual de transacciones de Bitcoin se debe principalmente a su pequeña base de usuarios. Una vez que la adopción aumente, el sistema necesitará escalar para procesar transacciones a una tasa más alta, y las garantías anteriores de seguridad ya no se pueden mantenerse. Teniendo en cuenta esto, en el artículo *Secure High-Rate Transaction Proccesing in Bitcoin* [8] los investigadores Yonatan Sompolinsky y Aviv Zohar estudian que tan susceptible es el protocolo ante el doble gasto cuando más transacciones son

procesadas por segundo. Ellos hacen notar que bloques de mayor tamaño y los eventos de creación de bloques más frecuentes dan como resultado más conflictos entre bloques, lo cual reduce drasticamente el nivel de seguridad ante ataques. Para resolver estos problemas de seguridad, los investigadores antes mencionados proponen una alternativa a la regla de la cadena más larga llamada *GHOST*, la cual cambia el procedimiento para la resolución de conflictos para la cadena de bloques.

Este capítulo está dedicado al estudio de *GHOST*. Primero, veremos algunas propiedades básicas de *GHOST*. Posteriormente realizaremos un análisis de la tasa de crecimiento de la cadena principal cuando se sigue *GHOST* y cuando se sigue la regla de la cadena más larga. Ya en la parte final retomaremos el tema de doble gasto en redes con retrasos.

4.1. El modelo

Modelamos la red de Bitcoin como una gráfica dirigida $G = (V, E)$. Cada nodo v tiene alguna fracción $p_v \geq 0$ del poder computacional de la red entera: $\sum_{v \in V} p_v = 1$. Cada nodo individual genera bloques de acuerdo a un proceso Poisson con tasa $p_v\lambda$, así que la red entera genera bloques de acuerdo a un proceso Poisson con tasa λ, y si un nodo v tiene alguna fracción $p_v \geq 0$ del poder computacional de la red entera, entonces el tiempo que transcurre entre dos bloques sucesivos generados por dicho nodo es una variable aleatoria exponencial de parámetro $p_v\lambda$. Asumimos que cada arista $e \in E$ tiene un retraso d_e asociado, el cual es simplemente el tiempo que se toma para enviar un bloque a través de la arista. Para cualquier bloque B, denotamos por $time(B)$ al tiempo de creación de dicho bloque. Los bloques forman esencialmente una estructura de árbol que está enraizado en el bloque génesis - el primer bloque creado al momento del lanzamiento de Bitcoin; denotamos a la estructura de este árbol al tiempo t por $tree(t)$, y por $subtree(B)$ al subárbol enraizado en B. Finalmente la profundidad del bloque B en el árbol será denotado por $depth(B)$.

El protocolo de Bitcoin actualmente requiere que los nodos construyan nuevos bloques al final de la cadena más larga que es conocida por ellos. De acuerdo a esto, denotamos por $longest(t)$ a la hoja más profunda en $tree(t)$. El término "cadena principal" corresponderá al camino del bloque génesis

a una hoja que es seleccionada por extensión (usualmente $longest(t)$). La cadena principal es considerada por los nodos como la única versión aceptada de la historia de transacciones. Formalmente, el tiempo que le toma a la cadena principal avanzar de longitud $n-1$ a n es una variable aleatoria que denotamos por τ_n. Denotamos $\tau = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \tau_n$, y $\beta = \frac{1}{E(\tau)}$. β es la tasa de adición de bloques a la cadena principal, mientras que λ es la tasa de adición de bloques al árbol de bloques.

Para un bloque B en un árbol de bloques T, sea $subtree(B)$ el subárbol enraizado en B, y sea $children(B)$ el conjunto de hijos de B. Denotamos por $GHOST(T)$ a la política de selección del padre, definido como la salida del siguiente algoritmo:

Algoritmo 1 (Greedy Heaviest-Observed Sub-Tree)**.**
input: Block tree T

1. set $B \leftarrow$ Genesis Block
2. if $Children_T(B) = \varnothing$ then return(B) and exit
3. else update $B \leftarrow \underset{C\in Children_T(B)}{argmax} |subtree_T(C)|$
4. goto line 2

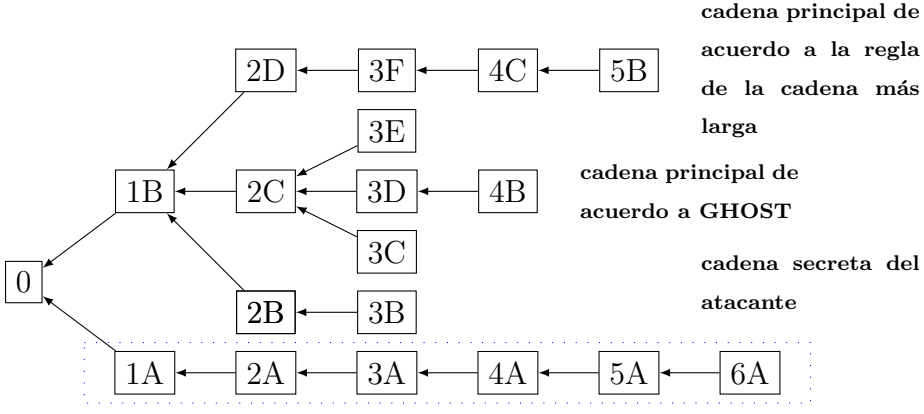

Figura 4.1. Un árbol en el cual la cadena más larga y la cadena seleccionada por GHOST difieren.

El algoritmo sigue un camino de la raíz del árbol (bloque génesis) y escoge en cada bifurcación el bloque que conduce al subárbol más pesado. En el árbol representado en la figura 4.1, por ejemplo, el subárbol enraizado en el bloque 1B contiene 12 bloques, mientras que el de 1A contiene sólo 6. El algoritmo escogerá a 1B como bloque perteneciente a la cadena principal y luego procederá a resolver las bifurcaciones dentro de $subtree(1B)$. De esta manera la cadena principal será la formada por los bloques 0, 1B, 2C, 3D, 4B (y no la cadena más larga que termina en el bloque 5B).

4.2. Propiedades básicas de GHOST

Es importante mostrar primero que todos los nodos eventualmente adoptan la misma historia cuando siguen $GHOST$. Para cualquier bloque B, sea ψ_B el momento más temprano en el cual el bloque B es ya sea abandonado por todos los nodos o adoptado por todos ellos.

Teorema 9. $\mathbb{P}(\psi_B < \infty)$. *En otras palabras cualquier bloque es eventualmente abandonado por todos los nodos o adoptado por todos ellos. Más aún* $\mathbb{E}(\psi_B) < \infty$.

Demostración. Sea D el diámetro de retraso de la red, suponga que al tiempo $t > time(B)$ el bloque B no ha sido abandonado por todos los nodos ni tampoco adoptado por todos ellos. Denotemos por ε_t el evento en el que la creación del siguiente bloque ocurre entre los tiempos $t + D$ y $t + 2D$, y no hay otro bloque que se produzca sino hasta el tiempo $t_0 \geq t+3D$. Afirmamos que una vez que tal evento ocurra el bloque B es abandonado o adoptado por todos los nodos. En efecto, entre los tiempos t y $t+D$ todos los se enteran de la existencia de todos los bloques, y por lo tanto cada par de hojas que tiene nodos activos tratando de extenderlas debe estar en un subárbol del mismo peso enraizado en un ancestro común. Luego un bloque es creado entre los tiempos $t + D$ y $t + 2D$ con lo cual se rompe el empate, y otras D unidades de tiempo permiten que este nuevo bloque se propague por toda la red, lo cual causa que todos los nodos adopten una misma historia.

Por otro lado, note que $\mathbb{P}(\varepsilon_t)$ es positiva, y además es la misma para cualquier t. Por lo tanto el tiempo de espera esperado para el primer evento ε_t es finito. Finalmente, ψ_B está acotado, por definición, por el tiempo de espera para el primer evento ε_t, en consecuencia $\mathbb{E}(\psi_B) < \infty$. ■

A continuación mostraremos la principal ventaja de la regla de selección de la cadena principal mediante GHOST, a saber, que es resistente a ataques del 50 %, incluso a tasas altas de creación de bloques con retrasos significativos en la red: al esperar un periodo de tiempo suficientemente largo τ después de la creación del bloque, la probabilidad de que su estado cambie de "aceptado" a "abandonado" puede hacerse arbitrariamente pequeña.

Teorema 10. *Supongamos que la tasa de creación de bloques del atacante es $q\lambda_h$, con $0 \leq q < 1$. La probabilidad de que un bloque B quede fuera de la cadena principal en algún momento después del tiempo $time(B) + \tau$, dado que este estaba en la cadena principal al tiempo $time(B) + \tau$, tiende a cero cuando τ tiende a infinito.*

Demostración. Sabemos que $\mathbb{E}(\psi_B)$ es finita (teorema 9), luego aplicando la desigualdad de Markov, se sigue que la probabilidad de que antes del tiempo $time(B) + \tau$ el bloque B ya estuviese abandonado o adoptado por todos los nodos honestos, tiende a 1 cuando τ tiende a infinito. En el primer caso, si B fue abandonado por todos los nodos honestos antes del tiempo $time(B) + \tau$, entonces la probabilidad de que sea abandonada después del tiempo $time(B) + \tau$ es cero. Y en el último caso, en el cual B fue adoptado por todos los nodos honestos, ahora ellos construyen bloques en $subtree(B)$ con una taza de λ_h, la cual es más grande que $q\lambda_h$. En consecuencia, conforme τ crece, la brecha entre el tamaño de $subtree(B)$ y la cadena del atacante crece, haciendo la probabilidad de un ataque exitoso en un momento en el futuro arbitrariamente bajo. ■

4.3. Crecimiento de la cadena principal

En esta sección vamos a comparar sistemáticamente las dos reglas de selección de la cadena principal realizando un análisis de la tasa de crecimiento de la cadena principal, bajo cada una de estas dos reglas. Para esto utilizamos el siguiente enfoque: supongamos que tenemos un grupo de nodos relativamente bien conectados (con diámetro de retraso D) que tiene una fracción $0 \leq \alpha \leq 1$ del poder computacional de la red entera. En este caso, los bloques creados dentro de esta subred se propagan internamente de manera relativamente rápida, y podemos acotar la tasa de crecimiento de la cadena principal por debajo.

Lema 11. *Sea $G = (V, E)$ una sugráfica de la red entera la cual genera bloques a una tasa de $\lambda' = a \cdot \lambda$, con diámetro de retraso D. Entonces bajo la regla de la cadena más larga se tiene que $\beta \geq \frac{\lambda'}{1+\lambda' \cdot D}$*

Demostración. Consideremos una secuencia de bloques $U_0, U_1, U_2, \ldots$ tal que el bloque U_{i+1} es el primero en ser creado después de que han pasado D segundos desde la creación del bloque U_i, es decir, U_{i+1} es el primer bloque B tal que $time(B) - time(U_i) > D$.

Afirmación 1. Sea $U_0, U_1, U_2, \ldots$ una serie de bloques tal que $time(U_{i+1}) - time(U_i) > D$ para cada $i \in \mathbb{N}$. Entonces para cualquier entero $n \geq 0$: $Depth(U_n) - Depth(U_0) \geq n$.

La afirmación puede ser probada por inducción. Es claro que cuando $n = 0$ la afirmación es verdadera. Ahora supongamos que la afirmación es cierta para $n = k$ y demostremos su validez para $n = k + 1$. Al tiempo $time(U_k)$ tenemos $Depth(U_k) - Depth(U_0) \geq k$. Ahora consideremos el tiempo en el cual el bloque U_{k+1} es creado. El nodo que creó este bloque lo ha hecho luego de escuchar acerca del bloque U_k y como $Depth(U_k) - Depth(U_0) \geq k$, implica que tal nodo tiene una cadena de longitud al menos k, y por lo tanto U_{k+1} es construida con una profundidad de al menos 1 más que U_k.

Ahora que ya hemos demostrado la afirmación podemos regresar a calcular la cota inferior para β. Para esto sea $X_i = time(U_i) - time(U_{i-1})$ la variable aleatoria que representa el tiempo entre la creación de los bloques U_{i-1} y U_i. Note que las X_i's son variables aleatorias i.i.d porque el tiempo que estas denotan es D unidades de tiempo mas un tiempo de espera distribuido exponencialmente para la creación del siguiente bloque. Luego por la afirmación anterior, la cadena más larga ha crecido al menos n durante el tiempo $\sum_{i=1}^{n} X_i$, esto implica que $\sum_{i=1}^{n} \tau_i \leq \sum_{i=1}^{n} X_i$, y entonces

$$\beta \geq \frac{1}{\mathbb{E}\left[\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} X_i\right]},$$

pero por la Ley Fuerte de los Grandes Números $\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} X_i = \mathbb{E}[X_1]$ c.s. En consecuencia $\mathbb{E}[\lim_{n\to\infty} \frac{1}{n} \sum_{i=1}^{n} X_i] = \mathbb{E}[X_1]$. Por otro lado observemos que $\mathbb{E}[X_1] = D + \mathbb{E}[Y]$, donde Y es una variable aleatoria con distribución

exponecial de parámetro λ'. Luego como $\mathbb{E}[Y] = \frac{1}{\lambda'}$ obtenemos $\beta \geq \frac{1}{D+\frac{1}{\lambda'}} = \frac{\lambda'}{1+\lambda' \cdot D}$.

■

Lema 12. *Sea G una sugráfica de la red entera, la cual genera bloques a una tasa de $\lambda' = a \cdot \lambda$, con diámetro de retraso D. Entonces bajo la regla de GHOST se tiene que $\beta \geq \frac{\lambda'}{1+2\lambda' \cdot D}$.*

Para la demostración usaremos la siguiente afirmación.

Afirmación 2. *Sea B un bloque en el árbol T en una red como en el lema 12, entonces a pesar de su historia, el tiempo de espera esperado para la creación del último hijo de B está acotado superiormente por $2D + \frac{1}{\lambda'}$.*

Demostración (de la Afirmación 2). Sea C el primer bloque creado después de que han pasado D unidades de tiempo desde la creación de B. Tomemos $\tau = time(C) - time(B) + D$, note que τ denota un intervalo de tiempo $2D$ unidades de tiempo más un tiempo de espera distribuido exponencialmente para la creación del bloque C, por lo cual $\mathbb{E}[\tau] = 2D + \frac{1}{\lambda'}$.

Afirmamos que después de τ segundos desde la creación de B, B ya no tendrá más hijos. Examinemos los dos posibles casos:

Caso 1: C es un descendiente de B. Una vez que C ha sido propagado a todos los nodos, ningún nodo considera a B una hoja, y por la regla de selección GHOST, únicamente se extienden hojas en el subárbol conocido por cada nodo.

Caso 2. C no es un descendiente de B. Como el bloque B fue propagado a todos los nodos antes de la creación de C, el nodo que creó a C era consciente de B, pero dicho nodo no creó a C sobre el subárbol enraizado en B, y por lo tanto, si consideramos F el primer ancestro común de B y C, entonces tal nodo tenía un subárbol enraizado en F del que C es parte estrictamente más pesado que el subárbol enraizado en F del que B es parte, después de la creación de C. D segundos después el bloque C es conocido por todos los nodos junto con todo su subárbol soporte, en consecuencia B ya no será extendido directamente, ya sea porque ningún nodo está tratando de extenderlo, o porque ahora estos están tratando de extender sobre los descendientes de B si es que este tiene hijos. ■

Demostración (del Lema 12). Sean $B_0, B_1, \ldots$ los bloques de la cadena más larga ordenados de acuerdo a su tiempo de creación. Para cada número entero $i \geq 1$ tomemos $\tau_i = time(B_i) - time(B_{i-1})$. Ahora por cada bloque B_i consideremos C_i el primer bloque creado después de que han pasado D unidades de tiempo desde la creación de B_i, y sea $\tau_i' = time(C_{i-1}) - time(B_{i-1}) + D$. En la afirmación 2 vimos que después de que han pasado τ_i' unidades de tiempo desde la creación de B_{i-1}, B_{i-1} ya no tendrá más hijos, eso significa que $\tau_i \leq \tau_i'$. En consecuencia

$$\beta = \frac{1}{\mathbb{E}\left[\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \tau_i\right]} \geq \frac{1}{\mathbb{E}\left[\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \tau_i'\right]}.$$

Por otra lado, $\tau_1', \tau_2', \ldots$ son variables aleatorias independientes e identicamente distribuidas, porque cada τ_i' denota un intervalo de $2D$ unidades de tiempo más un tiempo de espera distribuido exponencialmente para la creación del bloque C_{i-1}. Entonces por la ley fuerte de los grandes números $\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \tau_i' = \mathbb{E}[\tau_1']$ c.s. De aquí se sigue que $\mathbb{E}\left[\lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \tau_i'\right] = \mathbb{E}[\tau_1']$, pero $\mathbb{E}[\tau_1'] = 2D + \frac{1}{\lambda'}$ y por lo tanto $\beta \geq \frac{\lambda'}{1+2\lambda'\cdot D}$.

■

Ahora damos una cota superior para la tasa de crecimiento de la cadena principal en el caso no trivial más simple posible: una red con solo dos nodos.

Teorema 13. *Tomemos una red que consiste únicamente de dos nodos, u y v, y con diámetro de retraso D. Tomemos p_S y p_T ($p_S \neq p_T$) la fracción del poder computacional de u y v, respectivamente. Entonces bajo la regla de la cadena más larga y bajo GHOST se tiene que*

$$\beta(\lambda) = \frac{(p_s\lambda)^2 e^{p_S\lambda 2d} - (p_T\lambda)^2 e^{p_T\lambda 2d}}{p_s\lambda e^{p_S\lambda 2d} - p_T\lambda e^{p_T\lambda 2d}}.$$

Demostración. u y v crean bloques separadamente, y siempre que uno completa un bloque, este envía un mensaje con su nuevo bloque a traves del enlace para llegar a su contraparte d segundos después. En esos d segundos el nodo aún continua con el intento de construir nuevos bloques y alargar su propia versión de la cadena principal. Así que mensajes acerca de bloques a

la misma profundidad (los cuales fueron creados por u y v aproximadamente al mismo tiempo) pueden estar simultáneamente viajando en direcciones opuestas sobre el enlace, causando una bifurcación en el árbol de bloques.

Note que el árbol de bloques es en realidad un árbol binario, ya que en cualquier punto del tiempo hay a lo más dos ramas no abandonadas. Note que en este caso la cadena más larga y la más pesada coinciden. Entonces el análisis de abajo aplica igualmente para una red siguiendo la regla de la cadena más larga o la que sigue $GHOST$.

Con el objetivo de contar el número de bloques que fallan en entrar a la cadena principal, notemos que tal evento ocurre precisamente cuando dos bloques a la misma altura han sido creados.

Consideremos un bloque U de u. Decimos que la venta de U es creada d unidades de tiempo antes de la creación de U, y se va d unidades de tiempo después de la creación de U. Decimos que U está *amenazado* a un tiempo dado si la ventana de U ha sido creada y la cadena de v es de longitud $depth(U)-1$ (el intervalo de tiempo en el cual U esta amenazada está contenido en la ventana de U). Durante este periodo, el siguiente bloque creado por v estará a la misma profundidad que U y uno de los bloques será desperdiciado.

Definimos $abierto(U)$ como el tiempo que transcurrió desde la creación de la ventana de U hasta el momento en que se ve amenazado, y definimos $cerrar(U)$ como el tiempo que transcurrió desde la creación de la ventana de U hasta el momento que deja de estar amenazado.

Observemos que el cierre de U puede ocurrir de dos maneras: ya sea que $2d$ unidades de tiempo han pasado desde la creación de la ventana de U y v recibe un mensaje que contiene a U (ahora v tiene una cadena de longitud al menos $depth(U)$, o v generó un bloque a la misma profundidad que U antes de que pasaran las $2d$ unidades de tiempo desde la creación de la ventana de U. Por lo tanto $close(U)-open(U)$ es un valor entre 0 y $2d$. Además, notemos que dos bloques de u no pueden estar simultáneamente amenazados.

Asumiendo que la venta del bloque U fue creada al tiempo 0, $open(U_n)$ y $close(U_n)$ son variables aleatorias que toman valores en $[0, 2d]$, además $close(U_n) \geq open(U_n)$. La distribución de $open(U_n)$ está compuesta de una parte continua en la región $(0, 2d]$, y una parte discreta en el evento $\{open(U_n) = 0\}$. Denotamos al primero por $\alpha_n(x)$, para $x \in (0, 2d]$, y al segundo por $\alpha_{n,0}$. Similarmente, la distribución de probabilidad de $close(U_n)$ tiene una parte continua sobre el intervalo $[0, 2d)$ la cual denotamos por $\omega_n(x)$ y una parte

discreta $\omega_{n,2d}$ para el evento $\{close(U_n) = 2d\}$.

Denotemos por f_S y f_T las funciones de densidad de las variables aleatorias exponenciales con parámetros $p_S\lambda$ y $p_T\lambda$, respectivamente. Afirmamos que

$$\alpha_n(x) = \int_x^{2d} \omega_{n-1}(y) \cdot f_S(y-x)dy + \omega_{n-1,2d} \cdot f_S(2d-x), \quad 0 < x \leq 2d \quad (4.3.1)$$

$$\omega_n(x) = \int_0^x \alpha_n(z) \cdot f_T(x-z)dz + \alpha_{n,0} \cdot f_T(x), \quad 0 \leq x < 2d \quad (4.3.2)$$

En efecto, para probar 4.3.1 basta ver que si $x > 0$ entonces U_n abre x segundos después de la creación de su ventana si y sólo si para algún y, U_{n-1} cerró y segundos después de la creación de su ventana (con probabilidad $\omega_{n-1}(y)$ para $y < d$ y $\omega_{n-1,2d}$ para $y = 2d$), y la diferencia entre los tiempos de creación de U_n y U_{n-1} fue $y - x$ segundos ($f_S(y-x)$), ver figura 4.2.

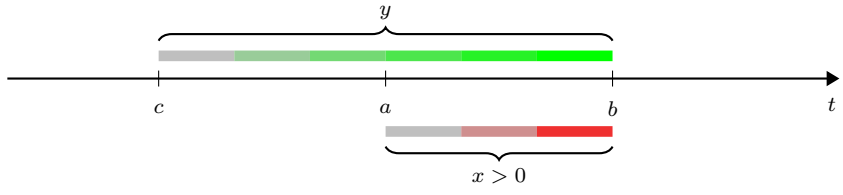

Figura 4.2. Los tiempos c y a son los tiempos en los que se crearon las ventanas de U_{n-1} y U_n, respectivamente. b es el primer momento en el que v tiene una cadena de longitud $depth(U_n) - 1$.

Supongamos que U_n abre $x > 0$ segundos después de la creación de su ventana, y que la creación de dicha ventana se dió en el tiempo a, entonces al tiempo $a+x = b$ es el primer momento en que v tiene una cadena de longitud $depth(U_n) - 1$ $(x > 0)$, pero $depth(U_n) - 1 \geq depth(U_{n-1})$, esto implica que al tiempo $a + x$, v ya tenía una cadena de longitud al menos $depth(U_{n-1})$. En consecuencia, si al tiempo c se creó la ventana de U_{n-1} y U_{n-1} cerró y segundos después de la creación de su ventana, entonces $c + y \leq b$.

Veamos que $c + y = b$, como $depth(U_n) - 1 \geq depth(U_{n-1})$ tenemos dos posibles casos. Si $depth(U_n) - 1 = depth(U_{n-1})$, entonces como b y $c+y$ fueron los primeros momentos en que v tuvo una cadena de longitud $depth(U_n) - 1$ y $depth(U_{n-1})$, respectivamente, obtenemos que $b = c + y$.

Ahora supongamos que $depth(U_n) - 1 > depth(U_{n-1})$. En este caso notemos que U_n es construido al tope de un bloque U_z, donde U_z es un bloque creado por v y cumple que $depth(U_n - 1) = depth(U_z) > depth(U_{n-1})$. Pero d segundos transcurrieron al menos entre la creación de U_z y la creación de U_n,(ya que sobre ese bloque se construyó U_n), entonces desde la creación de la ventana de U_n hasta la creación del bloque U_n pasaron al menos $x + d > d$ segundos, lo cual es una contradicción por como se define el inicio de la ventana de U_n. Esto quiere decir que el único caso posible es cuando $depth(U_n) - 1$ y $depth(U_{n-1})$ son iguales, y en tal caso $c + y = b$. Y ya finalmente de aquí obtenemos que la diferencia entre los tiempos de creación de los bloques U_n y U_{n-1} fue $y - x$ segundos (ver figura 4.2).

Inversamente, supongamos que U_n cerró y segundos después de la creación de su ventana y la diferencia entre el tiempo de creación de U_n y U_{n-1} fue $y - x$ $(x > 0)$. Sean c y a los tiempos en los que se crearon la ventanas de U_{n-1} y U_n, respectivamente, y $b = c + y$. Como la diferencia entre sus tiempos de creación fue $y - x$, implica que al tiempo $c + (y - x)$ se creó la ventana de U_n, y en consecuencia $c + (y - x) = a$. Supongamos que U_n abrió x' segundos después de la creación de su ventana. Veamos que $x' = x$.

Como b es el primer momento en el que v tuvo una cadena de longitud $depth(U_{n-1})$ se sigue que si $depth(U_{n-1}) = depth(U_n) - 1$, entonces b es el primer momento en el que v tuvo una cadena de longitud $depth(U_n) - 1$ y entonces $x' = x$. Por otro lado, si $depth(U_{n-1}) < depth(U_n) - 1$, entonces nuevamente como antes llegamos a que desde la creación de la ventana de U_n hasta la creación del bloque U_n pasaron al menos $x + d > d$ segundos, lo cual es una contradicción, por lo que este caso no es posible.

Por lo tanto, si $x > 0$ entonces U_n abre x segundos después de la creación de su ventana si y sólo si para algún y, U_{n-1} cerró y segundos después de la creación de su ventana (con probabilidad $\omega_{n-1}(y)$ para $y < d$ y $\omega_{n-1,2d}$ para $y = 2d$), y la diferencia entre los tiempos de creación de U_n y U_{n-1} fue $y - x$ segundos $(f_S(y - x))$. De esta manera queda demostrado 4.3.1.

Para 4.3.2, note que U_n cierra x segundos después de la creación de su ventana si y sólo si para algún z, z segundos pasaron entre la creación de la

venta de U_n y su apertura (con probabilidad $\alpha_n(z)$ para $z > 0$ y $\alpha_{n,0}$ para $z = 0$), y $x - z$ segundos entre la apertura y la clausura de U_n ($f_T(x - z)$).

Los procesos *open*(U_n) y *close*(U_n) son Markovianos, y ahora escribimos las ecuaciones 4.3.1 y 4.3.2 aplicadas a sus distribuciones límites, $\alpha(x)$, α_0, $\omega(x)$, ω_{2d}:

$$\alpha(x) = \int_x^{2d} \omega(y) \cdot f_S(y - x)dy + \omega_{2d} \cdot f_S(2d - x), \quad 0 < x \leq 2d \tag{4.3.3}$$

$$\omega(x) = \int_0^{x} \alpha(z) \cdot f_T(x - z)dz + \alpha_0 \cdot f_T(x), \quad 0 \leq x < 2d \tag{4.3.4}$$

Ahora consideremos

$$\gamma(x) = \int_x^{2d} \omega(y) \cdot f_S(y - x)dy + \omega_{2d} \cdot f_S(2d - x), \quad 0 \leq x \leq 2d \tag{4.3.5}$$

Note que

$$\begin{bmatrix}\gamma \\ \omega\end{bmatrix}' = \begin{bmatrix}p_S\lambda & -p_S\lambda \\ p_T\lambda & -p_T\lambda\end{bmatrix}\begin{bmatrix}\gamma \\ \omega\end{bmatrix}, \tag{4.3.6}$$

entonces con el objetivo de determinar γ y ω, resolveremos el siguiente sistema de ecuaciones diferenciales:

$$\begin{bmatrix}x \\ y\end{bmatrix}' = \begin{bmatrix}p_S\lambda & -p_S\lambda \\ p_T\lambda & -p_T\lambda\end{bmatrix}\begin{bmatrix}x \\ y\end{bmatrix}, \quad \begin{bmatrix}x(0) \\ y(0)\end{bmatrix} = \begin{bmatrix}\gamma(0) \\ \omega(0)\end{bmatrix}. \tag{4.3.7}$$

Para solucionar este sistema tomemos

$$A = \begin{bmatrix}p_S\lambda & -p_S\lambda \\ p_T\lambda & -p_T\lambda\end{bmatrix}.$$

Luego los valores propios de A son $\lambda_1 = 0$ y $\lambda_2 = p_S\lambda - p_T\lambda$, los cuales están asociados a los vectores propios $v_1 = (1, 1)$ y $v_2 = (p_S, p_T)$, respectivamente. En consecuencia la solución al sistema es

$$\begin{bmatrix}x(t) \\ y(t)\end{bmatrix} = c_1 e^{0t}\begin{bmatrix}1 \\ 1\end{bmatrix} + c_2 e^{(p_S\lambda - p_T\lambda)t}\begin{bmatrix}p_S\lambda \\ p_T\lambda\end{bmatrix} \tag{4.3.8}$$

donde c_1 y c_2 son valores que cumplen con el siguiente sistema de ecuaciones:

$$c_1 + c_2 p_S \lambda = \gamma(0)$$
$$c_1 + c_2 p_T \lambda = \omega(0).$$

Finalmente despejando a c_1 y c_2 se sigue que

$$\begin{bmatrix} x(t) \\ y(t) \end{bmatrix} = \frac{A}{S} \begin{bmatrix} p_S\lambda(e^{Sx} - 1) \\ p_T\lambda(e^{Sx} - 1 \end{bmatrix} + \begin{bmatrix} \gamma(0) \\ \omega(0) \end{bmatrix}, \tag{4.3.9}$$

con $A = \gamma(0) - \omega(0)$; $S = p_S\lambda - p_T\lambda$. Pero además la solución al sistema de ecuaciones diferenciales es única una vez establecida la condición inicial, entonces $x(t) = \gamma(t) = a(t)$, $y(t) = \omega(t)$, y por lo tanto

$$\begin{bmatrix} \gamma(t) \\ \omega(t) \end{bmatrix} = \frac{A}{S} \begin{bmatrix} p_S\lambda(e^{Sx} - 1) \\ p_T\lambda(e^{Sx} - 1 \end{bmatrix} + \begin{bmatrix} \gamma(0) \\ \omega(0) \end{bmatrix}. \tag{4.3.10}$$

A continuación mostraremos que la igualdad anterior implica

$$\omega_{2d} = \frac{p_S\lambda - p_T\lambda}{p_S\lambda - p_T\lambda e^{-(p_S\lambda - p_T\lambda)2d}}. \tag{4.3.11}$$

Por las ecuaciones 4.3.4 y 4.3.5, $\gamma(2d) = \omega_{2d} p_S \lambda$ y $\omega(0) = a_0 p_T \lambda$. Por lo tanto
$\omega(x) = \widehat{A} p_T \lambda (e^{(p_S\lambda - p_T\lambda)x} - 1) + a_0 p_T \lambda$, con $\widehat{A} = \dfrac{A}{p_S\lambda - p_T\lambda} \Rightarrow$
$\omega'(x) = p_T \lambda A e^{(p_S\lambda - p_T\lambda)x}$,
luego por 4.3.6, $\omega'(x) = p_T\lambda(\gamma(x) - \omega(x))$, por lo tanto
$\gamma(x) - \omega(x) = Ae^{(p_S\lambda - p_T\lambda)x} \Rightarrow \gamma'(x) = p_S\lambda(\gamma(x) - \omega(x)) \Rightarrow$
$\gamma(x) = \int_0^x p_S\lambda(\gamma(t) - \omega(t))dt + \gamma(0) = p_S\lambda \int_0^x Ae^{(p_S\lambda - p_T\lambda)x} dt + \gamma(0) =$
$\dfrac{p_S\lambda A}{p_S\lambda - p_T\lambda}(e^{(p_S\lambda - p_T\lambda)x} - 1) + \gamma(0).$
Como $\gamma(x) - \omega(x) = Ae^{(p_S\lambda - p_T\lambda)x}$, obtenemos
$\gamma(0) = \omega(0) + A \Rightarrow \gamma(x) = \dfrac{p_S\lambda A}{p_S\lambda - p_T\lambda}(e^{(p_S\lambda - p_T\lambda)x} - 1) + a_0 p_T\lambda + A \Rightarrow$
$\gamma(2d) = \dfrac{p_S\lambda A}{p_S\lambda - p_T\lambda}(E - 1) + a_0 p_T\lambda + A = \dfrac{p_S\lambda AE - p_T\lambda A}{p_S\lambda - p_T\lambda} + a_0 p_T\lambda$
donde $E = e^{(p_S\lambda - p_T\lambda)2d}$. Por lo tanto

$$\widehat{A} = \frac{\gamma(2d) - a_0 p_T \lambda}{p_S \lambda E - p_T \lambda} = \frac{\omega_{2d} p_S \lambda - a_0 p_T \lambda}{p_S \lambda E - p_T \lambda}.$$

Anteriormente obtuvimos expresiones explicitas para γ y ω :

$$\gamma(x) = \widehat{A}(p_S \lambda e^{(p_S \lambda - p_T \lambda)x} - p_T \lambda) + a_0 p_T \lambda, \tag{4.3.12}$$

$$\omega(x) = \widehat{A}(p_T \lambda e^{(p_S \lambda - p_T \lambda)x} - p_T \lambda) + a_0 p_T \lambda. \tag{4.3.13}$$

Sabemos que $\alpha(x) = \gamma(x)$ para $0 < x \leq 2d$, además por definición de α, la integral de α sobre el intervalo $(0, 2d]$ debe ser $1 - a_0$, entonces tenemos

$$\begin{aligned} 1 - a_0 &= \int_0^{2d} \left(\widehat{A}(p_S \lambda e^{(p_S \lambda - p_T \lambda)t} - p_T \lambda) + a_0 p_T \lambda \right) dt \\ &= \widehat{A} \left(\frac{p_S \lambda (E-1)}{p_S \lambda - p_T \lambda} - 2d \cdot p_T \lambda \right) + 2d \cdot a_0 p_T \lambda \\ &= \widehat{A} \left(\widehat{E} - 2d \cdot p_T \lambda \right) + 2d \cdot a_0 p_T \lambda \end{aligned}$$

donde $\widehat{E} = \dfrac{p_S \lambda (E-1)}{p_S \lambda - p_T \lambda}$. Luego despejando a_0 de la igualdad anterior y sustituyendo el valor de $\widehat{A}$ se obtiene

$$a_0 = 1 - \frac{\omega_{2d} p_S \lambda - a_0 p_T \lambda}{p_S \lambda E - p_T \lambda} \left(\widehat{E} - 2d \cdot p_T \lambda \right) - 2d \cdot a_0 p_T \lambda. \tag{4.3.14}$$

Similarmente, la integral de ω sobre el intervalo $[0, 2d)$ debe ser $1 - \omega_{2d}$, y combinando esto con la igualdad $\gamma(x) - \omega(x) = Ae^{(p_S \lambda - p_T \lambda)x}$ obtenemos

$$\begin{aligned} &1 - a_0 - (1 - \omega_{2d}) = \int_0^{2d} A e^{(p_S \lambda - p_T \lambda)t} dt = \widehat{A}(E-1) \quad \Rightarrow \\ &\omega_{2d} - a_0 = \widehat{A}(E-1) = \frac{\omega_{2d} p_S \lambda - a_0 p_T \lambda}{p_S \lambda E - p_T \lambda}(E-1) \quad \Rightarrow \\ &\frac{\omega_{2d}}{a_0} - 1 = \frac{\frac{\omega_{2d}}{a_0} p_S \lambda - p_T \lambda}{p_S \lambda E - p_T \lambda} \quad \Rightarrow \end{aligned}$$

$$1-\left(\frac{p_S\lambda(E-1)}{p_S\lambda E-p_T\lambda}\right)\frac{\omega_{2d}}{a_0}=1-\frac{p_T\lambda(E-1)}{p_S\lambda E-p_T\lambda}\quad\Rightarrow$$
$$\frac{\omega_{2d}}{a_0}=\frac{1-\frac{p_T\lambda}{p_S\lambda E-p_T\lambda}(E-1)}{1-\frac{p_S\lambda}{p_S\lambda E-p_T\lambda}(E-1)}=E.$$

En consecuencia

$$\omega_{2d}=E\cdot\alpha_0=E\left(1-\frac{\omega_{2d}p_S\lambda-\frac{\omega_{2d}}{E}p_T\lambda}{p_S\lambda E-p_T\lambda}\left(\widehat{E}-2d\cdot p_T\lambda\right)-2d\cdot\frac{\omega_{2d}}{E}p_T\lambda\right)=$$
$$E\left(1-\frac{\omega_{2d}}{E}\left(\widehat{E}-2d\cdot p_T\lambda\right)-2d\cdot\frac{\omega_{2d}}{E}p_T\lambda\right)=E-\omega_{2d}\widehat{E}\quad\Rightarrow$$
$$\omega_{2d}=\frac{E}{1+\widehat{E}}=\frac{e^{(p_S\lambda-p_T\lambda)2d}}{1+\frac{p_S\lambda(e^{(p_S\lambda-p_T\lambda)2d}-1)}{p_S\lambda-p_T\lambda}}=\frac{e^{(p_S\lambda-p_T\lambda)2d}(p_S\lambda-p_T\lambda)}{p_S\lambda e^{(p_S\lambda-p_T\lambda)2d}-p_T\lambda}$$
$$=\frac{p_S\lambda-p_T\lambda}{p_S\lambda-p_T\lambda e^{-(p_S\lambda-p_T\lambda)2d}}.$$

Finalmente por definición de ω_{2d}, este es precisamente la fracción de los bloques de u que no tienen conflicto con los bloques creados por v. Entonces los bloques los cuales contribuyen al crecimiento de la cadena principal se pueden contar considerando todos los bloques de v como válidos, y agregando todos esos bloques no conflictivos de u. Entonces así obtenemos

$$\begin{aligned}\beta(\lambda)&=p_T\lambda+\omega_{2d}\cdot p_S\lambda\\&=p_T\lambda+\frac{p_S\lambda-p_T\lambda}{p_S\lambda-p_T\lambda e^{-(p_S\lambda-p_T\lambda)2d}}p_S\lambda\\&=\frac{(p_s\lambda)^2e^{p_S\lambda 2d}-(p_T\lambda)^2e^{p_T\lambda 2d}}{p_s\lambda e^{p_S\lambda 2d}-p_T\lambda e^{p_T\lambda 2d}}.\end{aligned}$$

■

4.4. Seguridad en redes con retrasos

Enfatizamos que la suposición de Satoshi era que los tiempos de propagación de bloques son insignificantes en comparación con el tiempo que lleva crearlos. En su análisis, se muestra que si la tasa de generación de bloques

de la red honesta es p y la del atacantes es q, entonces la probabilidad de que el atacante pueda generar más bloques que los agregados a la cadena más larga de la red, dado que el atacante comienza con una desventaja de X_0 bloques, es exactamente $\left(\frac{q}{p}\right)^{X_0+1}$ (cuando $p \geq q$). Sin embargo, su análisis no se aplica a las redes con retrasos no despreciables, por lo que aquí en esta sección volvemos a revisar el tema del doble gasto.

Teorema 14. *Considere un red G con retrasos. Sea $1/\beta_1$ una cota superior para el tiempo de espera esperado para el próximo alargamiento de la cadena principal, para todos los posibles estados del sistema. Sea $\gamma < \beta_1$ la tasa de creación de bloques del atacante (de acuerdo a un proceso Poisson), y suponga que la brecha entre la cadena más larga de la red y la del atacante es de X_0 bloques. Entonces la probabilidad de que el atacante logre extender su cadena y que esta llegue a ser más larga que la de la red es a lo más $\left(\frac{\gamma}{\beta_1}\right)^{X_0+1}$.*

La demostración depende de los siguientes dos lemas:

Lema 15. *Sea ς una variable aleatoria no negativa con función de riesgo creciente. Entonces, para cualquier $n \in \mathbb{N}$*

$$\mathbb{E}[\varsigma^n] \leq n!\mathbb{E}^n[\varsigma].$$

Lema 16. *Sea ς una variable aleatoria como en el Lema 15, sea f su función de densidad, y denotemos por $\beta := \mathbb{E}[\varsigma]^{-1}$. Entonces para cualquier constante $0 < \gamma < \beta$ se tiene que $I(\frac{\gamma}{\beta}) \leq 0$, donde*

$$I(a) := \int_0^\infty f(\varsigma) e^{\gamma(\frac{1}{a}-1)\varsigma} d\varsigma - \frac{1}{a}.$$

Más aún, si β_1 es una constante tal que $0 < \gamma < \beta_1$ y $\beta \geq \beta_1$, entonces $I(\frac{\gamma}{\beta_1}) \leq 0$.

Demostración (del Lema 15). Por inducción sobre n. El caso base $n = 0$ es trivial. Ahora supongamos que la desigualdad es válida para $n = k$. Vamos a demostrar que también es cierta para $n = k + 1$. Para esto note que

$$\begin{aligned}\mathbb{E}[\varsigma^{k+1}] &= \int_0^\infty \varsigma^{k+1} f(\varsigma) d\varsigma = \int_0^\infty \varsigma^{k+1} \lambda(\varsigma) e^{-\Lambda(\varsigma)} d\varsigma \\ &= \left[\varsigma^{k+1}(F(\varsigma)-1)\right]_0^\infty - \int_0^\infty (k+1)\varsigma^k (F(\varsigma)-1) d\varsigma \\ &= \left[\varsigma^{k+1}(-e^{-\Lambda(\varsigma)})\right]_0^\infty + \int_0^\infty (k+1)\varsigma^k e^{-\Lambda(\varsigma)} d\varsigma \\ &= (k+1)\int_0^\infty \frac{\varsigma^k}{\lambda(\varsigma)} \lambda(\varsigma) e^{-\Lambda(\varsigma)} d\varsigma.\end{aligned}$$

Por otro lado

$$\begin{aligned}\mathbb{E}[\varsigma] &= \int_0^\infty \varsigma f(\varsigma) d\varsigma = \int_0^\infty \varsigma \lambda(\varsigma) e^{-\Lambda(\varsigma)} d\varsigma \\ &= \left[\varsigma(F(\varsigma)-1)\right]_0^\infty - \int_0^\infty (F(\varsigma)-1) d\varsigma \\ &= \left[\varsigma(-e^{-\Lambda(\varsigma)})\right]_0^\infty + \int_0^\infty e^{-\Lambda(\varsigma)} d\varsigma \\ &= \int_0^\infty e^{-\Lambda(\varsigma)} d\varsigma\end{aligned}$$

y por lo tanto

$$\begin{aligned}(k+1)!\mathbb{E}^{k+1}[\varsigma] &= (k+1)k!\mathbb{E}^k[\varsigma]\mathbb{E}[\varsigma] \\ &= (k+1)k!\mathbb{E}^k[\varsigma]\int_0^\infty e^{-\Lambda(\varsigma)} d\varsigma \\ &= (k+1)\int_0^\infty k!\frac{\mathbb{E}^k[\varsigma]}{\lambda(\varsigma)} \lambda(\varsigma) e^{-\Lambda(\varsigma)} d\varsigma.\end{aligned}$$

En consecuencia para probar que $(k+1)!\mathbb{E}^{k+1}[\varsigma] \geq \mathbb{E}[\varsigma^{k+1}]$, es suficiente demostrar que

$$(k+1)\int_0^\infty k!\frac{\mathbb{E}^k[\varsigma]}{\lambda(\varsigma)} \lambda(\varsigma) e^{-\Lambda(\varsigma)} d\varsigma \geq (k+1)\int_0^\infty \frac{\varsigma^k}{\lambda(\varsigma)} \lambda(\varsigma) e^{-\Lambda(\varsigma)} d\varsigma,$$

o equivalentemente, que

$$0 \geq \int_0^\infty \frac{\varsigma^k - k!\mathbb{E}^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma.$$

Por hipótesis de inducción $k!\mathbb{E}^k[\varsigma] \geq \mathbb{E}[\varsigma^k]$, entonces

$$k!\mathbb{E}^k[\varsigma] \geq \int_0^\infty \varsigma^k\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma,$$

pero

$$k!\mathbb{E}^k[\varsigma] = \int_0^\infty k!\mathbb{E}^k[\varsigma]\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma \tag{4.4.1}$$

en consecuencia

$$0 \geq \int_0^\infty (\varsigma^k - k!\mathbb{E}^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma. \tag{4.4.2}$$

Entonces

$$\begin{aligned}
&\int_0^\infty \frac{\varsigma^k - k!\mathbb{E}^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma = \\
&\int_0^{(k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}}} \frac{\varsigma^k - k!\mathbb{E}^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma + \int_{(k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}}}^\infty \frac{\varsigma^k - k!\mathbb{E}^k[\varsigma]}{\lambda(\varsigma)}\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma \\
&\leq \frac{1}{\lambda(k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}})}\int_0^{(k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}}} (\varsigma^k - k!\mathbb{E}^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma + \\
&\frac{1}{\lambda((k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}})}\int_{(k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}}}^\infty (\varsigma^k - k!\mathbb{E}^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma = \\
&\frac{1}{\lambda((k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}})}\int_0^\infty (\varsigma^k - k!\mathbb{E}^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}d\varsigma \leq 0,
\end{aligned}$$

donde en la primer desigualdad se usó que λ es creciente y que $h(\varsigma) < 0$ para $\varsigma \in (0, (k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}})$, $h(\varsigma) = 0$ para $\varsigma = (k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}}$ y $h(\varsigma) > 0$ para $\varsigma > (k!\mathbb{E}^k[\varsigma])^{\frac{1}{k}}$, con $h(\varsigma) = (\varsigma^k - k!\mathbb{E}^k[\varsigma])\lambda(\varsigma)e^{-\Lambda(\varsigma)}$. Y en la última desigualdad se usó 4.4.2. ■

Demostración (del Lema 16). Para la demostración de la primera parte del lema sea $H(\gamma) = I(\frac{\gamma}{\beta})$. Necesitamos mostrar que $H(\gamma) \leq 0$, y esto lo haremos mostrando que cada elemento en su serie de Taylor alrededor de β son todos menores o iguales que cero, esto es, mostraremos que $H^{(k)}(\beta)\frac{(\gamma-\beta)^k}{k!} \leq 0$, y

entonces esto implicará que $H(\gamma) = \sum_{k=0}^{\infty} H^{(k)}(\beta)\frac{(\gamma-\beta)^k}{k!} \leq 0$.

En efecto,

$$\begin{aligned} H^{(k)}(\gamma) &= \frac{d^k}{d\gamma^k}\left(\int_0^\infty f(t)e^{\gamma(\frac{1}{\frac{\gamma}{\beta}}-1)t}dt - \frac{1}{\frac{\gamma}{\beta}}\right) \\ &= \int_0^\infty \frac{d^k}{d\gamma^k}\left(f(t)e^{\gamma(\frac{1}{\frac{\gamma}{\beta}}-1)t}\right)dt + \frac{d^k}{d\gamma^k}\left(-\frac{1}{\frac{\gamma}{\beta}}\right) \\ &= \int_0^\infty \frac{d^k}{d\gamma^k}\left(f(t)e^{(\beta-\gamma)t}\right)dt + \frac{d^k}{d\gamma^k}\left(\frac{\beta}{-\gamma}\right) \\ &= \int_0^\infty (-t)^k f(t)e^{(\beta-\gamma)t}dt + \frac{k!\beta}{(-\gamma)^{k+1}}. \end{aligned}$$

Por lo tanto

$$H^{(k)}(\beta) = \int_0^\infty (-t)^k f(t)dt + k!\beta^{-k}(-1)^{k+1}.$$

Por el Lema 15 sabemos que $k!\mathbb{E}^k[\varsigma] \geq \mathbb{E}[\varsigma^k]$, entonces

$$0 \geq \int_0^\infty t^k f(t)dt - k!\beta^{-k},$$

luego note que cuando k es par $H^{(k)}(\beta) \leq 0$ y $(\gamma-\beta)^k > 0$, y cuando k es impar $H^{(k)}(\beta) \geq 0$ y $(\gamma-\beta)^k < 0$, así en cualquier caso se tiene que $H^{(k)}(\beta)\frac{(\gamma-\beta)^k}{k!} \leq 0$, ya de aquí se sigue que $H(\gamma) = \sum_{k=0}^{\infty} H^{(k)}(\beta)\frac{(\gamma-\beta)^k}{k!} \leq 0$.

Para la demostración de la segunda parte del lema, notemos primero que para toda $k \in \mathbb{N}$ se tiene $k!\beta^{-k} \leq k!\beta_1^{-k}$. Consideremos ahora $L(\gamma) = I(\frac{\gamma}{\beta_1})$, entonces nuevamente

$$L^{(k)}(\gamma) = \int_0^\infty (-t)^k f(t)e^{(\beta_1-\gamma)t}dt + \frac{k!\beta_1}{(-\gamma)^{k+1}}.$$

Por lo tanto

$$L^{(k)}(\beta_1) = \int_0^\infty (-t)^k f(t)dt + k!\beta_1^{-k}(-1)^{k+1}.$$

Por otro lado, sabemos que

$$0 \geq \int_0^\infty t^k f(t)dt - k!\beta^{-k},$$

así, cuando k es par

$$L^{(k)}(\beta_1) = \int_0^\infty t^k f(t)dt - k!\beta_1^{-k} \leq \int_0^\infty t^k f(t)dt - k!\beta^{-k} \leq 0,$$

y cuando k es impar

$$L^{(k)}(\beta_1) = -\int_0^\infty t^k f(t)dt + k!\beta_1^{-k} \geq -\int_0^\infty t^k f(t)dt + k!\beta^{-k} \geq 0,$$

en consecuencia $L(\gamma) = \sum_{k=0}^\infty L^{(k)}(\beta_1)\frac{(\gamma-\beta_1)^k}{k!} \leq 0$. ■

Demostración (del teorema 14). Sea τ_n el tiempo de espera para el n-ésimo crecimiento de la cadena principal. Denotemos por $\beta = \mathbb{E}[\tau_{n+1} \mid \tau_n, \ldots, \tau_1]^{-1}$, para alguna historia dada (esto es, para alguna realización de $\tau_n, \ldots, \tau_1$). Por hipótesis $\beta \geq \beta_1$, y entonces para toda $k \in \mathbb{N}$ se tiene $\beta^{-k} \leq \beta_1^{-k}$.

La variable aleatoria τ_{n+1} dada una historia es no negativa con función de riesgo creciente. En efecto, cuando un nodo crea un nuevo bloque este lo comparte a la red, y entre más y más nodos conozcan de su existencia, más poder computacional es contribuido al esfuerzo de crear el siguiente bloque, y así alargar la cadena principal. Si mientras tanto un nuevo bloque conflictivo fue creado en alguna otra parte, aún más poder computacional se está empleando para alargar la cadena principal, sólo que en una versión diferente.

La cadena del atacante es construida de acuerdo a un proceso Poisson con parámetro γ. Sea $N_2(t)$ la longitud de dicha cadena al tiempo t. Definamos $X_n = n - N_2(\sum_{j=1}^n \tau_j)$ y $Y_n = \left(\frac{\gamma}{\beta_1}\right)^{X_n}$.

El proceso $X = (X_n)$ representa la diferencia entre las longitudes de la cadena principal y la cadena del atacante, en favor del primero, ya que al tiempo $\sum_{j=1}^n \tau_j$ la cadena principal tiene longitud n.

Afirmamos que $Y = (Y_n)$ es una supermartingala. En efecto, mientra el valor de X_{n+1} depende naturalmente de $\tau_{n+1}, \ldots, \tau_1$, el incremento $X_{n+1} - X_n$

dada una historia $\tau_n, \ldots, \tau_1$ está controlada por la variable aleatoria τ_{n+1} dada esta historia, con función de densidad $f_{\tau_{n+1}|\tau_n,\ldots,\tau_1}$ la cual abreviamos como f. Entonces

$$\mathbb{E}\left[\left(\frac{\gamma}{\beta_1}\right)^{X_{n+1}} \middle| \left(\frac{\gamma}{\beta_1}\right)^{X_n} = \left(\frac{\gamma}{\beta_1}\right)^{k_n}, \ldots, \left(\frac{\gamma}{\beta_1}\right)^{X_0} = \left(\frac{\gamma}{\beta_1}\right)^{k_0}\right] =$$

$$\mathbb{E}\left[\left(\frac{\gamma}{\beta_1}\right)^{X_{n+1}} \middle| \left(\frac{\gamma}{\beta_1}\right)^{X_n} = \left(\frac{\gamma}{\beta_1}\right)^{k_n}\right] =$$

$$\mathbb{E}\left[\left(\frac{\gamma}{\beta_1}\right)^{n+1-N_2(\sum_{j=1}^{n+1}\tau_j)} \middle| \left(\frac{\gamma}{\beta_1}\right)^{n-N_2(\sum_{j=1}^{n}\tau_j)} = \left(\frac{\gamma}{\beta_1}\right)^{k_n}\right] =$$

$$\mathbb{E}\left[\left(\frac{\gamma}{\beta_1}\right)^{n+1-N_2(\sum_{j=1}^{n}\tau_j)-N_2(\tau_{n+1})} \middle| \left(\frac{\gamma}{\beta_1}\right)^{n-N_2(\sum_{j=1}^{n}\tau_j)} = \left(\frac{\gamma}{\beta_1}\right)^{k_n}\right] =$$

$$\left(\frac{\gamma}{\beta_1}\right)^{k_n+1} \mathbb{E}\left[\left(\frac{\gamma}{\beta_1}\right)^{-N_2(\tau_{n+1})}\right] =$$

$$\left(\frac{\gamma}{\beta_1}\right)^{k_n+1} \sum_{k=0}^{\infty} \int_0^{\infty} f(\tau_{n+1}) \frac{e^{-\gamma\tau_{n+1}}(\gamma\tau_{n+1})^k}{k!} \left(\frac{\gamma}{\beta_1}\right)^{-k} d\tau_{n+1}.$$

Por lo tanto

$$\mathbb{E}[Y_{n+1} \mid Y_n, \ldots, Y_0] = \mathbb{E}\left[\left(\frac{\gamma}{\beta_1}\right)^{X_{n+1}} \middle| \left(\frac{\gamma}{\beta_1}\right)^{X_n}, \ldots, \left(\frac{\gamma}{\beta_1}\right)^{X_0}\right] =$$

$$\mathbb{E}\left[\left(\frac{\gamma}{\beta_1}\right)^{X_{n+1}} \middle| \left(\frac{\gamma}{\beta_1}\right)^{X_n}\right] =$$

$$\left(\frac{\gamma}{\beta_1}\right)^{X_n+1} \sum_{k=0}^{\infty} \int_0^{\infty} f(\tau_{n+1}) \frac{e^{-\gamma\tau_{n+1}}(\gamma\tau_{n+1})^k}{k!} \left(\frac{\gamma}{\beta_1}\right)^{-k} d\tau_{n+1} =$$

$$\left(\frac{\gamma}{\beta_1}\right)^{X_n+1} \int_0^{\infty} f(\tau_{n+1}) e^{-\gamma\tau_{n+1}} \sum_{k=0}^{\infty} \frac{\left(\frac{\gamma\tau_{n+1}}{\frac{\gamma}{\beta_1}}\right)^k}{k!} d\tau_{n+1} =$$

$$\left(\frac{\gamma}{\beta_1}\right)^{X_n+1} \int_0^{\infty} f(\tau_{n+1}) e^{-\gamma\tau_{n+1}} e^{\frac{1}{\frac{\gamma}{\beta_1}}\gamma\tau_{n+1}} d\tau_{n+1} =$$

$$\left(\frac{\gamma}{\beta_1}\right)^{X_n+1} \int_0^{\infty} f(\tau_{n+1}) e^{\gamma\left(\frac{1}{\frac{\gamma}{\beta_1}}-1\right)\tau_{n+1}} d\tau_{n+1}.$$

Luego, como $0 < \gamma < \beta_1$ y $\beta \geq \beta_1$ se sigue de la segunda parte del lema 16 que

$$\int_0^\infty f(\tau_{n+1}) e^{\gamma\left(\frac{1}{\frac{\gamma}{\beta_1}}-1\right)\tau_{n+1}} d\tau_{n+1} - \frac{1}{\frac{\gamma}{\beta_1}} \leq 0 \quad \Rightarrow$$

$$\left(\frac{\gamma}{\beta_1}\right) \int_0^\infty f(\tau_{n+1}) e^{\gamma\left(\frac{1}{\frac{\gamma}{\beta_1}}-1\right)\tau_{n+1}} d\tau_{n+1} \leq 1 \quad \Rightarrow$$

$$\left(\frac{\gamma}{\beta_1}\right)^{X_n+1} \int_0^\infty f(\tau_{n+1}) e^{\gamma\left(\frac{1}{\frac{\gamma}{\beta_1}}-1\right)\tau_{n+1}} d\tau_{n+1} \leq \left(\frac{\gamma}{\beta_1}\right)^{X_n}.$$

Por lo tanto $\mathbb{E}[Y_{n+1} \mid Y_n, \ldots, Y_0] \leq Y_n$.

Ahora tomemos $x_1 < X_0 < x_2$ constantes fijas, y se τ el tiempo de paro definido por $\tau = \text{mín}\{n \mid X_n = x_1 \ o \ X_n = x_2\}$, y finalmente definamos el evento $E_{x_1,x_2} = \{X_\tau = x_2\}$. Así por el Teorema de Paro Opcional de Doob (ver [3], p. 100-101) aplicada a la supermartingala Y obtenemos:

$$\left(\frac{\gamma}{\beta_1}\right)^{X_0} = Y_0 \geq \ \mathbb{E}[Y_\tau] = \mathbb{P}(E_{x_1,x_2}) \left(\frac{\gamma}{\beta_1}\right)^{x_2} + \mathbb{P}(E^c_{x_1,x_2}) \left(\frac{\gamma}{\beta_1}\right)^{x_1} \quad \Rightarrow$$

$$\left(\frac{\gamma}{\beta_1}\right)^{X_0} - \left(\frac{\gamma}{\beta_1}\right)^{x_1} \geq \mathbb{P}(E_{x_1,x_2}) \left(\left(\frac{\gamma}{\beta_1}\right)^{x_2} - \left(\frac{\gamma}{\beta_1}\right)^{x_1}\right) \quad \Rightarrow$$

$$\mathbb{P}(E_{x_1,x_2}) \geq \frac{\left(\frac{\gamma}{\beta_1}\right)^{X_0} - \left(\frac{\gamma}{\beta_1}\right)^{x_1}}{\left(\frac{\gamma}{\beta_1}\right)^{x_2} - \left(\frac{\gamma}{\beta_1}\right)^{x_1}}.$$

Tomando $x_1 = -1$ se tiene $\lim_{x_2 \to \infty} \mathbb{P}(E_{x_1,x_2}) \geq 1 - \left(\frac{\gamma}{\beta_1}\right)^{X_0+1}$. Entonces la probabilidad de que la brecha entre la cadenas nunca alcanze -1 está acotada por debajo por $1 - \left(\frac{\gamma}{\beta_1}\right)^{X_0+1}$, y por lo tanto la probabilidad de que el atacante tenga éxito es a lo más $\left(\frac{\gamma}{\beta_1}\right)^{X_0+1}$.

■

Bibliografía

[1] https://blockchain.info/charts/n-transactions

[2] A. Narayanan, J. Bonneau, E. Felten, A. Miller, y S. Goldfeder, Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction, Princeton University Press, (2016).

[3] D. Williams, Probability with martingales, Cambridge University Press (1991).

[4] L. A. Rincón S., Introducción a los procesos estocásticos, Las prensas de ciencias, (2012).

[5] M. E. Caballero, Modelos elementales para el problema de la ruina, Miscelánea Matemática 60 (2015).

[6] M. Rosenfeld, Analysis of hashrate-based double-spending. arXiv preprint arXiv:1402.2009, (2014).

[7] S. Nakamoto, A peer-to-peer electronic cash system, (2008).

[8] Y. Sompolinsky, A. Zohar, Secure high-rate transaction processing in bitcoin, Financial Cryptography and Data Security, International Conference on Financial Cryptography and Data Security, FC 2015, p. 507–527 (2015).