FACULTAD DE INGENIERÍA – UDELAR
Montevideo, Uruguay

Análisis y Evaluación de Ataques a *Pools* de Minería en Bitcoin

Informe de Proyecto de Grado presentado al Tribunal Evaluador como requisito de graduación de la carrera Ingeniería en Computación

Mauro Saravia
Jonathan Javiel
Federico Paredes

Supervisores
Alfredo Viola
Germán Ferrari

Diciembre 2018

AGRADECIMIENTOS

Los autores expresan sus más sinceros agradecimientos a:
Los tutores del proyecto Alfredo Viola y Germán Ferrari por dedicarle su tiempo en la colaboración y guía del trabajo pese a las circunstancias especiales que en este se presentaron.
A la familia y amigos de cada uno de los integrantes del equipo por prestar su apoyo incondicional en todo momento.
A Octavio Perez por colaborar con la lectura y sugerencias sobre este trabajo.
A los compañeros de otros proyectos de grado relacionados con *blockchain* por compartir instancias de discusión e intercambio de opiniones.

RESUMEN

En la actualidad, minar bitcoins en forma privada requiere una inversión que pocos pueden pagar. Esto hace que las agrupaciones de mineros llamadas *pools* de minería (*mining pools* o simplemente *pools*) sean de suma importancia para Bitcoin, donde la mayor parte del poder de cómputo global ya se encuentra repartido en estas agrupaciones. En este trabajo, se elaboran criterios para clasificar y evaluar ataques que involucran a los *mining pools*. Se describen en detalle tres ataques presentados en la literatura, que se consideran relevantes en base a dichos criterios. Además se aplican los criterios elaborados para evaluar y comparar los tres ataques: *BiteCoin*, *Selfish Mining* y *Fork After Withholding*. Se concluye que actualmente es posible aplicar y obtener ganancias con cada uno de ellos, haciendo que la criptomoneda no sea tan segura como fue planteada en sus inicios.

TABLA DE CONTENIDO

Capítulo 1

INTRODUCCIÓN

Las criptomonedas han cautivado la atención del mundo entero en los últimos años. La primera y más popular de ellas, Bitcoin, inició su camino en el año 2008 con la aspiración de convertirse en una alternativa viable a los sistemas bancarios tradicionales [1]. Su éxito ha abierto las puertas a un nuevo paradigma capaz de descentralizar múltiples medios, de la mano de la tecnología *blockchain* [2]. El creciente interés económico también ha inspirado un nuevo mercado especulativo que pone a prueba a la descentralización [3]. La motivación subyace en el proceso para generar transacciones, conocido como minería. Cada participante de Bitcoin puede minar un bloque de transacciones, lo cual significa agregarlo a una estructura de cadena de bloques denominada *blockchain*. Para ello, deben aportar poder de cómputo a la resolución de un problema criptográfico. Aquel participante capaz de resolverlo recibe a cambio un pago en bitcoins, el cual disminuye a medida que se resuelven más de estos problemas, actualmente 12,5 BTC, equivalentes a aproximadamente cincuenta mil dólares al momento de la redacción de este trabajo. Debido a que un incremento en poder de cómputo aumenta las probabilidades de recibir la recompensa, la competencia crece rápidamente. La seguridad de Bitcoin se correlaciona con una buena distribución del poder de cómputo, y su centralización es una de las principales preocupaciones de la comunidad. En este contexto se elabora un protocolo que permite a los mineros formar agrupaciones denominadas *pools* de minería [4] (*mining pools*), donde se combina el poder de cómputo aportado por cada minero. Cada *mining pool* cuenta con un administrador que recibe las contribuciones de cada uno de sus mineros y reparte las ganancias acorde a lo aportado. Algunos de estos *pools* son públicos, permitiendo que cualquier minero se suscriba y forme parte de la agrupación. Hoy en día, los seis *pools* más populares contribuyen con el 63,5 % del poder de cómputo [5]. Esta acumulación de poder vuelve a generar preocupación, y a propulsar una fuerte investigación sobre las debilidades y amenazas que puedan existir.

En este trabajo se elaboran criterios para clasificar y evaluar aquellos ataques que se encuentran disponibles en la literatura, donde el componente principal

del ataque son los *mining pools*. En el capítulo 2, se presenta todo el estado del arte que es necesario comprender previo a abordar la descripción y evaluación de los ataques. La sección 2.1 introduce los conceptos básicos de las criptomonedas, así como las primitivas criptográficas relevantes, mientras que en las secciones 2.2 y 2.3 se detalla el funcionamiento de Bitcoin y los *mining pools* respectivamente.

El capítulo 3 comienza con la presentación de los criterios elaborados para la clasificación de ataques que involucran *mining pools*, seguido de criterios para evaluar y comparar los mismos. Tres de los ataques relevados son seleccionados en base a estos criterios de clasificación. En las secciones 3.2, 3.3 y 3.4 se detalla por separado el funcionamiento y la evaluación de cada ataque. La sección 3.5 resume y compara los resultados de clasificar y evaluar los tres ataques.

Por último, en el capítulo 4 se discuten las conclusiones encontradas, y en el capítulo 5 se dan a conocer aquellos puntos que quedaron por fuera del alcance de este trabajo, pero que pueden ser de interés para continuar ampliando el conocimiento en la temática.

Los objetivos principales de este trabajo son:

- Comprender el funcionamiento de la tecnología *blockchain*, los *mining pools* y su aplicación en Bitcoin.
- Hacer un relevamiento de los ataques existentes en la literatura, que involucran en forma directa a los *mining pools*.
- Definir criterios para clasificar y evaluar los ataques relevados.
- En base a los criterios definidos, seleccionar tres ataques, analizar su funcionamiento y evaluarlos en base a los criterios.
- Comparar los resultados de evaluar cada criterio en los ataques seleccionados.

Capítulo 2

REVISIÓN DE ANTECEDENTES

2.1 Criptomonedas

2.1.1 Introducción

En el dinero Fiat [6] que es usado en la actualidad, es necesaria la intervención de una entidad central que regule la emisión y proponga medidas para prevenir su falsificación. Una criptomoneda es una propuesta alternativa en la que se busca prescindir de la entidad central, reemplazándola por tecnología. Como su nombre sugiere, el componente tecnológico principal es la criptografía; un campo de estudio que utiliza técnicas de matemática avanzada para proveer mecanismos de seguridad en la comunicación de información [7] [8]. Una criptomoneda, consiste en una red de pares [9] (*peer-to-peer* o P2P) que hace uso de protocolos de sistemas distribuidos combinados con un fuerte uso de algunas primitivas criptográficas. Cuentan con su propia moneda, con la cual sus participantes pueden realizar transacciones sin intervención de ninguna organización central o banco que regule el proceso. El sistema gira entorno a una estructura de datos específica, denominada *blockchain*, que contiene todas las transacciones que se realizan. Esta se encuentra pública en internet, visible a todos los integrantes de la red P2P que conforma la criptomoneda. Dado este último punto, hay ciertas propiedades imprescindibles que el uso de la criptografía busca asegurar a sus participantes. Por una parte, es necesario garantizar que cada participante cuente con el dinero que efectivamente le pertenece. El dinero debe poder ser transferido sin inconvenientes a los demás participantes en una forma unívoca, en la que todos puedan tener la tranquilidad y la posibilidad de verificar que no se cometió un fraude. Por otra parte, igual de importante, que a pesar del hecho que las transacciones sean públicas, los participantes cuenten con un cierto grado de privacidad, similar al que tienen en los sistemas tradicionales.

El resto de esta sección introduce las bases criptográficas necesarias para comprender los protocolos utilizados en la implementación de Bitcoin. Ellas

son: funciones de *hash* [10], punteros *hash*, y firmas digitales[11].

Definición 2.1.1. Función de *hash*
Una función de *hash* es cualquier función matemática que correlaciona entradas de tamaño variable a una salida de tamaño fijo. A la salida comúnmente se le llama *hash* o *digest*.

Definición 2.1.2. Colisión
Sea $h : A \longrightarrow C$ una correlación entre 2 conjuntos finitos A y C. Si $a \neq b$, $a, b \in A$ / $h(a) = h(b)$ entonces a y b colisionan, y (a,b) es una colisión.

Definición 2.1.3. Función de *hash* criptográfica
Una función de *hash* criptográfica H es una función de *hash* que cumple con las siguientes propiedades:

- Resistencia a colisiones - H se dice que es resistente a colisiones si la probabilidad de encontrar dos valores que colisionen es despreciable.
- Ocultamiento - Dado $c = h(a)$, obtener $a = h^{-1}(c)$ es un valor difícil de computar en un tiempo razonable.
- *Puzzle-friendly* - H es *puzzle-friendly* si para cada salida c de n bits, dado k elegido de una distribución con alta entropía, la probabilidad de encontrar un a / $H(k||a) = c$, en un tiempo mucho menor a 2^n, es baja.

En el contexto de las criptomonedas, cuando se habla de funciones de *hash* siempre refiere a funciones de *hash* criptográficas.

Definición 2.1.4. Puntero *hash*
Un puntero *hash* es un puntero a donde se encuentra un conjunto de datos D, junto con el valor $c = H(D)$ calculado en algún momento fijo en el tiempo.

Definición 2.1.5. Firma digital
Una firma digital es un esquema matemático que permite verificar la autenticidad de un documento. El esquema está conformado por tres algoritmos:

- $(sk, pk) := generadorClaves(tamaño\_clave)$ - Función que genera un par de claves, comúnmente denominadas clave privada y clave pública respectivamente.
- $f := sign(sk, mensaje)$ - Función que dado un mensaje y la clave privada sk, devuelve una firma para dicho mensaje bajo sk.

- $esValida := verificar(pk, mensaje, f)$ - Función que a partir de la clave pública pk, un mensaje y una firma, retorna verdadero si f es una firma valida para el mensaje bajo pk, y falso en cualquier otro caso.

Para que el esquema sea útil debe garantizar que una firma válida siempre sea reconocida como tal para el mensaje deseado, y que ningún atacante sea capaz de falsificar una firma conociendo las firmas de mensajes anteriores y la clave pública. Por este motivo, un factor de vital importancia en el algoritmo se encuentra en la fuente de aleatoriedad usada en el generador de claves. De otra forma, no solo puede haber vulnerabilidades en el esquema, sino que dos participantes que utilicen el mismo *software* pueden obtener la misma clave. Otro factor importante es el tamaño de un documento que puede ser firmado por un algoritmo que siga este esquema. Sin embargo, gracias a las propiedades de las funciones de *hash* criptográficas, es posible aplicar el esquema al *hash* de un documento, que ya tiene un tamaño fijo y conocido asociado a la función elegida.

El esquema de firmas digitales ofrece a su vez la propiedad de poder utilizar las claves públicas como identidades. Considérese un par de claves (sk, pk). Si el esquema funciona correctamente, pk es la única clave que puede verificar todos los documentos que sean firmados con sk. En este sentido, los documentos se encuentran asociados a pk , permitiendo que pueda ser usada como una identidad del participante que posee sk. La ventaja de usar pk como identidad yace en la capacidad del usuario de crear múltiples pares de claves, no teniendo que asociar ninguna de estas identidades a su verdadera identidad.

2.1.2 La *blockchain*

La *blockchain* es una lista encadenada que en lugar de simples punteros, utiliza punteros *hash* [10]. Cada bloque de la cadena es un conjunto de datos sensibles, en este caso transacciones, que se quieren ingresar a la *blockchain*. Es importante que las transacciones no puedan ser modificadas en un punto posterior en el tiempo sin que cualquiera sea capaz de comprobarlo. A esta propiedad se le denomina *Tamper-Evident*, y es lo que motiva el uso de punteros *hash*. Si un atacante modifica el contenido de una transacción, como el *hash* del bloque que la contiene no es el mismo que el *hash* previo a la

modificación, el atacante debe modificar el valor del puntero *hash* del bloque siguiente. De lo contrario, cualquiera que vea la estructura puede determinar que el bloque fue alterado, ya que el *hash* no coincide con el valor almacenado. Luego de que el atacante modifica el *hash* almacenado en el bloque siguiente, el *hash* de este bloque también es distinto al *hash* previo, por lo que debe modificar el puntero *hash* del bloque que le sigue, y de todos los bloques que le siguen. Por ende, basta con almacenar en forma segura el puntero *hash* que referencia al primer bloque de la estructura para que sea posible verificar en cualquier momento si se ha alterado información en cualquier punto de la misma. (Figura 2.1).

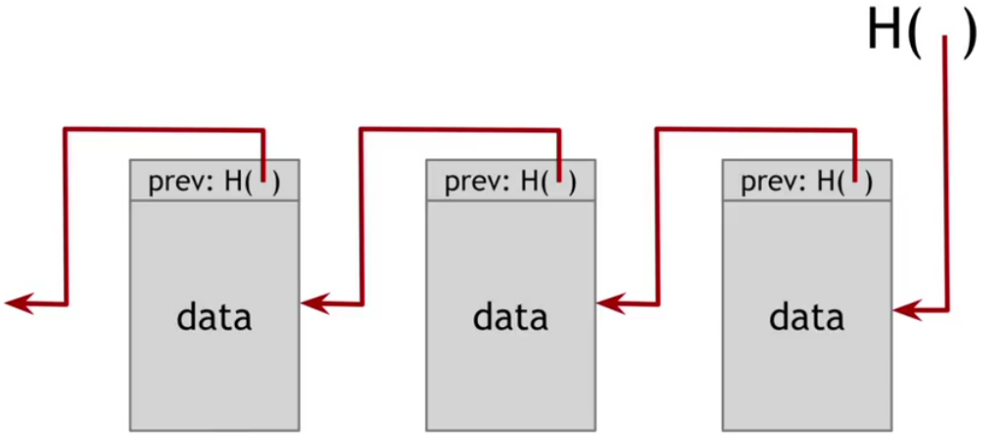

Figura 2.1: Esquema de la estructura de datos *blockchain*, formada por bloques conectados por punteros *hash* (Figura 1.5 en [10]).

La *blockchain* es la estructura principal, pero no es la única estructura de importancia usada que se basa en punteros *hash*. En la criptomoneda Bitcoin, si bien la *blockchain* contiene bloques de transacciones, cada transacción se encuentra a su vez en una estructura arborescente llamada árbol de Merkle [12] (*Merkle Tree*). Un *Merkle Tree* es un árbol binario donde las hojas son transacciones y los nodos intermedios contienen punteros *hash* para formar la estructura arborescente. Al igual que la *blockchain*, los *Merkle Tree* son *Tamper-Evident* y almacenar tan solo el *hash* de la raíz es suficiente para que no sea posible alterar una transacción sin volver a recalcular todos los *hashes* que forman el camino de la hoja a la raíz [10].

2.2 Bitcoin

2.2.1 Introducción

Bitcoin es la primera criptomoneda en funcionamiento, introducida por Satoshi Nakamoto en el año 2009, y desde entonces es la criptomoneda con mayor popularidad. Esta sección se basa principalmente en el libro de la universidad de Princeton [13], y en el primer artículo publicado por Satoshi Nakamoto [1]. La moneda en Bitcoin toma el mismo nombre que la criptomoneda, donde la unidad mínima es denominada un satoshi, si bien la unidad estándar que se utiliza comúnmente es el bitcoin (o BTC), donde 1 BTC = 10^8 satoshi. Los bitcoins se encuentran asignados a una clave pública, o más específico, al *hash* de una clave pública, que en este contexto se denomina dirección. Los participantes poseen una cierta cantidad de bitcoins en la medida que cuenten con la clave privada correspondiente a la dirección a la cual fueron enviados, siempre y cuando aún no hayan sido gastados en otra transacción. Un participante usualmente descarga un *software* denominado Wallet [14], que se encarga de la creación y el almacenamiento de claves, y determina a su vez el balance de bitcoins del participante. Los bitcoins no se encuentran representados directamente en ninguna estructura. Para comprender como se determina el balance de un participante, es necesario analizar el contenido de una transacción.

2.2.2 Transacciones

Una transacción es un conjunto de datos con los siguientes campos importantes:

- Entradas - Un vector que indica cuáles bitcoins se transfieren durante la transacción. No es explicitado en forma numérica, sino que se da una referencia a una transacción previa en la cual se recibieron estos bitcoins. La referencia se hace por medio de un puntero *hash* a la transacción de la que provienen. La transacción referenciada puede contener múltiples salidas, debiendo indicarse el índice en el vector de salidas de dicha transacción. Cada entrada debe ir acompañada de una firma digital. La

firma digital tampoco se encuentra explícita, sino que está embebida en un *script* de un lenguaje específico de Bitcoin.

- Salidas - Otro vector que refiere a quiénes reciben los bitcoins transferidos. Para cada destinatario se tiene un monto, que sí es un valor numérico, y en lugar de tener simplemente un *hash* de la clave pública para indicar al destinatario, se incluye otro *script*. Los *scripts* de salida se combinan con los de entrada para formar un programa de este lenguaje que permite validar la transacción. Al referenciar una salida en una entrada, no se puede especificar el monto a utilizar. Si el monto que se desea enviar es inferior a la cantidad de bitcoins referenciada, se debe agregar una salida que transfiere el monto sobrante.
- *Hash* de la transacción - sirve de identificador para los punteros *hash*.
- Metadata - los datos restantes ofrecen información respecto al tamaño de la transacción, cantidad de entradas y salidas, y un sello de tiempo (*timestamp*).

En la sección 2.2.3 se define la validez de una transacción. En lo que resta de esta sección se presupone que todas las transacciones son válidas. Entendiendo que cada entrada de una transacción no es más que una referencia a una salida de una transacción previa, se puede introducir otro concepto importante en Bitcoin: las UTXO.
Las UTXO (*Unspent transaction output*), son salidas de transacciones, que aún no han sido referenciadas por una entrada de otra. En esencia, son los bitcoins que pueden ser gastados. Dado un participante, todos los bitcoins que estén asociados a una UTXO con una dirección para la cual el participante tiene la clave privada, son los bitcoins que el participante puede gastar.

Cuando el participante desea realizar una transacción, entra en juego el uso de la red P2P. La transacción es construida haciendo uso de un *software* denominado cliente Bitcoin. Los desarrolladores ofrecen un cliente por defecto, que es usado ampliamente en la red. La transacción es propagada a través de la red por medio de un algoritmo de inundación de red [15] (*flooding*) al resto de los participantes. Todo participante puede optar por convertirse en un nodo especial en la red, denominado minero. Los mineros son los responsables de hacer el trabajo necesario para ingresar transacciones a la *blockchain*. Los

mineros juntan estas potenciales transacciones que son propagadas por la red para armar bloques que son los que efectivamente conforman la *blockchain*.

2.2.3 Consenso a través de *Proof of Work*

Si bien se habla de la *blockchain*, no tiene por qué ser una sola. Desde el punto de vista de la tecnología, nada impide que un conjunto de mineros inicie su propia cadena de bloques, o que a partir del enésimo bloque de la cadena, se originen dos o más cadenas que tienen los primeros n bloques en común. Sin embargo, es evidente que no todas estas cadenas pueden ser aceptadas como válidas. Si una cadena contiene referencias a bitcoins que ya fueron gastados en otra, los punteros *hash* que referencian a una cadena no pueden ofrecer información acerca de si se modificaron valores en la otra, y se vuelve más compleja la tarea de llevar un registro de transacciones sobre el cual todos los participantes estén de acuerdo. Para que el sistema funcione, se debe lograr un consenso entre los participantes acerca de cuál es la *blockchain* y qué transacciones son válidas. El algoritmo de consenso utilizado en Bitcoin sigue un paradigma conocido como prueba de trabajo (*Proof of Work* o PoW).

Definición 2.2.1. Esquema de *Proof of Work*
Dado un minero M de Bitcoin, el *Proof of Work* funciona de la siguiente forma:

- M considera a la *blockchain* como la cadena más larga de bloques que se origina desde un bloque especial denominado bloque génesis, minado por Satoshi Nakamoto. Comúnmente se la denomina la cadena principal.
- M recibe transacciones de sus nodos vecinos en la red P2P. Si no tiene registro de la transacción, se la envía a su vez a sus demás nodos vecinos.
- M valida las transacciones, comprobando que las firmas digitales sean correctas y los bitcoins formen parte de una UTXO.
- M agrupa las transacciones en un bloque que apunte al último bloque de la cadena principal, y agrega una transacción especial al *Merkle Tree* denominada transacción de *Coinbase*, en la que no hay entradas, pero que en la salida puede destinar un número determinado de bitcoins a una dirección que el minero elija. Esta es una recompensa que busca incentivar a los mineros a comportarse honestamente.

- Una vez conformado el bloque, M debe encontrar un valor de 32 bits, que se denomina *Nonce*. El *Nonce* debe ser tal, que cuando se compute el *hash* del bloque, el *hash* resultante cuente con un número determinado de ceros a la izquierda, o dicho de otra forma, sea menor que un valor objetivo.
- Si M encuentra un *Nonce* válido, debe proponer el bloque a la red enviándolo a sus nodos vecinos.
- Durante todo el proceso, M se encuentra recibiendo bloques de sus vecinos. M valida las transacciones, y verifica que el *Nonce* hallado cumple con las restricciones impuestas. Si el bloque es válido y M aún no propagó un bloque que apunte al mismo bloque, ni recibió otro que lo haga, entonces descarta el bloque sobre el que se encuentra trabajando y comienza nuevamente sobre el nuevo bloque de lo que considera la cadena principal.

Debido a que los bloques se propagan en una red P2P, puede que por latencias en la red, o por la propia libertad de los mineros de elegir el bloque sobre el cual minar, existan dos cadenas de igual largo. El comportamiento por defecto es que M trabaje sobre el bloque que reciba primero. Como consecuencia, puede ocurrir que en un momento dado, parte de la red se encuentre minando sobre una cadena distinta al resto. Lo importante es que eventualmente una de las cadenas será mas larga que la otra, y al enterarse de su existencia, si M se comporta honestamente, debe dejar de minar en su cadena y pasar a minar en la que ahora es la cadena más larga y por tanto la principal.

¿Cuándo se puede estar seguro de que un bloque pertenece a la cadena principal?

Cuando un bloque recién ingresa a la *blockchain*, se le considera un bloque con cero confirmaciones. Luego de que un minero haya ingresado un bloque que extiende la cadena que lo contiene, se lo considera un bloque con una confirmación. En ningún momento existe una garantía absoluta, pero es considerado que seis confirmaciones es un valor aceptable, ante el cual la probabilidad de que el bloque no pertenezca en algún momento a la cadena principal es despreciable. Esta consideración parte de la base de que la mayoría de los mineros se comporten en forma honesta.

Dificultad del *Proof of Work*

El requisito de que el *Nonce* agregado produzca un *hash* del bloque con una determinada cantidad de ceros a la izquierda, es equivalente a decir que pertenezca a un subespacio del espacio total de *hashes*. Este tipo de problema criptográfico se denomina *hash puzzle*. En Bitcoin, la dificultad del *hash puzzle* es sumamente importante. Uno de los motivos es limitar los problemas que puedan surgir por latencias en la red. Si la frecuencia de bloques es muy alta, aumenta la probabilidad de que las latencias afecten al proceso de consenso. Por otra parte, para que el esfuerzo invertido, aún con el costo asociado en tiempo y electricidad, motive a los mineros a trabajar en beneficio de que la moneda siga prosperando y la recompensa obtenida tenga un valor que dé ganancia. El *hash puzzle* debe cumplir las siguientes propiedades:

- Ser difícil de computar - Si la función de *hash* cumple con la propiedad de *puzzle-friendly* requerida en la definición 2.1.3, encontrar un *Nonce* tal que el *hash* sea menor a un determinado valor, es modelable por un ensayo de Bernoulli [16], en el que el minero debe probar con distintos *Nonces* hasta que uno derive en un bloque válido. No hay ninguna estrategia en la selección del *Nonce* que sea más favorable. Si el subespacio tiene un tamaño de $\frac{1}{20}$ del espacio total de *hashes*, como lo era a finales del año 2014, es de esperar que se deban computar 10^{20} hashes antes de encontrar el *Nonce* adecuado.

- Ser parametrizable - La dificultad ha ido incrementando con el tiempo. Esto se debe a que Bitcoin establece que la dificultad sea recalculada cada un valor arbitrario de 2016 bloques, y la cantidad de poder de cómputo ha ido aumentando con el aumento en popularidad de Bitcoin y las mejoras en los procesadores. El reajuste se hace de manera que la tasa de creación de bloques se mantenga en un promedio de un bloque cada diez minutos. Sea D el valor de la dificultad y T el tiempo requerido para minar los últimos 2016 bloques, entonces

$$D_{siguiente} = \frac{D_{anterior} * 2016 * 10\ minutos}{T}.$$

- Ser fácil de verificar - Para que cada minero pueda validar los bloques recibidos por otros mineros, es importante que el proceso de verificación sea rápido. Como el bloque incluye al *Nonce* encontrado, la validación

requiere calcular un único *hash* y comprobar que cumple con la dificultad.

2.2.4 ¿Dónde se almacena la *blockchain*?

La *blockchain* es pública, pero no hay ninguna base de datos central donde almacenarla. Como es común en los sistemas P2P, son los propios participantes quienes conservan una copia de la *blockchain*. La *blockchain* completa se encuentra actualmente superando los 180 GB de espacio requerido para almacenarla. Es poco practicable que todos deban descargarse la *blockchain* completa para poder participar. Sin embargo, no toda la información de cada bloque debe ser almacenada por todos los participantes. Un bloque de transacciones se compone de los siguientes ítems:

- Versión - número que informa al minero cómo interpretar la información.
- Un *timestamp*.
- Puntero *hash* al bloque anterior.
- Un campo de 32 bits que indica la dificultad del PoW.
- *Nonce*.
- Un valor igual a la cantidad de transacciones del bloque.
- El tamaño del bloque en bytes.
- *Hash* de la raíz del *Merkle Tree* de transacciones.
- *Merkle Tree* de transacciones.

Esta información se separa en lo que es el cabezal del bloque, y el árbol de transacciones. Cuando se habla del *hash* del bloque, a lo que en verdad se hace referencia es al *hash* del cabezal del bloque, que contiene toda la información excepto el último ítem. Al almacenar la raíz del *Merkle Tree*, la propiedad de *Tamper-Evident* del *Merkle Tree* se encuentra contenida en el *hash* del bloque. Esta distinción permite separar entre lo que son nodos completos (*Fully validating nodes*) y nodos livianos (*Lightweight nodes*). Los nodos completos,

son mineros que almacenan la *blockchain* completa y así pueden verificar la validez de los cabezales de los bloques y de cada transacción. Los nodos livianos, en cambio, solo almacenan la lista encadenada de los cabezales de los bloques, y solicitan a los demás nodos las partes de los *Merkle Tree* que necesiten. Por ejemplo, aquellas partes que contengan las transacciones con direcciones de un participante. Almacenar tan solo los cabezales reduce el costo a los cientos de MB.

2.2.5 Desafíos de una criptomoneda

Bitcoin enfrenta algunos desafíos que son comunes a todas las criptomonedas que se basan en la tecnología *blockchain*. En esta sección se describen algunos de los desafíos de mayor importancia.

Ataque de doble gasto

Toda moneda debe garantizar que no pueda ser gastada dos veces. El protocolo de consenso, junto con las precauciones respecto al número de confirmaciones, son herramientas esenciales en la prevención de este problema. El escenario es el de un atacante que genera una transacción en la que envía un conjunto de bitcoins a otro participante a cambio de un servicio o bienes, potencialmente ajenos a Bitcoin. El atacante puede, una vez hecho el intercambio, generar una nueva transacción que referencia a los mismos bitcoins que acaba de gastar. En la medida que la primera transacción se encuentre en la cadena principal, los mineros honestos pueden ver que la nueva transacción no es válida, pues no referencia en su entrada a una UTXO. Sin embargo, si el atacante tiene suficiente poder de cómputo, o es capaz de influenciar a otros mineros a continuar minando en una cadena alterna, esta puede llegar a convertirse en la cadena principal. En este caso, es la primera transacción la que deja de ser válida ya que las UTXO son tomadas en base a la cadena principal. Esto es lo que se denomina ataque o problema del doble gasto. El bloque o rama que divide a la cadena de la *blockchain* es denominado bifurcación (*fork*), y el o los bloques que finalmente no forman parte de la cadena principal se consideran bloques huérfanos (Figura 2.2).

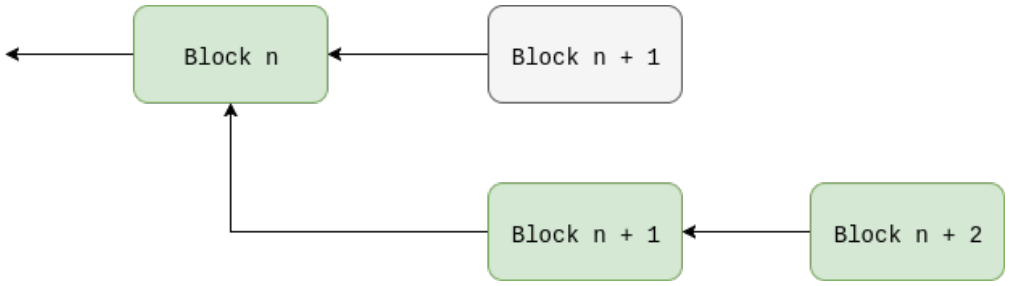

Figura 2.2: *Fork* en la *blockchain*. La cadena verde es actualmente la cadena que los mineros honestos favorecerán, y el bloque $n+1$ de la otra cadena es candidato a ser un bloque huérfano.

Desde el punto de vista moral es claro que la segunda transacción es la que comete fraude, pero desde el punto de vista tecnológico, ambas transacciones pueden ser válidas. Es el protocolo de consenso el que decide la validez, y por ello es clave que la mayoría de los mineros sean honestos.

Ataque del 51 %

Si bien el objetivo de Bitcoin es descentralizar la moneda, cuando se trata de decidir qué bloques ingresan a la *blockchain*, el poder ya no se encuentra tan descentralizado. Dado que el consenso se basa en poder de cómputo, y minar bloques es una fuente de mucho dinero, se han hecho inversiones en desarrollar procesadores especializados en resolver con mayor rapidez las tareas involucradas en el PoW. Estos procesadores, denominados ASIC (Circuitos integrados de aplicación específica), son costosos y han centralizado gran parte del poder de minería.
Un atacante o conjunto de atacantes que alcance el 51 % del poder de cómputo, puede minar bloques mucho más rápido que los demás participantes. El sistema controla la dificultad para que no haya bloques generados a más de un bloque cada diez minutos, pero este atacante es capaz de generar cada uno de estos bloques con mayor probabilidad que el resto de la comunidad, permitiéndole manipular la *blockchain* en la forma que desee, potencialmente generando doble gastos o ignorando/favoreciendo determinadas transacciones. El resto de la comunidad aún puede validar el estado de la *blockchain* y minar en otra cadena, pero el sistema se vuelve vulnerable a fraudes y la moneda inestable.

Anonimato

Dado que todas las transacciones realizadas se encuentran públicas al alcance de todos, es importante que Bitcoin pueda ofrecer a sus participantes cierto anonimato. Sin embargo, lo que Bitcoin ofrece es denominado pseudoanonimato. No hay ningún tipo de información pública que asocie al participante con su identidad real. La identidad está en las claves públicas usadas como direcciones. El problema es que al encontrarse relacionadas las entradas con las salidas de transacciones previas, es posible determinar la o el conjunto de direcciones usadas por un participante. Si no se almacenan adecuadamente las claves y se logra asociar una de ellas a una identidad real, se puede llegar a comprometer el anonimato de todas las transacciones realizadas. El estudio de estos problemas y sus soluciones no son foco de este estudio, pero por más información se puede consultar la literatura referida en [17]

2.2.6 Otras características de Bitcoin

- Función de *hash* - Bitcoin eligió la función estándar de la NIST al momento de su creación, llamada SHA256. En particular, para obtener el *hash* de los bloques y de los *Merkle Tree* se aplica SHA256 dos veces, para incrementar la seguridad, evitando un posible ataque conocido como *length extension attack* [18]. En las direcciones de los participantes también se utiliza un *hash* doble, pero si bien el primer *hash* es SHA256 sobre la clave pública, el segundo *hash* es otro llamado RIPEMD-160, debido a que produce una salida más corta de 160 bits.

- Recompensas - Los incentivos en Bitcoin no son constantes. Cada doscientos diez mil bloques se reduce a la mitad la recompensa por minar un bloque. El número total de bitcoins que pueden llegar a existir se encuentra limitado a veintiún millones. Aún así, existe otra forma de compensar a los mineros, hacia la cual se espera ir transicionando para que los mineros sigan contando con un incentivo que fomente el comportamiento esperado. El participante que crea la transacción puede colocar un monto menor en las salidas de la transacción que los recibidos en las entradas, y el minero puede recolectar la diferencia para sí mismo. No es obligatorio, pero los mineros pueden favorecer aque-

llas transacciones que les den una mejor recompensa, otorgándoles un servicio más rápido a quienes lo hagan.

- Lenguaje *script* - Los *script* de las entradas y salidas que permiten validar la transacción forman parte de un lenguaje que puede llegar a ser usado para realizar transacciones más complejas, que por ejemplo requieran presentar múltiples firmas digitales para que se efectúe la transacción. Este lenguaje, sin embargo, se encuentra muy limitado en su expresividad para poder garantizar la finalización de cada programa, a diferencia de los lenguajes en otras criptomonedas. En el capítulo 5 se introduce en forma breve la importancia que estos lenguajes pueden tener en continuar el desarrollo de este trabajo.

2.3 *Pools* de Minería

2.3.1 Introducción

En la sección 2.2.5 se habla acerca de la preocupación de la comunidad Bitcoin respecto a la centralización del poder computacional. Este hecho, junto con que la varianza de la ganancia es grande [19] [20], llevó al desarrollo de un protocolo por encima de Bitcoin (y otras criptomonedas), mediante el cual varios mineros pueden minar en conjunto, sin necesidad de que exista confianza entre todos ellos. A estas agrupaciones mineras se les denomina *pools* de minería [4] (*mining pools* o simplemente *pools*). Cada agrupación consiste de dos clases de participantes: los mineros que se unen al *pool* para resolver el PoW, y un administrador comúnmente denominado *Pool-Manager*. Este administrador es quien se encarga de delegar el trabajo a los mineros, recibir las recompensas de minar un bloque, y repartirlas entre los mineros de manera proporcional al esfuerzo que realizaron.

2.3.2 ¿Cómo medir el esfuerzo de un minero?

Para que la repartición de la ganancia se pueda hacer con un esquema justo, todos los participantes del *pool* deben dar una muestra del trabajo realizado. Una forma de estimar la cantidad de poder computacional invertido, es fijando una dificultad menor a la que tiene Bitcoin, para que cada minero pueda conseguir, con mayor facilidad, *hashes* válidos para esa dificultad. A las soluciones que cumplen con la dificultad del *pool* se les llama acciones (*shares*). Cada *share* puede ser un PoW completo (*Full*-PoW o FPoW) si cumple a su vez con la dificultad de Bitcoin, o un PoW parcial (*Partial*-PoW o PPoW) si tan solo cumple con la dificultad del *pool*. El administrador se encarga de verificar la correctitud de las *shares*, y a su vez controla si además de cumplir con la dificultad fijada para el *pool*, alguna cumple con la dificultad actual de Bitcoin, es decir, si es un FPoW. En caso de que así sea, se encarga de propagar el bloque en la red. Finalmente, el administrador debe repartir las ganancias entre los participantes. En la mayoría de los casos, el administrador se queda con un porcentaje de las ganancias (aproximadamente 1 %) como comisión por administrar el *pool* [20].

2.3.3 Implementación

Para implementar los *mining pools* se han utilizado diferentes protocolos. En esta sección se explica el funcionamiento de algunos de ellos:

Getwork

Getwork es el primer protocolo ampliamente aceptado por la comunidad de *pools* en Bitcoin. Los comienzos de este protocolo no tienen una fecha clara. Sin embargo, es posible tomar como referencia la primera fecha en que se observa un bloque minado por un *pool*, que es el 16 de diciembre del año 2010, minado por Slush-Pool. Getwork se encuentra obsoleto desde fines del año 2012 por la aparición del protocolo Stratum.

Características [21]:

- Trabaja sobre HTTP.
- Envía mensajes JSON-RCP.
- Cada conexión http permite operar a un máximo de 4.2 GHash/s.
- No es cifrado, siendo así vulnerable a varios tipos de ataques. No es utilizado con HTTPS por las implicancias en el costo adicional respecto al poder de procesamiento.

El administrador crea un servidor HTTP al cual los mineros se conectan para poder solicitar y enviar la información respecto a los trabajos a realizar. El problema principal es que el servidor no tiene forma de enviarle información al minero sin que este la pida (por basarse en el protocolo HTTP). Por lo tanto, se requiere que el minero consulte continuamente si hay nueva información. A su vez, El procesamiento de los cabezales HTTP conlleva un gasto extra de tiempo de procesamiento, tanto para el administrador como para los mineros.

Stratum

Este protocolo se origina debido a las carencias de Getwork.

Características [22]:

- Comunicación sobre TCP. Los administradores pueden enviar información a los mineros registrados al *pool* sin necesidad de que la pidan.
- El contenido de los paquetes TCP intercambiados son mensajes JSON. De esta forma, no se generan problemas de compatibilidad, ya que todo minero tiene una librería para generar e interpretar código JSON.
- Permite operar a una velocidad de hasta 18 EHash/s por conexión TCP.
- No cuenta con demasiada documentación oficial, y gran parte de la que hay disponible se encuentra desactualizada.
- Implementa un mecanismo de asignación única de trabajo a los mineros a través de un conjunto de *N* bits, denominado *Extranonce*, que son asignados a cada minero. Estos *N* bits son fijados por el minero como parte del *Nonce*.
- Al igual que Getwork, no es cifrado, haciéndolo vulnerable a varios tipos de ataques. No se utiliza sobre TLS por las implicancias en el costo adicional respecto al poder de procesamiento.

Definición 2.3.1. Esquema de Stratum
Sea M un minero que se quiere unir a un *pool* P.

- M se registra en P, indicando usuario, contraseña y clave pública (en la que recibe las recompensas).
- M descarga un *software* que implemente el protocolo Stratum, y en el mismo debe configurar el nombre de usuario, contraseña (si P la requiere), dominio y puerto de acceso a P.
- M envía un mensaje de suscripción. P lo autoriza enviándole un valor hexadecimal denominado *Extranonce*1, un valor numérico denominado *Extranonce*2*_size*, y el valor de dificultad para las *shares*.
- A continuación, P envía a M un mensaje de asignación de trabajo con la siguiente información:
  - *job_id* - Identificador del trabajo asignado.
  - *prevhash* - *Hash* del bloque anterior.
  - *Coinbase*1 - Parte inicial de la transacción de *Coinbase*.

  - *Coinbase*2 - Parte final de la transacción de *Coinbase*.
  - *merkle_branch* - Lista de *hashes* de las transacciones del bloque (necesario para poder calcular la raiz del *Merkle Tree*, o *merkle_root*).
  - *version* - Versión del bloque de Bitcoin.
  - *nbits* - Dificultad de la red.
  - *ntime* - Un *timestamp*.
  - *clean_jobs* - Una bandera que indica al minero si debe descartar todo lo referente a otros *job_id* y trabajar únicamente en el actual.

- Al recibir el mensaje, M debe generar otro valor hexadecimal denominado *Extranonce*2, de forma que su largo respete el tamaño *Extranonce*2*_size* que le fue indicado durante la suscripción.

- M genera la transacción de *Coinbase* tal que *Coinbase* = *Coinbase*1 + *Extranonce*1 + *Extranonce*2 + *Coinbase*2 donde la suma indica concatenación.

- M comienza a trabajar en busqueda de *shares*. M puede calcular el cabezal del bloque (*block_header*) como *block_header* = *version* + *prevhash* + *merkle_root* + *ntime* + *nbits* + *Nonce* + *padding*
  donde *padding* = 00000080000000000000000000000000000000
  00000000000000000000000000000000000000000000000000
  80020000. El motivo de utilizar este *padding* corresponde al tamaño en el que SHA256, la función de *hash* de Bitcoin, procesa los mensajes.

- Cuando M encuentra un *Nonce* que cumple con la dificultad de P, envía la *share* al administrador. Los datos enviados son:

  - usuario.
  - *job_id*.
  - *Extranonce*2.
  - *ntime*.
  - *Nonce*.

- P le paga a M a la clave pública indicada en la suscripción. El pago no tiene por qué ser inmediato. Comúnmente la recompensa es otorgada luego de que uno de los mineros encuentra un FPoW, pero depende del sistema de pago implementado.

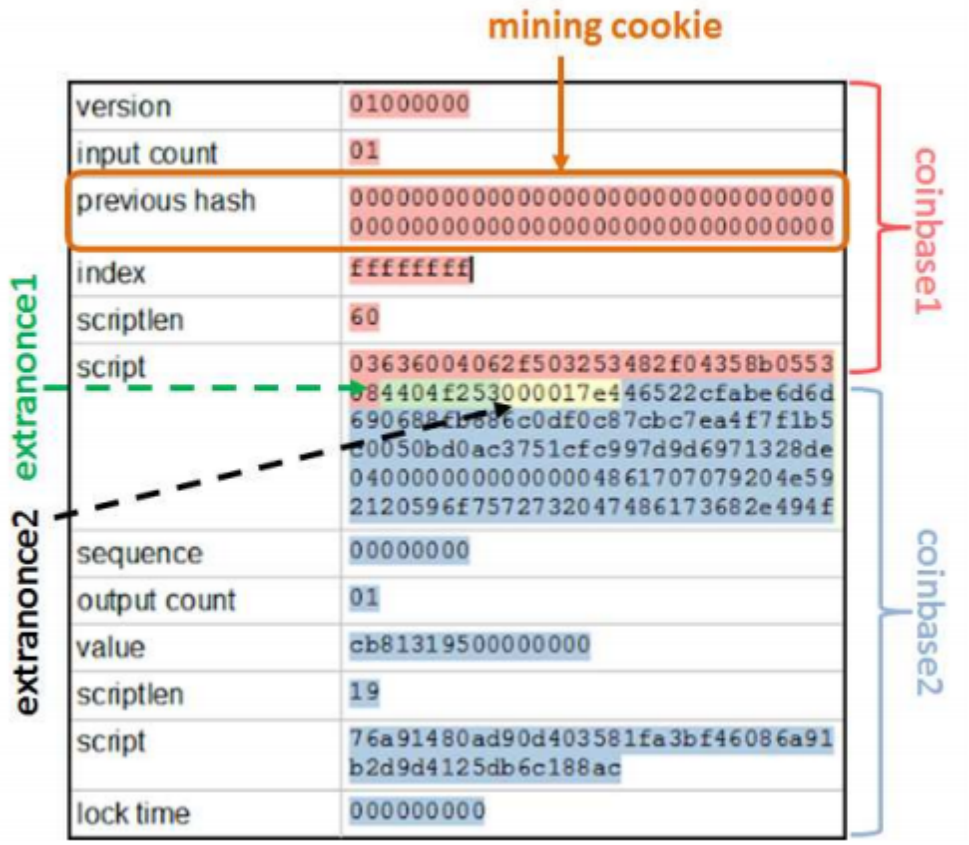

Figura 2.3: Estructura de la transacción de *Coinbase*, con el detalle del contenido de *Coinbase*1 y *Coinbase*2, junto con la ubicación de los *Extranonce*. (Figura 2 en [23]).

2.3.4 Sistemas de pago

Cada *pool* puede implementar diferentes formas de recompensar a los mineros por sus esfuerzos [19]. Se denomina ronda al tiempo entre que se encuentra un bloque y el siguiente.

Proporcional

Si el *pool* implementa un sistema de pago proporcional, en cada ronda el administrador paga a cada minero $\frac{n}{N}(1-f)B$ donde n es la cantidad de *shares* entregadas por ese minero, N el total de *shares* recibidas en la ronda, B es la ganancia del *pool* y fB es la comisión del administrador.

Pago por *share* (PPS)

En este caso, el administrador decide hacer a su *pool* más atractivo asumiendo el riesgo de la varianza de los mineros. Luego de recibir una *share*, y siendo p la probabilidad de que cada uno sea un bloque valido, el administrador paga $(1-f)pB$ al minero en forma adelantada, como valor estimado por la contribución de su *share*. El administrador eventualmente debe ajustar el valor de su comisión para no caer en bancarrota. Variaciones de este sistema se han tratado de desarrollar para reducir los riesgos sin perder el atractivo para los mineros. En general, consisten en mantener un balance en base a las *shares* entregadas por cada minero y garantizar una recompensa intermedia entre el pago proporcional y el definido por el balance.
Otros sistemas han sido definidos con el fin de contrarrestar ataques, algunos de los cuales se analizan en el capítulo 3.

2.3.5 Problemas de los *pools* de minería

Los *mining pools* pueden ser privados, pero también los hay públicos donde cualquier minero es capaz de unirse y solicitar trabajos al administrador.

Actualmente, los dos *pools* más populares ya sobrepasan el 30 % del poder de cómputo de Bitcoin (ver figura 2.4). Esta distribución ha sido incluso más desigual en el pasado, con *pools* sobrepasando el 25 % por si solos. En Junio del año 2014, un *pool* logró superar el 50 %, lo que rápidamente causó una reacción de la comunidad y el *pool* colapsó poco tiempo después [24].
Si bien la creación de los *pools* sin lugar a duda responde a los problemas presentados en la sección 2.2.5, la existencia de estos *pools* masivos pone en cuestionamiento nuevamente a la descentralización de Bitcoin. A pesar de contar con un gran número de mineros que pueden ser honestos y tienen en todo momento la opción de abandonar el *pool*, si el administrador fuera capaz de utilizar el poder de cómputo concentrado de sus mineros para otros fines, la estabilidad de la criptomoneda estaría en riesgo. La literatura especializada cuenta con varios estudios sobre ataques que pueden afectar tanto a miembros de *pools* como a la comunidad entera, y son algunos de estos ataques los que se analizan en el resto de este trabajo.

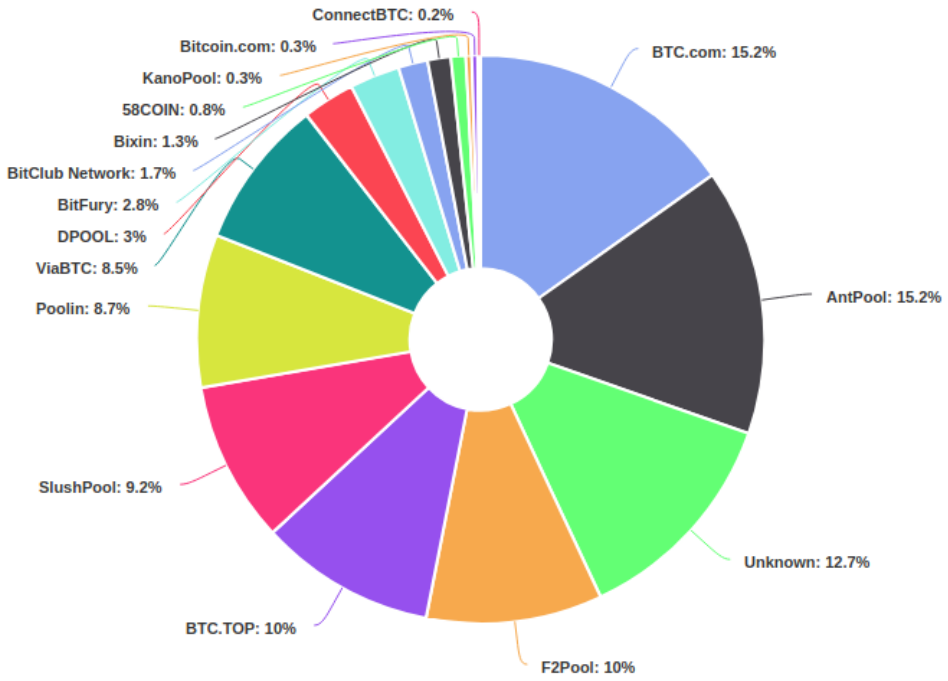

Figura 2.4: Distribución del poder de cómputo a la fecha 3 de octubre del 2018 [5].

Capítulo 3

ANÁLISIS Y EVALUACIÓN

En el capítulo 2 se presentaron los fundamentos de diseño y tecnológicos detrás de Bitcoin y de los *mining pools*. A su vez, se introdujo la motivación detrás de intentar generar una distribución uniforme del poder de cómputo a lo largo de la comunidad. Para que un sistema de este tipo sea de amplio uso y perdure en el tiempo, tiene que presentar fortalezas para resistir diversos tipos de ataques. En este capítulo se analizan en detalle algunos de los ataques presentados en la literatura especializada. En algunos de los trabajos encontrados, se relevan ataques a Bitcoin en forma genérica [25] [26]. Sin embargo, durante el relevamiento de este trabajo se pudo observar que existe un gran número de ataques donde el objetivo o la herramienta principal son los *mining pools*. Se considera que es relevante generar un espacio para clasificar y comprender en mayor profundidad este tipo de ataques. En la sección 3.1, se elaboran criterios para poder clasificarlos, seguidos de criterios para poder evaluar los riesgos que presentan para Bitcoin. Al final de la sección se seleccionan tres de los ataques encontrados, en base a su vigencia y a poder ilustrar mejor el uso de los criterios elaborados. Las tres secciones siguientes se dedican a profundizar en cada uno de estos ataques, y finalmente en la sección 3.5 se presentan a modo de resumen los resultados encontrados, buscando comparar los ataques en base a los criterios.

3.1 Criterios de clasificación y evaluación

3.1.1 Clasificación por objetivo

El primer criterio que se define es respecto al objetivo, no en el sentido de propósito, los cuales pueden ser múltiples aún dentro de un mismo ataque, sino respecto a la debilidad explotada en el ataque. Durante el relevamiento se encontró que todos los ataques caen en una o más de las siguientes tres categorías:

- Ataques al protocolo de los *pools*.
- Ataques al protocolo de consenso de Bitcoin.
- Ataques de red.

Ataques al protocolo de los *pools*

En esta categoría se encuentran todos los ataques que explotan falencias en los protocolos que permiten a los mineros de un *pool* y su administrador comunicarse y llevar a cabo sus tareas.

Ejemplos de estos ataques son los ataques a Stratum [23], el protocolo de comunicación más difundido entre los *mining pools*, donde debido a que la comunicación no es cifrada, pueden existir dos tipos diferentes de atacantes según sus capacidades. Estos son:

- Pasivos - El atacante tiene la capacidad de inspeccionar el tráfico entre el minero y el administrador, o bien tiene acceso a metadatos de las conexiones, como pueden ser los registros (*logs*) de un proveedor de servicios de internet (ISP) [27].
    - ***StraTap*** - En este ataque se asume que el atacante tiene la capacidad de escuchar todos los mensajes que se intercambian entre el minero y el administrador. Esto le permite contar las *shares* enviadas por el minero, el tiempo entre ellas, y ver la dificultad de las mismas para inferir su poder de cómputo. Esta información, junto con cierto conocimiento público (recompensa por bloque insertado, dificultad actual de Bitcoin, etc), permite estimar fácilmente las ganancias del minero.
    - **ISP*log*** - En este caso se asume que el atacante tiene acceso a los metadatos de las conexiones. El administrador envía la dificultad establecida por el *pool* inmediatamente luego de establecer la comunicación (*handshake*). A partir de este momento, el minero trabaja hasta haber enviado cincuenta *shares*. Al pasar las cincuenta *shares*, el administrador le asigna una nueva dificultad. Alcanza con ver cuanto tarda el minero en enviar esas cincuenta *shares* a una dificultad conocida para poder inferir su poder de cómputo y por lo tanto la ganancia que espera recibir.

- Activos - El atacante es capaz de violar la confidencialidad y la integridad de los paquetes enviados entre el minero y el administrador, incluso pudiendo personificar a cualquiera de las dos partes.

    - ***BiteCoin*** - En este ataque se secuestran los pagos de algunos mineros del *pool*. El atacante debe interferir en la comunicación entre el minero y el administrador, para que el minero trabaje en hallar las *shares*, pero el atacante sea quien cobre las recompensas.

Ataques al protocolo de consenso

Estos ataques buscan manipular el resultado del PoW. Un caso particular son ataques que busquen generar un doble gasto de una moneda, como fue visto en la sección 2.2.5. Sin embargo, también existen otros ataques que tan solo se busca obtener una mayor ganancia que la esperada dado el porcentaje de poder de cómputo, y/o reducir la ganancia de otros.

Ejemplos de estos ataques, que involucran a *mining pools*, son los siguientes:

- ***Block Withholding*** [28] - Este ataque tiene dos variantes, ***Sabotage*** y ***Lie in Wait***, pero en ambos casos consiste en minar en un *pool*, entregando PPoW al administrador, pero ocultando o descartando los FPoW encontrados.

- ***Pool-Hopping*** [28] - En lugar de invertir todo el poder de cómputo en un único *pool* con el fin de hallar un FPoW y generar ganancias al *pool*, este ataque propone ir alternando el poder de cómputo entre diferentes *pools*, aplicando criterios que le permitan maximizar la ganancia por las *shares* entregadas.

- ***Selfish Mining*** [29] - Consiste en ocultar bloques de forma estratégica para causar *forks* intencionales en la *blockchain*.

- ***Fork After Withholding*** [30] - Combina nociones de *Selfish Mining* con *Block Withholding* para incrementar las ganancias y las garantías de obtenerlas.

Ataques de red

Dado que las grandes criptomonedas como Bitcoin llevan a cabo sus actividades a través de una red P2P en internet, se encuentran expuestas a varios tipos de ataques a las capas que la componen. En particular, a las capas de transporte y de red. Existen varios ataques de partición para aislar nodos, o de retraso de paquetes que resultan efectivos contra las criptomonedas [31] [25]. Dentro de esta categoría existe un subconjunto de ataques que pueden usarse de forma específica contra los participantes de los *mining pools*, o como parte de otros ataques a ellos.

Ejemplos de estos ataques son:

- ***Eclipse*** [32] - El atacante logra aislar al nodo del participante del resto de la red P2P, particionando a la red. Existen estudios que muestran mejoras en al menos uno de los ataques al protocolo de consenso al combinarlo con este ataque. En el capítulo 5 se discute más al respecto.
- ***Sybil*** [33] - El atacante logra esparcir múltiples nodos a lo largo de la red P2P. En los ataques a *mining pools* que buscan manipular el protocolo de consenso puede incrementar la probabilidad de éxito.
- ***Session Hijacking*** [34] - Consiste en infiltrar una sesión de comunicación, por ejemplo una conexión TCP, y ser capaz de comunicarse con un nodo de la conexión aparentando ser el otro y viceversa. Es particularmente importante en ataques al protocolo de los *pools*, donde la falta de cifrado en la comunicación permite efectuar este tipo de ataques con éxito.

3.1.2 Clasificación por roles en el ataque

El segundo criterio que se define es para determinar quién perpetra el ataque, y quién resulta afectado por el mismo. Durante el relevamiento se pudo observar que dentro de todos los ataques que conciernen a los *mining pools*, hay tres posibilidades.

Una Entidad ataca a un *pool*

Uno o un conjunto de participantes se suscriben a un *pool*, pero no necesariamente con el fin de reducir la varianza de sus ganancias. En algunos casos, el atacante puede lograr una ganancia mayor a la que le corresponde por su poder de cómputo. En otros, puede estar buscando perjudicar a ciertos miembros del *pool* o al *pool* en su totalidad. Dentro de esta categoría se puede destacar a su vez que algunos ataques no le generan pérdidas al *pool*, afectando solo a algunos miembros del mismo.

De los ejemplos mencionados en la clasificación anterior, *StraTap*, ISP*log*, *BiteCoin*, *Block Withholding*, *Pool-Hopping* y *Fork After Withholding* pueden ser utilizados de esta manera.

Un *pool* ataca a la comunidad

En la sección 2.3.5 se pudo observar la existencia de *pools* masivos. Con tanto poder de cómputo, es esperable que un ataque perpetrado por uno de estos *pools* pueda generar impacto sobre la comunidad entera de Bitcoin. Estos ataques pueden simplemente ser usados para beneficiar al *pool* con una ganancia extra, pero si el *pool* cuenta con suficiente poder de cómputo, pueden incluso usarse para manipular el contenido de la *blockchain* de forma que logre desestabilizar a la criptomoneda.

De los ejemplos mencionados en la clasificación anterior, *Selfish Mining* pertenece a esta categoría.

Un *pool* ataca a otro *pool*

Por último, existen ataques en los que un *pool* ataca a un *pool* rival. El objetivo puede nuevamente ser el de generar mayores ganancias que las esperadas, o puede simplemente ser usado para eliminar a la competencia. En algunos de los ataques encontrados existe el escenario en que dos o más *pools* pueden llegar a atacarse simultáneamente. El estudio de estos casos deriva en el planteo de un problema similar al conocido dilema del prisionero [35], denominado en forma intuitiva como el dilema del minero [36].

De los ejemplos mencionados en la clasificación anterior, *Block Withholding* y *Fork After Withholding* pertenecen a esta categoría.

3.1.3 Criterios de Evaluación

Se desea que estos criterios permitan evaluar diferentes aspectos de los ataques en forma objetiva, y que a su vez permitan comparar a los ataques entre sí. Se buscan a su vez criterios que puedan ser aplicados a lo largo de varios ataques por medio de la literatura disponible. El conjunto de criterios definidos es el siguiente:

- **Efectividad** - Con este criterio se pretende medir la ganancia de efectuar el ataque. Como métrica se toma la ganancia extra relativa (*Relative Extra Reward* o RER). Es la ganancia que se obtiene de utilizar los recursos en efectuar el ataque respecto a solo minar honestamente.
  $RER = \frac{R_a - R_h}{R_h}$ donde R_a es la ganancia por atacar y R_h es la ganancia por minar honestamente.
- **Requerimientos** - Algunos ataques pueden dar mayores ganancias que otros, pero a su vez pueden requerir condiciones o recursos no alcanzables. Este criterio se enfoca en analizar qué requerimientos tiene el ataque en términos de poder de cómputo, memoria y acceso a la red.
- **Posibilidad de ser detectado** - En este criterio el foco se encuentra en cómo saber que un ataque ha ocurrido. ¿Existe evidencia de la ocurrencia del ataque? ¿Es posible detectar la ocurrencia de un ataque en particular? y ¿Es posible identificar al atacante?
- **Posibilidad de ser contrarrestado** - El último criterio se basa en analizar la existencia de medidas para prevenir o contrarrestar la ocurrencia de estos ataques. En caso que existan, ¿Es posible mitigarlos completamente? y ¿Qué costos tienen las medidas a tomarse?

3.1.4 Selección de ataques

Dentro de los ataques observados en la literatura especializada, se decidió profundizar en tres de ellos, que ya fueron presentados en forma breve en los

ejemplos de las secciones 3.1.1 y 3.1.2. En la selección se intentó abarcar la mayor cantidad de categorías posibles, y priorizar ataques que presenten estudios con suficiente profundidad para abarcar los criterios en detalle, así como ataques con características que los mantengan vigentes y con relevancia para la comunidad Bitcoin. Por último, se desea que los ataques presenten características que permitan continuar su estudio en otros contextos no abarcados en este trabajo, sobre los cuales se discute en el capítulo 5. Los tres ataques seleccionados, que son detallados en detalle en las secciones siguientes, son ***BiteCoin***, ***Selfish Mining*** y ***Fork After Withholding***.

3.2 *BiteCoin*

3.2.1 Clasificación

BiteCoin permite a una entidad atacar a un *pool* público. Si bien el ataque se encuentra dirigido a sus participantes y no afecta directamente las ganancias del *pool*, se aprovecha de debilidades en la seguridad del protocolo que implementa, y el impacto en su reputación puede afectar su competitividad con otros *pools*.

Además de ser un ataque al protocolo de los *pools* con un atacante de carácter activo tal como se define en la sección 3.1.1, *BiteCoin* presenta elementos de ataque de red y al protocolo de la capa de transporte de internet [37], para poder infiltrarse en la conexión entre los mineros y el administrador.

Objetivo	Roles
Protocolo de los *pools* y Ataque de red	Entidad → *Pool*

Tabla 3.1: Clasificación de *BiteCoin*.

3.2.2 ¿Cómo funciona?

Definición 3.2.1. Esquema de *BiteCoin*
Dado un atacante, un *pool*, y un minero que participa en dicho *pool*, el esquema del ataque es el siguiente:

- El atacante hace un *Session Hijacking* [34] de la conexión TCP entre el administrador del *pool* y el minero.
- El atacante realiza una conexión TCP con el *pool* y se suscribe para recibir trabajos.
- Cuando recibe el mensaje de asignación de trabajo por parte del administrador (a través de su propia conexión TCP), se lo retransmite al minero (para que realice el trabajo asignado al atacante)

- Cuando el atacante recibe una *share* del minero, modifica el parámetro del usuario, ingresando el suyo, previo a enviar la *share* al administrador por medio de su propia conexión TCP. De esta forma, el atacante es quien recibe la recompensa.

En la actualidad, el ataque *BiteCoin* tal como fue presentado no es posible de realizar. Esto se debe a que actualmente el *Extranonce*1 es enviado por el *pool* al minero como respuesta de su pedido de suscripción (previo al *Session Hijacking*). Por lo tanto, si se intenta ejecutar *BiteCoin* tal como está definido, no funciona, ya que el minero víctima va a trabajar con el *Extranonce*1 de su propia conexión, y no el del atacante.

Modificación a *BiteCoin*

Para que *BiteCoin* pueda ser exitoso bajo el funcionamiento actual de los *pools*, se debe hacer una modificación al esquema. Se debe establecer primero la conexión entre el atacante y el *pool*. Cuando la víctima intente suscribirse al *pool*, hacer el *Session Hijacking* de la conexión TCP, para **modificar la respuesta del *pool*,** sustituyendo el *Extranonce*1 por el asignado al atacante.

3.2.3 Evaluación

Efectividad

La aplicación de este criterio depende de cómo se interprete el ataque. En la descripción original del ataque, no se requiere que el atacante tome parte en la minería. Calculando el RER bajo estas condiciones, el resultado tiende a infinito ya que $R_a > 0$ y $R_h = 0$, entonces $RER = \frac{R_a - R_h}{R_h}_{R_h \to 0} = +\infty$.
Es posible asumir que el atacante es un minero con poder de cómputo α tal que $R_h = \alpha$, atacando a n mineros de un *pool*. A diferencia de los ataques de las secciones 3.3 y 3.4, en *BiteCoin*, efectuar el ataque no involucra el uso de poder de cómputo en término de *hashes* calculados. Esto hace que sea difícil estimar la cantidad de poder de cómputo perdido por cada minero atacado. Sin embargo, en la medida que n no sea demasiado grande, se puede considerar despreciable ya que los requisitos de *BiteCoin* en términos de cómputo son despreciables para los procesadores de la actualidad. Bajo estas hipótesis, $R_a = \alpha + \sum_{i=1}^{n} \delta_i$ donde δ_i es la ganancia del minero víctima i del *pool*. De esta forma,
$RER = \frac{\sum_{i=1}^{n} \delta_i}{\alpha}$.

Requerimientos

En cuanto a CPU/Memoria, este ataque requiere una cantidad despreciable, ya que solo necesita hacer un *Session Hijacking* y hacer simples modificaciones en unos pocos paquetes que intercepta, como se explica en la sección 3.1.1. Si se mira el uso de CPU desde el punto de vista del poder de cómputo para el cálculo de *hashes*, el atacante no necesita calcular ni un solo *hash* como se puede ver en la definición del ataque. En cuanto a Memoria RAM únicamente se necesitan mantener tres conexiones TCP el nombre del atacante y el código que se este ejecutando. Pueden existir variantes de este ataque que requieran guardar más información en memoria y utilizar más poder de procesamiento, pero no es el caso que se esta analizando aquí.

En cuanto a acceso a la red, es necesario lograr comprometer totalmente la conexión entre el minero y el *pool*, teniendo la capacidad de leer y modificar el tráfico entre ellos. Un insumo que es de utilidad para que se realice el ataque de forma más efectiva es la capacidad de leer los *logs* de algún ISP, para,

ayudándose de ello, se infieran las ganancias de los mineros y se seleccione a cuáles atacar.

Posibilidad de ser detectado

De acuerdo al planteo original del ataque [23], es posible detectar que se está siendo víctima del mismo al no recibir recompensas por el trabajo realizado. En la práctica, el minero podría no detectarlo si no conoce el ataque, ya que este podría asumir que por la varianza que existe aún no ha recibido recompensas. Es posible implementar en los clientes de minado una alerta al minero en caso de estar enviando *shares* y que al cabo de un tiempo dado no se haya recibido recompensa por ellas.

Posibilidad de ser contrarrestado

Según [23] es posible contrarrestar el ataque sin necesidad de cifrar toda la comunicación, ya que hacerlo implica un gran peso en el trabajo a realizar, tanto del *pool* como los mineros, lo que provoca que su implementación sea inviable.

Requerimientos de la solución [23]

La solución debe cumplir con un conjunto de características para satisfacer las necesidades de los usuarios. Estas son:

- La seguridad, que implica que no sea posible realizar ninguno de los ataques antes mencionados.
- La eficiencia, ya que la primera solución que se piensa es el cifrado, pero esta no es viable ya que no es eficiente para esta tarea (y de todas formas no soluciona el ataque de ISP *log*).
- La adoptabilidad, ya que debe introducir modificaciones mínimas a Stratum que permitan una fácil migración para los *pools* y para los mineros.

Solución

Las solución a esto es utilizar el protocolo *Bedrock* [23]. Una extensión de

Stratum creada para solucionar estos problemas y que cumple con los requisitos de seguridad, adaptación y eficiencia en términos de rendimiento (*performance*), disminuyendo lo menos posible la tasa de *hashes* del minero, y aumentando lo menos posible el tiempo que le toma al administrador del *pool* atender a un minero. A continuación se detallan las tres características principales que permiten que esto sea posible. A lo largo de las mismas, se asume la existencia de una clave K_m conocida únicamente por el minero m y el administrador(intercambiada por ejemplo mediante *Diffie y Hellman* [38]).

Mining cookies

Este punto soluciona el hecho de que un atacante pueda inferir la tasa de *hashes* del minero y apoderarse de su trabajo. La forma en que lo hace es utilizando una *cookie*, la cual es generada por el administrador en base a un valor aleatorio y el nombre de usuario del minero. La manera de utilizarla es haciendo que esta sea una campo secreto adicional en el PoW de Bitcoin.
La forma en que esta se comparte entre el administrador y el cliente es enviando un mensaje cifrado con el valor aleatorio para que el cliente pueda generar la *cookie*. Por lo tanto, cuando un atacante escuche la comunicación, no va a poder saber si las *shares* enviadas por el cliente son correctas ya que el atacante no cuenta con la *cookie*, y tampoco va a poder dirigir las recompensas a su nombre de usuario ya que la *share* es válida para la *cookie* que se genera con el nombre de usuario del cliente.
Para evitar que un atacante pueda, observando la *blockchain*, obtener la *cookie* de un minero, cada vez que un minero encuentra un bloque válido para insertar en la *blockchain*, se cambia su *Mining cookie*. Esto no es un problema para la eficiencia ya que es un evento poco frecuente.

Protección en la comunicación de secretos

Es necesario proteger datos sensibles en la comunicación, como el valor aleatorio para la *cookie*, o la dificultad de los trabajos, ya que si por ejemplo este último es conocido por el atacante, puede ser de utilidad para inferir la tasa de *hashes* del minero y por lo tanto su ganancia.
La forma en que se propone protegerlos es enviando la lista de secretos en un mensaje cifrado con K_m.

Seleccionar dificultad de forma segura

Para esto, el minero estima su propia tasa de *hashes*, y lo envía al administrador en el mensaje de suscripción, el cual se envía cifrado. El administrador puede

ajustar la dificultad enviándola cifrada al minero.

Performance

En [23] , se compara la utilización de la solución recién presentada, con *Blanket encryption* y con *TLS encryption decryption* con AES-256 en modo CBC.

En los resultados obtenidos en [23] bajo el supuesto de que ha realizado trabajo durante un día por un *pool* de dieciséis mil integrantes, se concluye que para cada minero, en cualquiera de los casos, el costo en tiempo adicional sería menor a 1s. Sin embargo, para el administrador, en los casos de cifrado de todos los mensajes el costo es de $1.36hs$, mientras que con la solución recién presentada es de $12.03s$.

Por lo tanto, entre las posibles soluciones consideradas, la presentada es la menos costosa.
Pese a esto, los *pools* no la implementan actualmente. La razón por la que se cree que sucede esto es porque los *pools* asumen que es responsabilidad de cada minero no ser víctima de un *Session Hijacking*, ya que de implementarse esta solución, por más que los costos de hacerlo sean bajos, no son nulos. Además, el único afectado es el minero que es atacado. Ni el administrador ni el resto del *pool* pierden ganancias.

3.3 *Selfish Mining*

3.3.1 Clasificación

Selfish Mining es un ataque en el que un *pool* ataca a toda la comunidad, desperdiciando el esfuerzo colectivo de sus mineros y perjudicando sus ganancias, lo que puede debilitar la seguridad de Bitcoin. En la sección 3.3.2, se observa como el *pool* atacante debe alcanzar un mínimo poder de cómputo si pretende que el ataque sea rentable.

Es un ataque al protocolo de consenso, que aprovecha las características del *Proof of Work* y en particular la noción de interpretar a la cadena más larga como cadena principal de la *blockchain*.

Objetivo	Roles
Protocolo de consenso	*Pool* → Comunidad

Tabla 3.2: Clasificación de *Selfish Mining*.

3.3.2 ¿Cómo funciona?

Definición 3.3.1. Esquema de *Selfish Mining*
Dado un atacante y una cadena principal de largo n, el esquema del ataque es el siguiente:

- El atacante comienza minando el bloque $n+1$ en forma honesta sobre la cadena principal.
- Si otro participante encuentra un bloque antes que él, continúa minando sobre dicho bloque en forma honesta, buscando el bloque $n+2$.
- Cuando el atacante encuentra un bloque, no lo publica, sino que continúa minando sobre su propia cadena privada, que en ese momento es igual a la cadena pública a menos del bloque adicional del atacante. A partir de este punto pueden ocurrir dos eventos:

- Otro participante encuentra el bloque $n+1$ de la cadena pública antes que el atacante encuentre el $n+2$ de la cadena privada. El atacante corre el riesgo de que su esfuerzo en el bloque $n+1$ se pierda, por lo que publica inmediatamente su bloque. Así comienza una **condición de carrera** por el bloque que pueda permanecer en la cadena principal.
- El atacante encuentra el bloque $n+2$ en su cadena privada antes que el resto de la red logre encontrar el $n+1$ en la cadena pública. El atacante extiende la cadena privada que ahora supera en largo a la cadena pública por dos bloques. Continúa minando en su cadena privada mientras que $Largo(cadena\_pública) < Largo(cadena\_privada)-1$. Cuando se alcance la igualdad (cuestión que es esperable si el atacante no posee la mayoría del poder de cómputo), el atacante publica todos los bloques desde el $n+1$. En este punto el atacante vuelve a empezar el ataque sobre la nueva cadena principal.

Cabe destacar que la cadena privada es en todo momento completamente válida y sería aceptada por la red en caso de que fuera publicada, por lo que el atacante podría reclamar la recompensa por el esfuerzo invertido en cualquier momento. Sin embargo, si el ataque tiene éxito, el resto de la red se encuentra gastando poder de cómputo en un eslabón anterior de la cadena, ya que no conocen la cadena privada, y el poder de procesamiento de toda la red será desperdiciado.

¿Cómo ganar la condición de carrera?

La carrera la gana la primera cadena para la que se encuentre el siguiente bloque válido. De esta forma pasa a ser la más larga. El protocolo de Bitcoin indica que siempre se debe minar sobre la cadena más larga, pero no obliga a los mineros a escoger por una opción en caso de tener dos cadenas válidas de igual largo. Debido a esta indiferencia planteada por el protocolo, la mayoría de los nodos simplemente toma la primera cadena válida que haya recibido y mina sobre ella, ignorando cualquier otra cadena válida del mismo largo.

El atacante puede intentar tomar provecho de esto. Mientras más conexiones tenga, y se encuentre en un lugar más privilegiado de la red, más probabilida-

des tiene de escuchar el bloque minado antes de que este sea ampliamente propagado. Además puede propagar con mayor velocidad su propio bloque por la red. Esto le puede dar una gran ventaja para ganar la carrera.

También influye el poder de cómputo de los nodos que deciden minar sobre la cadena del atacante, y más específicamente el poder de cómputo del propio atacante.

Resultados de la carrera

La carrera puede terminar con tres resultados distintos:

- El atacante mina un nuevo bloque. De esta forma su cadena es más larga y gana la recompensa de dos bloques.
- Otro participante mina un bloque sobre la cadena del atacante. En este caso esta cadena también pasa a ser la más larga y el atacante gana la recompensa por haber minado un bloque.
- Un participante mina un bloque sobre la otra cadena. En este caso el atacante no gana la recompensa del bloque válido que encontró legítimamente.

En los tres casos la siguiente acción a tomar es la misma: adoptar la cadena que gana la carrera y repetir el ataque desde el principio.

¿Dónde se encuentra la ganancia del ataque?

Las situaciones anteriores no presentan un panorama muy positivo para el ataque, que debe apostar a poder ganar la condición de carrera para posiblemente solo obtener la ganancia de un bloque, lo cual pudo obtener desde un principio sin realizar el ataque.

La verdadera ganancia del ataque se encuentra en el caso donde el atacante encuentra dos bloques válidos antes que el resto de la red encuentre uno.

En este caso el atacante tiene la ganancia asegurada, ya que para evitar entrar en la posibilidad de una condición de carrera puede publicar la cadena privada exactamente antes de que la pública tenga la posibilidad de igualar su largo.

Dado que el protocolo de Bitcoin lleva a los mineros honestos a abandonar una cadena luego de ser publicada otra válida con mayor largo, el atacante deja huérfanos a todos los bloques encontrados por mineros honestos, haciendo que estos pierdan las recompensas junto con el tiempo y cómputo invertidos para conseguirlas. Por otro lado, el atacante gana todas las recompensas por los bloques encontrados mientras el resto de la red desperdicia sus recursos.

Selfish Mining como cadena de Markov

Definición 3.3.2. Cadena de Markov Se define como *cadena de Markov a tiempo discreto* a una secuencia de variables aleatorias $X_1, X_2, \ldots$ que toma valores en un conjunto enumerable donde X_i es el estado del proceso en el tiempo i, además se satisface la siguiente propiedad:

$$\begin{aligned} &\forall\, n \in \mathbb{N}, P(X_{n+1} = x_{n+1} | X_n = x_n, X_{n-1} = x_{n-1}, \ldots, X_1 = x_1) \\ &\quad = P(X_{n+1} = x_{n+1} | X_n = x_n) \end{aligned}$$

donde x_i es el estado del proceso en el momento i.

Es decir, el estado del proceso en un futuro depende solamente del estado actual del proceso.

El algoritmo de *Selfish Mining* también puede ser visto como una cadena de Markov gracias a esto. Es posible debido a que se considera que encontrar un bloque puede ser modelado por una variable aleatoria con distribución exponencial, es decir, sin memoria. Esto significa que aunque el resto de la red lleve más tiempo minando un bloque que el atacante intentando minar un segundo bloque, las probabilidades que cada uno tiene no se ven afectadas. Dicho en términos de una cadena de Markov, las probabilidades de moverse de un estado (lo cual sucede al encontrar un bloque) no dependen de los estados anteriores, sino solamente del estado actual.

¿Por qué modelar *Selfish Mining* como cadena de Markov? Resulta interesante realizar este modelado para dar una base al análisis matemático de la eficiencia del ataque, con el objetivo de poder ser comparado de una forma más objetiva. Por otro lado se provee de una idea más gráfica del funcionamiento del ataque.

A continuación se presenta el algoritmo en términos de la cadena de Markov (figura 3.1). Los estados representan la actitud que debe tomar el atacante

frente a cada situación para realizar correctamente *Selfish Mining*. Por otro lado, las transiciones implican que algún minero ha encontrado un bloque, y finalmente la probabilidad de cada transacción es la probabilidad que tiene dicho minero de encontrar un bloque.

Dado un atacante con poder de cómputo α, la cadena cuenta con los siguientes estados y transiciones:

- S_0: En este estado el atacante se comporta como un nodo normal y minará sobre la *blockchain* pública con la esperanza de encontrar un bloque antes que los demás.
    - Con probabilidad $1 - \alpha$ la red encuentra un bloque antes que el atacante, por lo que este debe volver al estado S_0 (es decir, aceptar el bloque de la red y empezar a minar sobre la *blockchain* como cualquier minero honesto).
    - Con probabilidad α el atacante encuentra un bloque antes que el resto de la red, moviéndose al estado S_1 y pasando a minar en una cadena privada.
- S_1: El atacante empieza a minar en su cadena privada, la cual es un bloque más largo que la cadena pública.
    - Con probabilidad α el atacante encuentra otro bloque antes que el resto de la red, por lo cual debe añadirlo a su cadena privada y avanzar al estado S_2.
    - Con probabilidad $1 - \alpha$ la red encuentra un bloque antes que el atacante y lo publica, haciendo que se iguale el largo de la cadena pública al de la privada del atacante, y poniendo en peligro de pérdida las ganancias del bloque minado por el atacante. Este debe pasar inmediatamente al estado S_{0b}.
- S_{0b}: El atacante debe publicar su bloque minado. En este momento se genera un *fork* el cual la red debe resolver. Algunos mineros de la red se encuentran trabajando sobre la cadena del atacante y otros sobre la originalmente pública. Se define como γ a la fracción del poder de cómputo combinado de los mineros que trabajan sobre la cadena del atacante. Naturalmente el atacante decide minar sobre su cadena, aunque su poder de cómputo (α) no se incluye en γ para diferenciar

correctamente los posibles casos. En todos los casos el siguiente estado es S_0.

- Con probabilidad α, el atacante encuentra un bloque y lo publica, debido a que su cadena resulta ser la más larga. La red acepta esta como la cadena válida y el atacante obtiene una ganancia correspondiente a dos bloques.
- Con probabilidad $(1-\alpha)\gamma$, un minero de la red encuentra un bloque sobre la cadena del atacante, haciendo que este gane la recompensa de un bloque.
- Con probabilidad $(1-\alpha)(1-\gamma)$, un minero encuentra un bloque sobre la cadena originalmente pública, haciendo que el atacante no tenga ganancia y además pierda el esfuerzo invertido en encontrar un bloque válido.

- S_N ($N > 1$, $N \in \mathbb{N}$): El atacante se encuentra minando sobre su cadena privada, la cual es de largo N bloques mayor que la cadena pública.
  - Con probabilidad α, el atacante encuentra un bloque y pasa al estado S_{N+1}.
  - Con probabilidad $1-\alpha$ y $N > 2$, la red encuentra un bloque y el atacante pasa al estado S_{N-1}.
  - Con probabilidad $1-\alpha$ y $N = 2$, la red encuentra un bloque y el atacante publica su cadena privada. Debido a que la cadena privada del atacante sigue siendo un bloque más larga que la cadena pública, los mineros toman la cadena del atacante como la válida, dándole ganancia por la cantidad de bloques encontrados hasta el momento. Se pasa al estado S_0 y se repite el proceso de *Selfish Mining*.

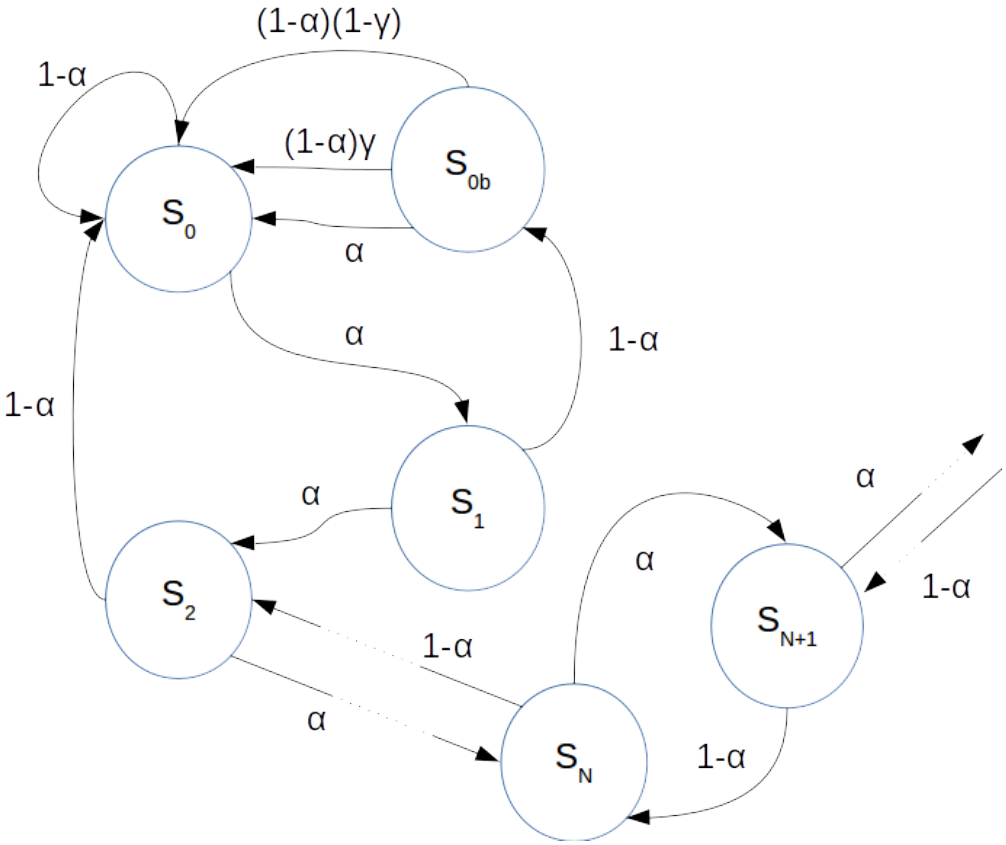

Figura 3.1: Cadena de Markov que representa el *Selfish Mining*.

3.3.3 Evaluación

Efectividad

El análisis de las ganancias de *Selfish Mining* en Bitcoin puede ser encontrado en mayor profundidad en [39] y [29], ambos con resultados equivalentes.

Previo a profundizar en el análisis, se deben se deben realizar ciertas restricciones para simplificar el estudio:

- Si la red cuenta con n mineros, y α_i es el poder de cómputo del minero i, entonces $\sum_i^n \alpha_i = 1$.
- No hay otros ataques que ocurriendo en simultáneo.
- La ganancia de un bloque se toma como una especulación probabilística en cada ronda, al normalizarla al valor de $1BTC$.
- Dado que la tasa de bloques huérfanos es baja, tal como se indica en [40], se considera que no hay *forks* no intencionales, y por lo tanto, la recompensa de un minero es igual a la probabilidad que tiene de encontrar un bloque en cada ronda.

Para el cálculo de las ganancias se utilizará *Selfish Mining* como cadena de Markov. Más específicamente, se calcula la distribución estática de la cadena, es decir, la probabilidad de pertenecer a cada estado. Esto significa obtener el vector

$$\pi = (p_0, p_{0b}, p_1, p_2, \ldots) / \sum_{i=0}^{+\infty} p_i + p_{0b} = 1$$

Definición 3.3.3. Matriz de transición Se define como matriz de transición a la matriz T formada por las probabilidades de transición $T_{ij} = P(X_{n+1} = j | X_n = i)$

En este caso, la matriz es la siguiente:

$$T = \begin{bmatrix} 1-\alpha & 0 & \alpha & 0 & 0 & 0 & 0 & \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & \\ 0 & 1-\alpha & 0 & \alpha & 0 & 0 & 0 & \dots \\ 1-\alpha & 0 & 0 & 0 & \alpha & 0 & 0 & \\ 0 & 0 & 0 & 1-\alpha & 0 & \alpha & 0 & \\ 0 & 0 & 0 & 0 & 1-\alpha & 0 & \alpha & \\ & & & \vdots & & & & \ddots \end{bmatrix}$$

Todos los estados S_N a partir de S_2 tienen el mismo comportamiento, por lo que todas las siguientes filas de T repiten el mismo patrón

Se busca resolver la siguiente ecuación:

$$\pi * T = \pi$$

Lo que se traduce en resolver el siguiente sistema de ecuaciones:

$$\begin{aligned} p_{Ob} &= (1-\alpha)p_1 \\ p_0 &= (1-\alpha)p_0 + p_{0b} + (1-\alpha)p_2 \\ &\Rightarrow \alpha p_0 = (1-\alpha)p_1 + (1-\alpha)p_2 \\ p_1 &= \alpha * p_0 \\ &\Rightarrow \alpha p_1 = (1-\alpha)p_2 \\ p_2 &= \alpha p_1 + (1-\alpha)p_3 \\ &\Rightarrow \alpha p_2 = (1-\alpha)p_3 \end{aligned}$$

Resultando en:

$$\begin{aligned} \alpha p_0 &= (1-\alpha)p_1 + (1-\alpha)p_2 \\ p_{Ob} &= (1-\alpha)p_1 \\ \alpha p_1 &= (1-\alpha)p_2 \\ \forall_{n>2} \alpha p_n &= (1-\alpha)p_{n+1} \\ & \sum_{i=0}^{+\infty} p_i + p_{0b} = 1 \end{aligned}$$

Por otro lado, para el cálculo de las ganancias se toma a la cadena de Markov y se tiene en cuenta los caminos mediante los cuales el atacante obtiene

ganancia. Esta es la suma de cada camino, y cada camino es el producto entre la probabilidad de estar en un estado, la probabilidad de realizar el cambio de estado que dispara la ganancia, y la propia ganancia.

Casos en los que el *pool* obtiene ganancias:

- La cadena del atacante y la originalmente pública tienen largo uno (S_{0b}) y el *pool* encuentra un nuevo bloque (transición α). Se obtiene una ganancia de dos.
- La cadena del atacante y la originalmente pública tienen igual largo (S_{0b}) y un minero encuentra un bloque sobre la cadena del atacante (transición $\gamma * (1-\alpha)$). Se obtiene ganancia de uno.
- La cadena privada es dos bloques de largo mayor que la pública (S_2) y alguien encuentra un bloque (transición $(1-\alpha)$). Se obtiene una ganancia de dos.
- La cadena privada es de largo n mayor a dos (S_n) y un minero encuentra un bloque (transición $(1-\alpha)$). La ventaja del atacante se acorta, y cuando publique su cadena ganará una recompensa extra por el bloque minado. Se obtiene una ganancia de uno.

$$r_{pool} = (2\alpha)\,p_{0b} + (1-\alpha)\,p_{0b} + 2(1-\alpha)\,p_2 + \sum_{n=0}^{+\infty}(1-\alpha)\,p_n$$

Casos en los que el resto obtiene ganancias:

- No hay cadena privada (S_0), y el resto encuentra un nuevo bloque (transición $(1-\alpha)$). Se obtiene una ganancia de uno.
- La cadena del atacante y la originalmente pública tienen igual largo (S_{0b}), y se encuentra un nuevo bloque sobre esta última (transición $(1-\gamma)(1-\alpha)$). Se obtiene una ganancia de dos.
- La cadena del atacante y la originalmente pública tienen igual largo (S_{0b}), y los mineros encuentran un nuevo bloque sobre la cadena del atacante (transición $\gamma(1-\alpha)$). Se obtiene una ganancia de uno.

$$r_{resto} = (1-\alpha)\,p_0 + 2(1-\gamma)(1-\alpha)\,p_{0b} + \gamma(1-\alpha)\,p_{0b}$$

Teniendo en cuenta que parte del poder computacional se desperdicia minando sobre una cadena que finalmente no es la principal (ya sea el atacante que pierde la condición de carrera o los mineros que sufren del ataque) las recompensas r_{pool} y r_{resto} son en total menor a 1. No se obtiene un bloque cada diez minutos, sino que el flujo de bloques se vuelve más lento, pero dado que el protocolo de Bitcoin ajusta la dificultad para oponerse al enlentecimiento, la ganancia del *pool* también es ajustada.

$$R_{pool} = \frac{r_{pool}}{r_{pool} + r_{resto}}$$

Resolviendo el sistema de ecuaciones resultante de la distribución estática y suplantando en R_{pool} se obtiene el siguiente resultado:

$$R_{pool} = \frac{\alpha(1-\alpha)^2(4\alpha + \gamma(1-2\alpha)) - \alpha^3}{1 - \alpha(1 + (2-\alpha)\alpha)}$$

Para que el ataque sea rentable, R_{pool} debe ser mayor a α (ganancia al minar honestamente), ya que de lo contrario, al minar honestamente se obtendrían más recompensas. De esta forma se obtiene la relación que se debe cumplir para tener ganancia:

$$\frac{(1-\gamma)}{(3-2\gamma)} < \alpha$$

Se deduce que $\alpha < \frac{1}{2}$, ya que en caso de tener la mayoría del poder de cómputo no es necesario realizar este tipo de ataque para controlar el contenido de la *blockchain*. La gráfica de la relación que se debe cumplir para tener ganancia puede apreciarse en la figura. La misma permite a su vez apreciar la utilidad del ataque, ya que Bitcoin resulta inseguro sin necesidad de alcanzar el 50 % del poder de cómputo global. Con $\frac{1}{3} < \alpha < \frac{1}{2}$ se puede hacer uso de este ataque con $\gamma = 0$, y mientras más alto sea γ menos poder de cómputo es necesario para que el ataque sea rentable. (Figura 3.2)

En la práctica no es tan sencillo. *Selfish Mining* es detectable, especialmente por los mineros que forman parte del *pool* atacante. Los mineros en general no tienen interés en realizar un ataque a Bitcoin ya que eso haría que la moneda pierda confianza y subsecuentemente pierda valor, por lo que el *pool* atacante con mayores posibilidades de éxito es un *pool* privado en donde todos estén dispuestos a atacar. En [41] se señala que este tipo de *pools* se encuentra limitado a un máximo de $\gamma = 0.89$, lo que según la figura 3.2 dicta que el

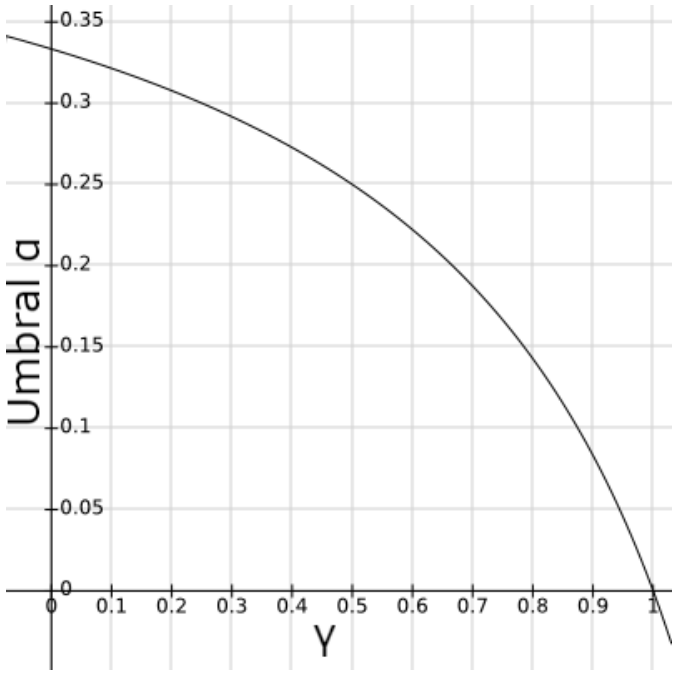

Figura 3.2: α mínimo requerido para superar en ganancias al minero honesto para un γ dado.

atacante requiere un $\alpha = 0.09$ como mínimo para superar las ganancias que de minar honestamente. Dicho α es actualmente un poder computacional muy alto para un minero en solitario o un *pool* cerrado, incluso tomando el γ más alto posible.

Utilizando la ecuación obtenida para la ganancia del *pool* se puede calcular la RER para distintos valores de γ y α (Figura 3.3) , y observar que efectivamente para poderes de cómputo menores el resultado esperado es de pérdida.

Requerimientos

Selfish Mining se basa fuertemente en dos variables que determinan la efectividad del ataque: la proporción del poder de cómputo global del atacante (α) y el poder de cómputo de los mineros a los cuales el atacante es capaz de hacer llegar sus bloques durante la condición de carrera (γ).

Esto implica que el ataque tiene altos costos en poder de cómputo, lo cual es natural para un ataque que tiene como base el minado de bloques. Este costo puede resultar difícil de sostener, pero de todas forma el ataque muestra que Bitcoin presenta vulnerabilidades sin necesidad de tener un poder de cómputo

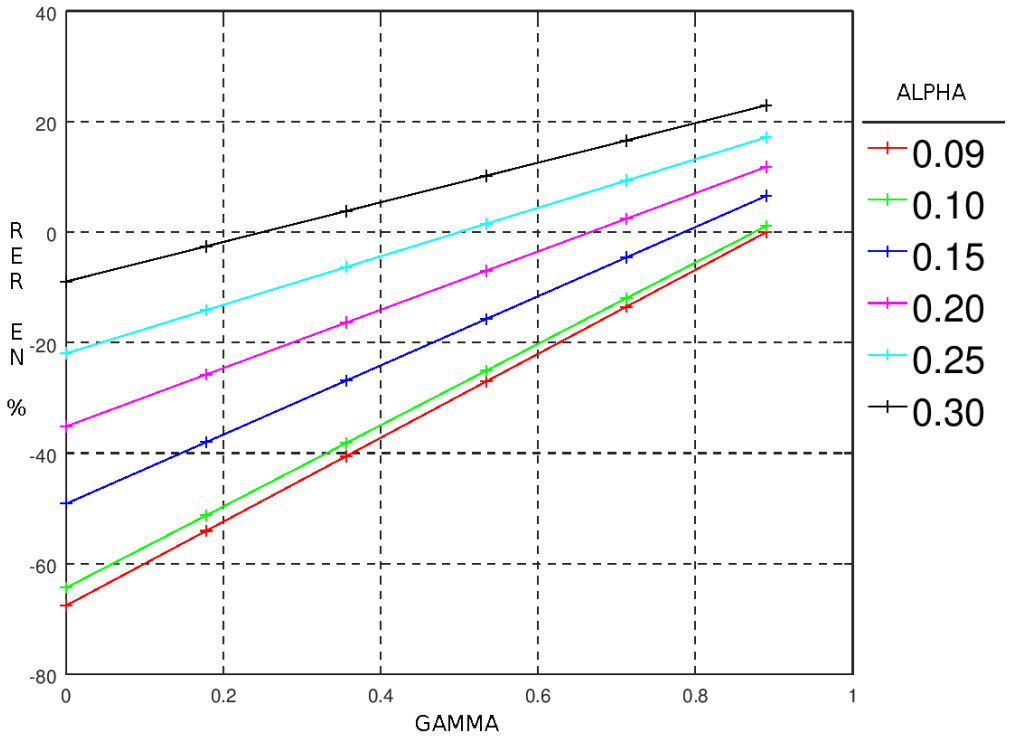

Figura 3.3: Ganancias en función de γ para atacante con diferentes poderes de cómputo.

mayor a la mitad, desafiando el concepto que originalmente se plantea acerca de su seguridad.

El otro costo compete a la red. Si el atacante se enfrenta ante la condición de carrera, se requiere una posición realmente privilegiada en la red para poder ejecutar el ataque, haciendo que sus bloques proliferen sobre los bloques publicados por otros mineros, más aún cuando los otros bloques son publicados primero y la publicación del atacante es reactiva. Este costo también implica una gran dificultad. Lograr una posición tan privilegiada tanto en la red física como en la red de Bitcoin no es algo que se pueda cotizar monetariamente como el poder de cómputo, por lo que este ataque no puede ser realizado por cualquier *pool*.

Posibilidad de ser detectado

El ataque puede ser detectado al mismo tiempo que el atacante puede ser identificado como responsable de éste.

Un minero honesto trabajando para el *pool* atacante puede identificar con facilidad este ataque simplemente escuchando la red. Si al momento en que el minero presenta un FPoW al *pool* no ve que dicho bloque sea publicado, significa que el *pool* muy probablemente está realizando *Selfish Mining* o un ataque similar, ya que se sigue minando sobre una cadena no pública.

Si el *pool* es público, mineros externos al *pool* atacante también pueden identificar el ataque (o por lo menos intuirlo), ya que la información de la *blockchain* es pública para todos y se sabe cuándo un *pool* público presenta un bloque. Los mineros pueden observar en distintos servicios de uso abierto qué bloques son publicados por cuáles *pools* públicos, lo que significa que se puede ver al atacante publicando en un corto período de tiempo bloques consecutivos. Este es un comportamiento anormal y debe generar sospechas. Luego de que el *pool* sea sospechoso de realizar *Selfish Mining* solo se debe observar su comportamiento para confirmarlo.

Posibilidad de ser contrarrestado

En la literatura se han investigado posibles medidas para contrarrestarlo o inhibirlo.

En el punto 6 de [39] se propone generar una cota para γ. El estudio indica que si γ fuera cercano a 1 el atacante no necesita de un gran poder computacional para realizar *Selfish Mining*. A su vez, si el *pool* es capaz de combinar el ataque con un *Sybil-Attack* que coloque nodos por la red P2P para que solamente propaguen los bloques del *pool* e ignoren los demás, es posible acercar el γ a dicho valor. La solución que se plantea no requiere un *fork* en el protocolo y puede ser adoptada gradualmente por los nodos. Mientras más nodos la adopten mejor es el efecto de la medida. Los nodos que reciben dos cadenas válidas de igual largo deben propagar ambas, y cuando se recibe más de una cadena válida el nodo debe seleccionar una aleatoriamente para minar sobre ella. El funcionamiento actual de la propagación de bloques deja que la topología de la red y las latencias decidan el γ, mientras que haciendo la elección explícitamente aleatoria, y en caso de que todos los nodos acepten esta estrategia, se limitaría a $\gamma = \frac{1}{2}$, lo que implicaría un umbral mínimo de ($\alpha = \frac{1}{4}$) para que el ataque sea rentable.

El punto 6 de [29] se enfoca en otro aspecto necesario para que el ataque sea rentable. Este aspecto es más discreto: el ajuste de la dificultad. *Selfish Mining*

se beneficia del reajuste automático de la dificultad, haciendo que esta baje y se obtengan más ganancias por unidad de tiempo aunque se desperdicie poder de cómputo minando en cadenas que no finalmente no quedan en la cadena principal. En caso de que la dificultad no disminuya ante la perpetración de éste ataque, se tendría la misma proporción de ganancias globales pero en cantidad neta la ganancia sería menor, lo cual haría que el ataque no sea redituable en comparación a minar honestamente, donde todo el poder de cómputo es usado en la cadena principal. El cálculo de la dificultad busca mantener el tiempo de generación de bloques en diez minutos, y se basa en la cantidad de bloques encontrados, pero solo tienen en cuenta los bloques de la cadena principal. Dado que *Selfish Mining* resulta en la generación de bloques huérfanos, disminuye el ritmo de generación de bloques en la cadena principal y obliga a la dificultad a decrementarse acordemente. La solución planteada consiste en incluir a los bloques huérfanos en el reajuste de la dificultad, y buscar la forma de incentivar a los mineros a incluir un listado de "Tíos"(bloques huérfanos al mismo nivel del bloque predecesor), por ejemplo tomando como regla el propagar por la red los bloques que cuenten con mayor cantidad de Tíos listados. De todas formas, como se ve en el capítulo 5, se debe tener cuidado con estos incentivos ya que un estudio reciente muestra que otra criptomoneda llamada Ethereum ha incorporado recompensas de este estilo que favorecen al *Selfish Mining*.

3.4 *Fork After Withholding*

3.4.1 Clasificación

Fork After Withholding (o FAW), es un ataque en el que una entidad puede atacar a un *pool*, afectando sus ganancias y obteniendo más ganancias en el proceso, que las que se obtienen minando en forma honesta. A su vez, puede ser usado para ataques entre *pools*. Lo que hace particularmente interesante a este ataque presentado por el instituto de ciencia y tecnología avanzada de Korea (KAIST) [30], es que a diferencia de *Selfish Mining*, si se ejecuta en forma adecuada, la decisión de atacar siempre es favorable, sin importar el poder de cómputo.

Es un ataque al consenso, ya que intenta manipular el contenido de la *block-chain*, pero es a su vez un ataque al protocolo de los *pools*, y en particular al protocolo para medir el esfuerzo realizado por sus mineros y las ganancias que les corresponden.

Objetivo	Roles
Protocolo de consenso y Protocolo de los *pools*	Entidad $\rightarrow$ *Pool* y *Pool* $\leftrightarrow$ *Pool*

Tabla 3.3: Clasificación de *Fork After Withholding*.

3.4.2 ¿Cómo Funciona?

FAW es una combinación de *Selfish Mining* con otro ataque denominado *Block Withholding* [28] (o BWH). Tener una noción del funcionamiento de BWH es importante para comprender las mejoras que ofrece FAW.

Definición 3.4.1. Esquema del BWH
Dado un atacante con poder de cómputo α y un *pool* objetivo P, el esquema del ataque es el siguiente:

- Se define un valor $\tau \leq 1$, de forma que $\tau\alpha$ sea una porción del poder de cómputo del atacante, destinada a minar en P.

- El atacante envía al administrador de P todos los PPoW que encuentre.
- Si encuentra un FPoW minando por fuera de P (minando honestamente), lo publica inmediatamente.
- Si encuentra un FPoW minando para P, lo descarta.

El administrador debe repartir la ganancia obtenida por los demás mineros honestos del *pool* entre todos los mineros, incluido el atacante, que no tiene intenciones de aportar ganancias al *pool*. ¿Qué obtiene el atacante? Un objetivo puede simplemente ser querer perjudicar al *pool*. Sin embargo, al entregar los PPoW, hay estudios que demuestran que dividiendo adecuadamente su poder de cómputo, un atacante con a lo sumo 20 % del poder de cómputo total, puede obtener más ganancias que un minero honesto [28].

En BWH también se plantea el escenario en que dos *pools* se atacan simultáneamente. Si para uno de los *pools* siempre es de mayor beneficio efectuar el ataque sobre el otro respecto a no atacar, es de esperar que se observen ocurrencias de este tipo de ataque en la práctica. Entre dos *pools*, la decisión de atacarse o no, es conocida como el dilema del minero [36], en referencia al problema de teoría de juegos conocido como el dilema del prisionero [35]. La similaridad entre ambos dilemas se encuentra en que no atacar no es un equilibrio de Nash[36]. Es decir, cualquiera de los *pools* puede mejorar sus ingresos si decide atacar. Sin embargo, si ambos *pools* atacan (situación de equilibrio), ambos ganan menos que si ninguno lo hace. Esta es la suposición que se realiza respecto a por qué no se observan ataques de BWH en forma constante.

FAW, por otra parte, garantiza en el peor caso una ganancia igual a la obtenida en BWH, y a su vez resuelve el dilema del minero, haciendo que bajo ciertas condiciones, la opción de atacar pueda ser siempre favorable para el *pool* de mayor tamaño.

Definición 3.4.2. Esquema del FAW
Dado un atacante con poder de cómputo α y un *pool* objetivo P, el esquema del ataque es el siguiente:

- Se define un valor $\tau \leq 1$, de forma que $\tau\alpha$ sea una porción del poder de cómputo del atacante destinada a minar en P.

- El atacante envía al administrador del *pool* todos los PPoW que encuentre.
- Si encuentra un FPoW minando honestamente, lo publica inmediatamente.
- Si encuentra un FPoW minando en P, a diferencia de BWH, lo guarda. A partir de este punto pueden ocurrir tres casos:
    - Un minero de P encuentra un FPoW. El atacante descarta el suyo y cobra su parte por las PPoW que ha entregado hasta el momento.
    - El atacante encuentra un FPoW minando honestamente. Propaga el bloque correspondiente en forma inmediata y descarta el FPoW de P.
    - El atacante descubre que un minero no perteneciente a P ha propagado un bloque. Entrega inmediatamente el FPoW guardado al administrador de P con el fin de generar un *fork* en la *blockchain*, y si su bloque logra ganar la condición de carrera, cobrar su parte de la recompensa.

3.4.3 Evaluación

Efectividad

El análisis de las ganancias de FAW llevado a cabo por el KAIST presenta resultados interesantes, y permite, conociendo ciertas características de la red, determinar la RER. Previo a profundizar en el análisis, se deben se deben realizar la mismas restricciones planteadas en la sección 3.3.3, además de restricciones adicionales:

- El administrador del *pool* siempre se comporta en forma esperada, propagando inmediatamente los bloques que recibe de sus mineros.
- El *pool* implementa un sistema de pago proporcional.

Al igual que en los estudios de *Selfish Mining*, dado que el periodo para encontrar un bloque puede ser modelado con una variable aleatoria con distribución exponencial de media igual al poder de cómputo del minero, la probabilidad de que un minero encuentre un bloque en una ronda es igual a su poder computacional relativo.

Teorema 3.4.1. Dado un atacante con poder de cómputo α, que utiliza una fracción τ para atacar a un *pool* con poder de cómputo β (que no incluye a $\tau\alpha$), la recompensa total que el atacante puede obtener es

$$R_a = \frac{(1-\tau)\alpha}{1-\tau\alpha} + \left(\frac{\beta}{1-\tau\alpha} + c.\tau\alpha.\frac{1-\alpha-\beta}{1-\tau\alpha}\right).\frac{\tau\alpha}{\beta+\tau\alpha}$$

donde el coeficiente c es la probabilidad de que un FPoW presentado al administrador gane la condición de carrera.

A continuación se analiza el significado de cada término:

- $\frac{(1-\tau)\alpha}{1-\tau\alpha}$ - El término $(1-\tau)\alpha$ es la porción de poder de cómputo del atacante destinada a minar honestamente. $1-\tau\alpha$ es el poder de cómputo honesto total de la red. Por lo tanto, este es el poder de cómputo honesto relativo del atacante. Dado que la ganancia es, bajo las restricciones

planteadas, equivalente al poder de cómputo relativo, esta es la ganancia proveniente del poder de cómputo usado por el atacante para minar en forma honesta.

- $\frac{\beta}{1-\tau\alpha}.\frac{\tau\alpha}{\beta+\tau\alpha}$ - El término $\frac{\beta}{1-\tau\alpha}$ corresponde análogamente a la ganancia que obtiene el *pool* por sus participantes honestos. A ello se lo multiplica por el porcentaje de poder de cómputo que el atacante posee en el *pool*, para obtener su parte de la ganancia.

- $(c.\tau\alpha.\frac{1-\alpha-\beta}{1-\tau\alpha}).\frac{\tau\alpha}{\beta+\tau\alpha}$ - Por último, este término corresponde al caso en que el atacante genera un *fork* por medio del FPoW minado en el *pool*. Es la probabilidad de que un minero externo al *pool* encuentre un bloque luego de que el atacante ha guardado un FPoW minando en el *pool*. Posteriormente es ponderado por el coeficiente c que depende de la topología de la red. Esta es la ganancia que el *pool* obtiene por medio del atacante. Nuevamente, se debe multiplicar por la porción que le corresponde al atacante bajo el sistema de pago planteado.

Con las ganancias de atacar y de minar honestamente, es posible calcular la recompensa relativa que se obtiene de efectuar el ataque. Si el atacante cuenta con los parámetros β y c a su disposición, puede calcular el τ óptimo derivando R_a. Se decide considerar en primera instancia que no se tiene acceso a estos datos y ver qué tan favorable puede resultar el ataque para distintos valores. El artículo del KAIST presenta resultados de simulaciones que avalan los resultados teóricos, acompañados de gráficos donde se varía c y β, asumiendo que el atacante puede calcular el τ óptimo. En este trabajo se opta por quitar este requerimiento y presentar los resultados para distintas combinaciones de τ, c, α y β.

Tal como se concluye en el artículo, Siempre es posible obtener ganancias del ataque. Para *pools* con un poder de cómputo cercano al 20 % (como son los más populares en la actualidad), se observa (Figura 3.4) que teniendo un poder de cómputo del 1 %, el ataque puede mejorar la ganancia entre un 0.05 % y 0.25 %, de acuerdo a qué tan probable es que el *fork* del atacante sea elegido.

Para un atacante del tamaño de un *pool* pequeño o de dimensiones similares a las del *pool* objetivo, el ataque mejora la ganancia entre un 0.53 % y 2.95 % (Figura 3.5). Para un *pool* como SlushPool (0.98 %) que afirma ganar en prome-

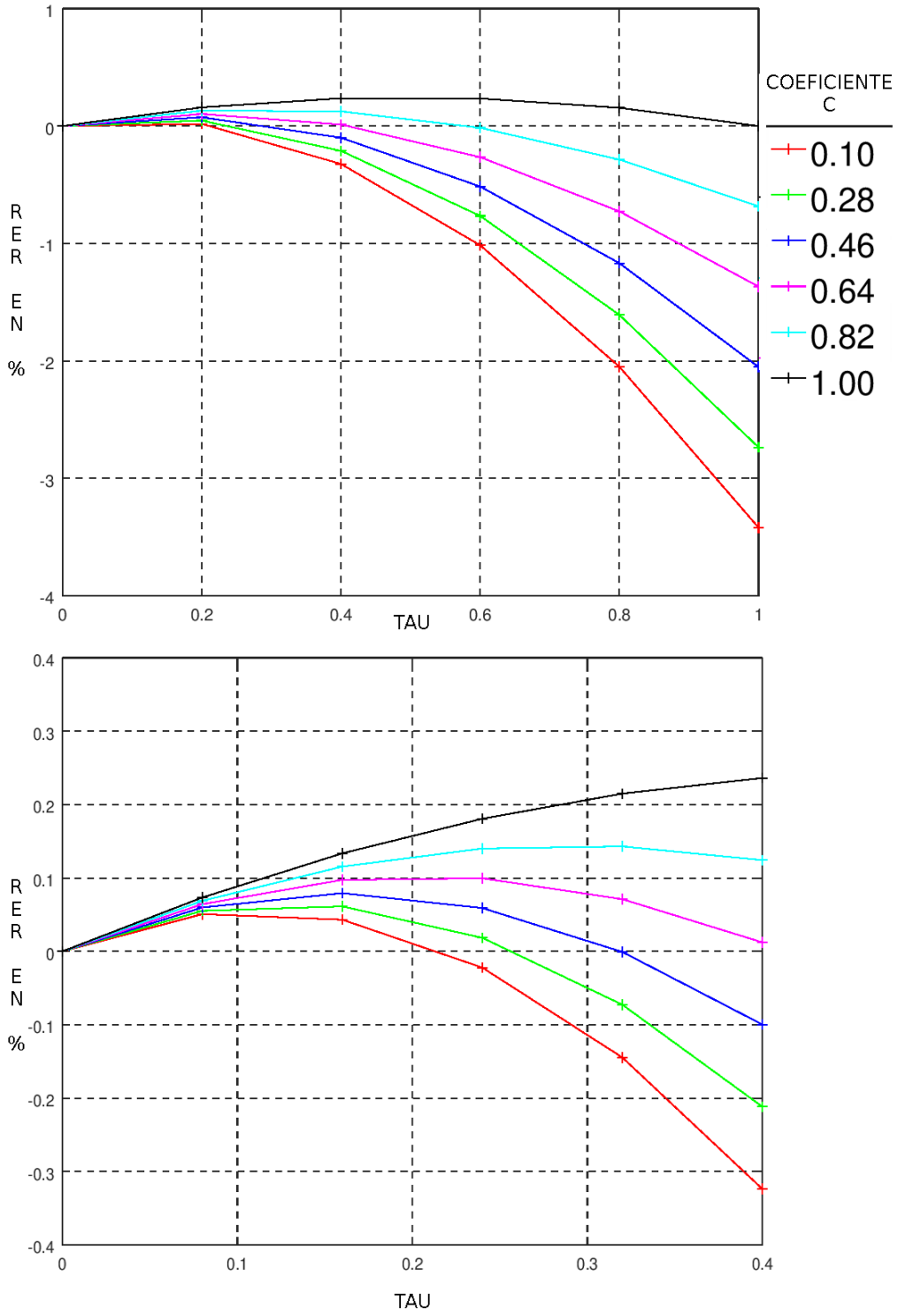

Figura 3.4: Ganancias en función de τ para diferentes valores de c, dado un atacante con poder de cómputo mucho menor que el *pool* objetivo ($\alpha = 0.01, \beta = 0.2$). La figura inferior es un acercamiento del mismo gráfico en la región $0 < \tau < 0.4$ donde se presentan los máximos.

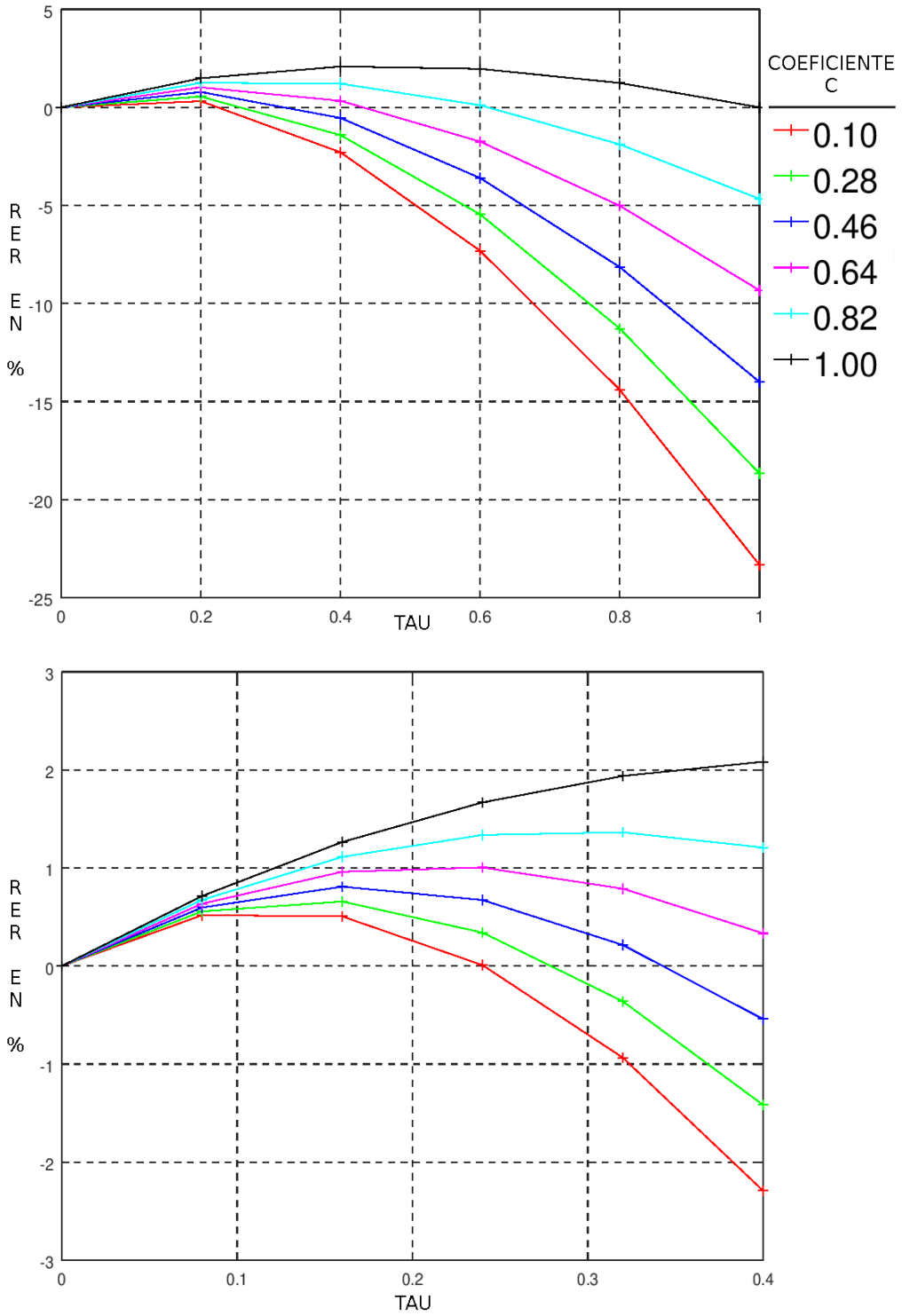

Figura 3.5: Ganancias en función de τ para diferentes valores de c, dado un atacante con poder de cómputo cercano al *pool* objetivo ($\alpha = 0.1, \beta = 0.2$). La figura inferior es un acercamiento del mismo gráfico en la región $0 < \tau < 0.4$ donde se presentan los máximos.

dio 200 BTC por día, significa que el ataque en el peor caso le puede generar aproximadamente 7081 dolares extra si destina el 10 % del poder de cómputo a atacar un *pool* de mayor tamaño.

Si se invierten los roles y el *pool* de mayor tamaño ataca al más pequeño (Figura 3.6), para un caso igual al anterior se observa que la ganancia es ligeramente menor para valores pequeños de c, y ligeramente superior a partir de $c = 0.4$. En general, se observa que a mayor tamaño del atacante, el ataque da más ganancias, y la mayor ganancia que da el ataque se observa en el caso extremo de que solo existan dos *pools* con 50 % del poder de cómputo cada uno, donde quien ataca puede mejorar su ganancia en un 11.11 %.

Requerimientos

Como se puede apreciar en la sección anterior, el ataque brinda una mayor recompensa cuanto mayor sea el poder de cómputo del atacante y del *pool* objetivo. Aún para un atacante con 0.05 % del poder de cómputo, se puede calcular que su frecuencia esperada de bloques es de uno cada veinte mil minutos. Aproximadamente un bloque cada dos semanas. Esto implica que su ganancia por mes es en promedio 25 BTC, y aún siendo despreciable en comparación a dicha ganancia, el ataque le puede dar en el peor caso ($c = 0$) una ganancia extra de 398 dolares mensuales, atacando a un *pool* de 0.2 % del poder de cómputo.

Para poder optimizar y predecir acertadamente la ganancia se debe conocer el poder de cómputo del *pool*, y el valor c, denominado comúnmente como capacidad de red. El valor de c es por definición mayor a 0 y menor a 1, dado que es la probabilidad de que el bloque sea seleccionado en la cadena principal, y tanto el atacante como el minero honesto van a elegir su propia cadena por sobre la del *fork* con el que compite. Si el administrador se comporta de forma esperada, el *pool* va a elegir a su vez a la cadena iniciada por el FPoW del atacante, haciendo que las probabilidades aumenten considerablemente cuanto mayor sea el poder de cómputo del *pool* atacado. De esta forma se puede utilizar al menos $\alpha + \beta$ como un piso para el valor de c. Esta cota ya hace que el ataque mejore respecto a BWH. Más aún, si el valor se mantiene constante, el atacante puede comparar las ganancias obtenidas con las teóricas para aprender y mejorar su estimación de c. Para realizar otro tipo

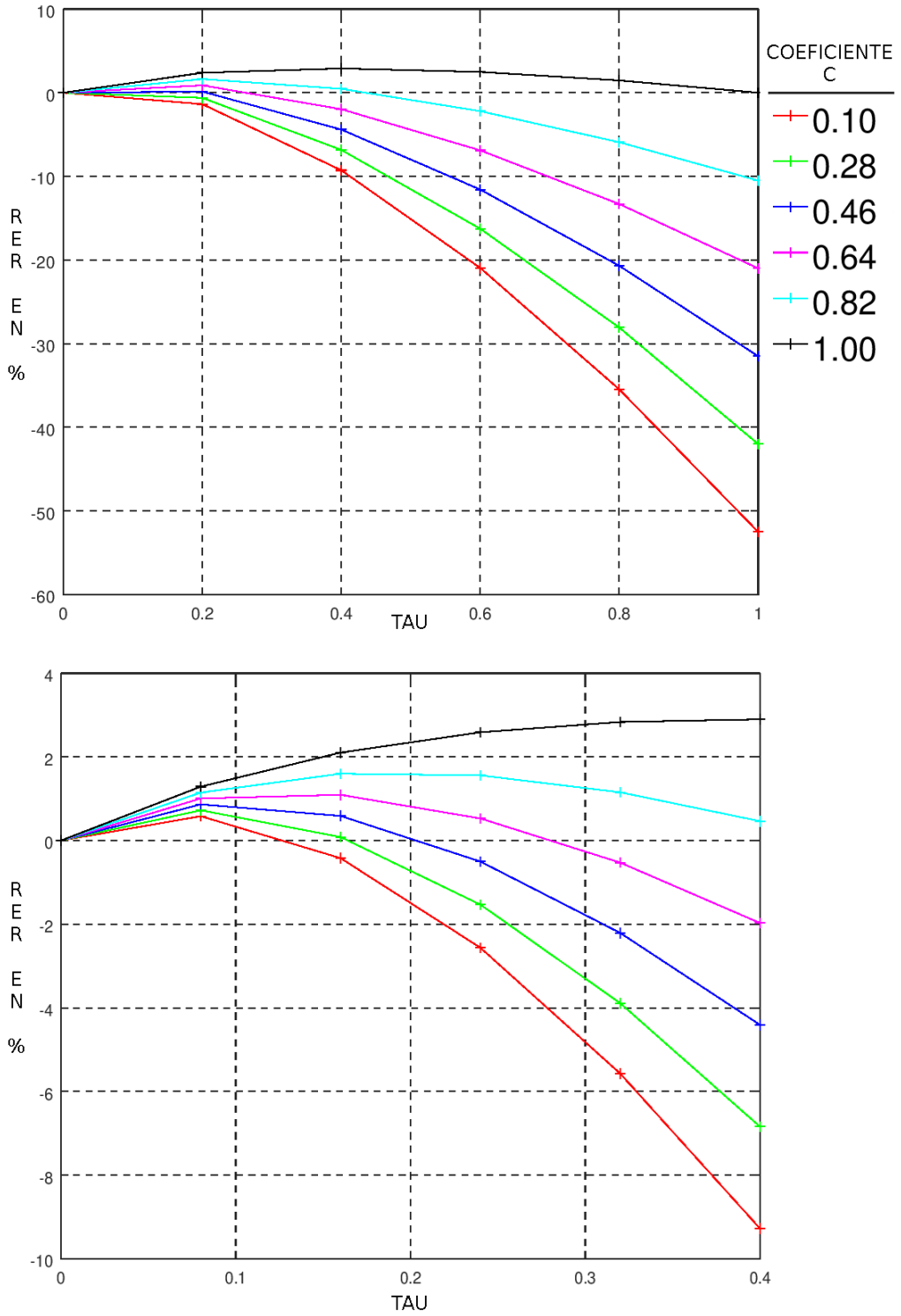

Figura 3.6: Ganancias en función de τ para diferentes valores de c, dado un atacante con poder de cómputo superior al *pool* objetivo ($\alpha = 0.2, \beta = 0.1$). La figura inferior es un acercamiento del mismo gráfico en la región $0 < \tau < 0.4$ donde se presentan los máximos.

de estimaciones más precisas debe tener conocimiento en tiempo real de la topología de la red y la distribución del poder de cómputo en la misma.

Si el atacante cuenta con suficiente poder de cómputo, puede intentar optimizar su ganancia aún más al atacar múltiples *pools*. Sin embargo, los resultados acerca de qué tan factible es mejorar la ganancia no son concluyentes, y cuanto más se fracciona el poder de cómputo, menos probable es encontrar un FPoW minando en cada uno de los *pools*. A su vez, las variables a considerar crecen rápidamente por cada *pool* agregado, dificultando la simulación de los resultados.

Ataque entre dos *pools*

El último caso, en que dos *pools* se ataquen simultáneamente, es el que presenta los resultados de mayor interés, dado que prueba que el dilema del minero no se cumple. Para facilitar la notación, se expresa como α_i al poder de cómputo del $pool_i$, y $f_i = \tau_i \alpha_i$ como el porcentaje de α_i usado para atacar (poder de infiltración).

Teorema 3.4.2. Dados dos *pools* con poder de cómputo α_1 y α_2, y $f_i = \tau_i \alpha_i$ con $i \in \{1,2\}$ igual al porcentaje de α_i usado para infiltrar al *pool* rival. La recompensa total para cada *pool* es:

$$R_1 = \frac{\alpha_1 - f_1}{1 - f_1 - f_2} + c_2 f_2 \frac{1 - \alpha_1 - \alpha_2}{1 - f_2} + c_2' f_1 f_2 \left(\frac{1}{1 - f_1} + \frac{1}{1 - f_2} \right) \frac{1 - \alpha_1 - \alpha_2}{1 - f_1 - f_2} + R_2 \frac{f_1}{\alpha_2 + f_1}$$

$$R_2 = \frac{\alpha_2 - f_2}{1 - f_1 - f_2} + c_1 f_1 \frac{1 - \alpha_1 - \alpha_2}{1 - f_1} + c_1' f_1 f_2 \left(\frac{1}{1 - f_1} + \frac{1}{1 - f_2} \right) \frac{1 - \alpha_1 - \alpha_2}{1 - f_1 - f_2} + R_1 \frac{f_2}{\alpha_1 + f_2}$$

donde c_i es la probabilidad de que el FPoW minado con el poder de infiltración f_i gane la condición de carrera cuando solo el FPoW del atacante i provoca un *fork*, y c_i' es el análogo cuando el *pool* rival también cuenta con un FPoW, provocando un segundo *fork* en la *blockchain*.

Se analizan a continuación los escenarios de ganancia del $pool_1$

- $\frac{\alpha_1 - f_1}{1-f_1-f_2}$ Es, al igual que en el teorema anterior, la ganancia por el porcentaje de poder computacional del atacante usado para minar honestamente.

- $c_2 f_2 \frac{1-\alpha_1-\alpha_2}{1-f_2}$ Es la probabilidad de que el $pool_2$ tenga un FPoW guardado como parte de su ataque al $pool_1$, al momento que un minero honesto propague un bloque.

- En el último caso, al momento de que el minero honesto propague el bloque, ambos *pools* tienen un FPoW minado en el *pool* rival, por lo que se comienza una condición de carrera entre ambos y el bloque del minero externo a los *pools*. Si el $pool_2$ encuentra un FPoW primero, la ganancia del $pool_1$ es $c_2^{'} f_2 \frac{f_1}{1-f_2} \frac{1-\alpha_1-\alpha_2}{1-f_1-f_2}$. Si en cambio el $pool_1$ encuentra el FPoW previo al $pool_2$, la ganancia es $c_2^{'} f_1 \frac{f_2}{1-f_1} \frac{1-\alpha_1-\alpha_2}{1-f_1-f_2}$. Por lo tanto, combinando ambos casos se obtiene que la ganancia es $c_2^{'} f_1 f_2 (\frac{1}{1-f_1} + \frac{1}{1-f_2}) \frac{1-\alpha_1-\alpha_2}{1-f_1-f_2}$.

- El $pool_2$ a su vez obtiene análogamente las ganancias descritas. Una parte de sus ganancias es recibida por el $pool_1$ como parte de su poder de cómputo infiltrado.

De igual manera que para el ataque a un *pool*, es posible calcular la ganancia relativa para un α_i, f_i, conociendo o estimando los c_i. La diferencia, como se puede apreciar en el teorema, es que las ecuaciones para cada *pool* son interdependientes. Lo interesante de este caso es ver si a diferencia de BWH, FAW se comporta de manera diferente respecto al dilema del minero. Lo que se espera, es que cuando ambos atacan, se observe un equilibrio de Nash en el que la ganancia de al menos uno de los *pools* obtenga una RER positiva. Para tener un equilibrio de Nash, se desea encontrar un par (f_1, f_2) que cumpla las condiciones $\frac{\delta R_1}{f_1} = 0$ y $\frac{\delta R_2}{f_2} = 0$, o lo que es equivalente a decir que alcancen un punto donde ninguna acción por parte de los *pools* sea favorable. Para probar esto, los autores del artículo partieron de la situación $(f_1, f_2) = (0,0)$ actualizando secuencialmente los valores en base al cambio que maximiza cada R_i. Las únicas restricciones aplicadas son respecto a los $c_i, c_i^{'}$, que se consideran simétricos para reducir variables. Como se aprecia en los resultados (figura 3.7), en la medida que el $pool_2$ tiene poder computacional menor o igual al $pool_1$, el $pool_1$ tiene posibilidades de obtener mayores ganancias que minando honestamente una vez alcanzado el equilibrio, mientras que el $pool_2$

obtiene perdidas. Si el $pool_2$ es mayor que el $pool_1$, entonces se da la situación inversa, y a mayor diferencia entre los poderes computacionales, menor es la necesidad de un valor alto de c para alcanzar ganancias relativas positivas. Esto es exactamente lo que se necesita para no cumplir con el dilema del minero, y se puede concluir que el *pool* de mayor tamaño es capaz de beneficiarse del ataque sin importar la acción del *pool* rival.

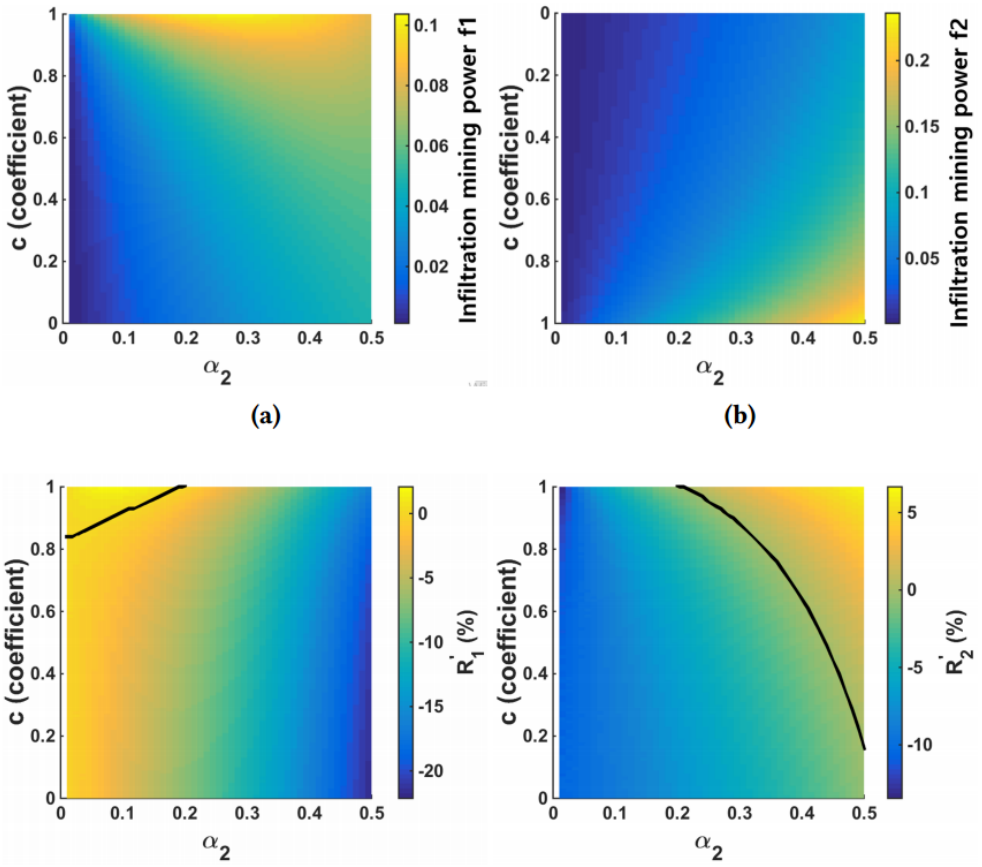

Figura 3.7: Para $\alpha_1 = 0.2$, se encuentran en las gráficas superiores los diferentes valores para f_1, f_2 en el equilibrio de Nash, variando c y α_2, mientras que en las gráficas inferiores se encuentran representadas las ganancias de cada *pool*. Con una linea de color negro se marca el límite donde la RER es igual a la ganancia. Por encima la RER es positivo reflejando un resultado favorable del ataque, y por debajo es negativo.(Figura 6 en [30]).

Posibilidad de ser detectado

Es posible detectar que el ataque ocurre. La detección puede ocurrir en forma indirecta si el *pool* comienza a producir *forks* sistemáticamente. El adminis-

trador puede llegar a identificar al atacante si el mismo intenta propagar repetidamente un FPoW luego de que el administrador ya haya recibido un bloque de un minero externo. La acción del administrador puede ser comportarse en forma honesta al protocolo, descartar el FPoW y expulsar al minero infiltrado. Sin embargo, la decisión favorable al *pool* es propagar el bloque del atacante. El atacante puede a su vez haberse infiltrado con múltiples nodos, cobrando igualmente parte de la recompensa del ataque, y reemplazando posteriormente al nodo infiltrado. No se han podido encontrar registros de que el ataque se haya observado en la práctica, a pesar de que las conclusiones de los autores indican que es de esperar que el ataque ocurra. Sí existen indicios de que un ataque de BWH fue realizado sobre un *pool* llamado Eligius en el año 2014, causando pérdidas estimadas en 300 BTC (ver sección XI A en [28]).

Posibilidad de ser contrarrestado

Si bien expulsar al sospechoso no es una medida eficiente, existen otras propuestas que pueden prevenir la ocurrencia del ataque, algunas de las cuales son expuestas a continuación:

- Oblivious Shares[19] - Es un PoW en dos fases (también denominado 2p-PoW), cuyo objetivo es que el minero no sepa al momento de entregar la *share*, si efectivamente ha encontrado un FPoW. El administrador es el único capaz de comprobarlo. Lo que se propone es agregar tres campos a los bloques: *SecretSeed*, *ExtraHash* y *SecretHash*. *ExtraHash* es el *hash* de *SecretSeed*; un valor numérico elegido por el administrador. *ExtraHash* forma parte del cabezal del bloque, usado en el cálculo del *hash*, y *SecretHash* es un *hash* de la concatenación del *hash* del bloque con el *SecretSeed*. Para que el bloque sea válido, en lugar de requerir que el *hash* sea menor a $\frac{2^{256}}{2^{32}.Dificultad}$, se requiere que sea menor a $\frac{2^{256}}{2^{32}}$, y que a su vez *SecretHash* sea menor a $\frac{2^{256}}{Dificultad}$. El administrador es el único que conoce *SecretSeed*, y por lo tanto el único que puede comprobar si las *shares* entregadas que cumplen con la primer desigualdad, cumplen además con la segunda. El motivo por el cual esta solución es un problema, es que no tiene compatibilidad hacia atrás con el PoW actual. Todos los mineros deben actualizar su *software* (y *hardware* especializado en la solución existente del PoW) para poder seguir minando,

y la seguridad de Bitcoin depende directamente de la cantidad de poder computacional invertido en minar. El cambio del protocolo beneficia a los *pools* públicos que son vulnerables a este tipo de ataque, pero no le aporta ningún beneficio al resto de mineros que no participan de *pools*, ni a los que participan en *pools* privados que no son vulnerables a estos ataques.

- Otro 2p-PoW propuesto por Ittay Eyal y Emin Gün Sirer [42]. Mejora los problemas de compatibilidad, pero su objetivo no es evitar este tipo de ataques, sino evitar la centralización del poder de cómputo en *mining pools*. La propuesta es que la primera fase del PoW se mantenga igual a la actual, y se introduce un nuevo *hash puzzle*. El administrador debe calcular la firma digital del cabezal del bloque usando la clave de la dirección en la cual recibe la recompensa del bloque, y el *hash* de esta firma debe ser menor a otro parámetro Y. Para el primer *hash puzzle* es posible disminuir la dificultad de forma que los mineros encuentren más candidatos a soluciones, que luego deben pasar por el segundo *hash puzzle*. El costo de realizar la firma digital y el *hash*, es similar al del doble *hash* del *hash puzzle* original. De esta forma, no es viable la existencia de *pools* de grandes proporciones donde el administrador deba efectuar el control de que la solución cumple con la dificultad. Compartir la información que permita distribuir la resolución del PoW completo implica dar a conocer la clave privada a los mineros. De ser así, los mineros pueden acceder a la recompensa completa y el administrador no es capaz de cumplir sus funciones. Nuevamente, el cambio no aporta ninguna ventaja a los mineros que no participan de *pools*, pero sí impone mayores dificultades a la existencia de *pools* de grandes proporciones. Los autores de FAW plantean que la comunidad no ha adoptado este cambio debido a los inconvenientes para aquellos que no son afectados por los ataques, y porque si bien la solución evita que los mineros ataquen al *pool*, el administrador aún es capaz de aplicar ataques como BWH. Si los mineros no saben que han encontrado un FPoW, el administrador puede retener parte de los bloques encontrados gracias a los esfuerzos de sus mineros, y cobrar la recompensa minando en forma privada.

- Otros autores ([36], [43]), han propuesto modificar el sistema de recompensa del *pool* de forma tal que R_a sea menor a R_h resultando en una

RER negativa. Si una fracción X de la ganancia del bloque es entregada al minero que encuentra el FPoW, la ganancia del atacante resulta $R_a = \frac{(1-\tau)\alpha}{1-\tau\alpha} + \frac{\beta}{1-\tau\alpha}.\frac{(1-X)\tau\alpha}{\beta+\tau\alpha} + c.\tau\alpha.\frac{1-\alpha-\beta}{1-\tau\alpha}.(X + \frac{(1-X)\tau\alpha}{\beta+\tau\alpha})$. Planteando la desigualdad donde $R_a < \alpha$, se obtiene que $X \geqslant \frac{\alpha}{1-c_{max}(1-\alpha-\beta)}$.
El administrador no conoce ni α ni c_{max} , pero puede estimar cotas para al menos evitar atacantes de hasta determinado poder computacional. Por ejemplo, para un *pool* de $\beta = 0.2$ y un atacante de iguales dimensiones, en el peor caso ($c = 1$), $X = 0.5$. Esto significa que la mitad de la recompensa va para el que minero que encuentra el bloque. En la misma situación con $\alpha = 0.1$, X desciende a 33 %.
El problema con esta solución, es que cuanto mayor sea X, menos atractivo es el *pool* para los mineros, dado que incrementa la varianza en la ganancia.

3.5 Resumen y Comparación

3.5.1 Clasificación

	BiteCoin	***Selfish Mining***	***Fork After Withholding***
Clasificación por rol			
De una entidad a un *pool*	✓		✓
De un *pool* a la comunidad		✓	
Pool vs *Pool*			✓
Clasificación por objetivo			
Al protocolo de consenso		✓	✓
Al protocolo de los *pools*	✓		
De red	✓	Puede combinarse con ataques de red para mejorar el ataque (ver capítulo 5)	

Tabla 3.4: Resumen de la clasificación de los ataques.

3.5.2 Efectividad

Para el caso de *BiteCoin* no se ha encontrado forma de tomar una medida comparativa de la eficiencia, ya que en los demás ataques se puede hablar de poder de cómputo destinado al ataque versus poder de cómputo usado para minar honestamente, mientras que en *BiteCoin*, los insumos para realizar el ataque son diferentes y no es trivial deducir cuánto poder de cómputo se resta al implementar el ataque, o cuánta ganancia por poder de cómputo invertido brinda el mismo.

En los casos de *Selfish Mining* y FAW, para poderes de cómputo pequeños, *Selfish Mining* no es rentable y produce pérdidas respecto a minar honestamente, mientras que FAW sigue siendo favorable atacar. Para grandes poderes de cómputo, ambos son favorables, pero la RER muestra que *Selfish Mining* ofrece una mayor recompensa respecto a FAW, siempre que el γ sea suficientemente alto.

Para $\alpha = 0.2$, *Selfish Mining* puede llegar a incrementar las ganancias en un 10 % en el mejor caso, mientras que en FAW, con el mismo poder de cómputo, y un *pool* rival de iguales proporciones, la ganancia relativa se encuentra entre 1.11 % y 3.73 % según el parámetro c. A su vez, para γ pequeño, *Selfish Mining* puede seguir dando pérdidas, por lo que FAW es una opción con menor varianza en la ganancia relativa.

3.5.3 Requerimientos

<table>
<tr><th></th><th>Poder
de
cómputo
mínimo</th><th>Uso
de memoria
RAM</th><th>Acceso
a la
red</th></tr>
<tr><td>BiteCoin</td><td colspan="2">El cuello de botella para
implementarlo no es el poder
de computo ni la memoria RAM,
sino que depende del acceso a la red</td><td>Se requiere que
el atacante pueda
realizar el Session Hijacking
entre el minero y el
administrador, que le
permita leer y modificar
los paquetes que se envían</td></tr>
<tr><td>Selfish Mining</td><td>Con $\gamma = 0.5$, un 25 %
del poder total
de cómputo
de Bitcoin.</td><td rowspan="2">Mientras se siga usando
el PoW actual y no uno
basado en memoria
(memory hard) [44],
el cuello de
botella para minar
es el poder de cómputo
y no la memoria RAM</td><td rowspan="2">No requieren, pero
cualquier conocimiento
extra de la topología de la
red para estimar o
influenciar el γ o
c y así predecir mejor
las ganancias esperadas,
es beneficioso al atacante.</td></tr>
<tr><td>FAW</td><td>El ataque
es
siempre
favorable</td></tr>
</table>

Tabla 3.5: Resumen de los requerimientos de los ataques.

3.5.4 Posibilidad de ser detectado

Todos los ataques estudiados, ejecutados tal y como fueron planeados y sin ninguna variante, son posibles de detectar aunque en la mayoría de los casos de forma pasiva, es decir, se puede inferir que se está siendo atacado, pero no tener una total certeza de esto. Cuando la detección es activa, es posible detectar el origen del ataque. A continuación, una tabla comparativa con algunos aspectos:

	¿Quién puede detectarlo?	Método de detección	Detecciones registradas
BiteCoin	El minero víctima	El minero víctima sabe cuánto le corresponde en base al sistema de pago del *pool*.	No encontrado
Selfish Mining	Toda la red indirectamente. Los mineros del *pool* directamente	Indirectamente por la forma de la *blockchain*. Directamente al ver que el administrador no propaga los FPoW.	No en Bitcoin. Primer registro en Monacoin en mayo del año 2018 con pérdidas cercanas a noventa mil dólares [45] [46].
FAW	Toda la red indirectamente. El administrador del *pool* directamente	Indirectamente por la generación sistemática de *forks*. Directamente por recibir FPoW luego de recibir un bloque de la red.	No encontrado. Sí hay registro de un ataque de BWH, que es similar en funcionamiento. Sección XI A en [28]

Tabla 3.6: Resumen de la posibilidad de ser detectado de los ataques.

Recordar que para *Selfish Mining*, uno de los problemas principales asociados a la detección, es que si es realizado por un *pool* público, retener a sus mineros puede resultar difícil. Los mineros honestos del *pool* pueden optar por abandonarlo. En FAW, es menos probable que los mineros del *pool* atacante puedan detectar quién está realizando el ataque, ya que el *pool* propaga sus FPoW inmediatamente. Para sus mineros, es el *pool* rival quien se comporta en forma anómala, y no el *pool* propio. Esto aporta a que si bien la eficiencia del FAW para *pools* grandes es menor que en *Selfish Mining*, el FAW presenta mayores garantías de éxito en el análisis teórico.

3.5.5 Posibilidad de ser contrarrestado

En los tres ataques existen medidas que se pueden tomar para contrarrestarlos, ya sea una vez detectado o en forma preventiva. Sin embargo, la gran mayoría presenta inconvenientes que las hacen impracticables o que disminuyen el interés de aplicarlas. A continuación se ilustran a modo de resumen las medidas discutidas en cada ataque.

	Solución completa	¿Es una medida preventiva?	Costo
BiteCoin			
Cifrar comunicación	✓.	✓	Pérdida significativa de eficiencia.
Usar protocolo *Bedrock*	✓	✓	Disminuye la eficiencia en menor medida, y requiere un intercambio previo de claves.
Selfish Mining			
Elección aleatoria de Cadena	No, solo busca aumentar el mínimo de poder de cómputo requerido para poder realizar el ataque.	✓	No, pero debe ser adoptado por toda la comunidad para que funcione tal como se plantea.
Ajuste del nivel de dificultad	No elimina la posibilidad de realizar el ataque, pero sí hace más atractivo el minar honestamente respecto a unirse al ataque.	✓	Introducir un cambio al algoritmo de consenso de Bitcoin, y que sea adoptado por toda la comunidad.
Fork After Withholding			
Expulsar al atacante	No, el atacante puede infiltrar múltiples nodos e igual beneficiarse del ataque.		
Oblivious Shares	✓	✓	Implica cambiar el algoritmo PoW de toda la red Bitcoin, lo cual induce un peligro para la misma, y solo beneficia a los *pools* públicos.
2p-PoW de Ittay	✓	✓	Hace la existencia de *pools* de gran tamaño impracticable. No evita que el administrador aplique otros ataques similares como BWH.
Cambiar fracción X de la recompensa por el FPoW	No, según el valor de X se desincentiva el ataque para infiltración con hasta cierto poder de cómputo.	✓	Cuanto mayor el X elegido, menor es el atractivo para los usuarios de elegir dicho *pool*, dado que aumenta la varianza.

Tabla 3.7: Resumen de la posibilidad de ser contrarrestado de los ataques.

Capítulo 4

CONCLUSIONES

A lo largo de este trabajo se lograron comprender las estructuras y los algoritmos que permiten el funcionamiento de una criptomoneda como Bitcoin. Se pudo comprender a su vez el funcionamiento y la necesidad de contar con un protocolo que permita agrupaciones como los *mining pools*. El análisis y la evaluación de los ataques seleccionados muestra, sin embargo, que estas agrupaciones presentan múltiples debilidades que pueden afectar no solo a los participantes de los propios *mining pools*, sino a toda la comunidad de Bitcoin. Respecto a este punto las observaciones de mayor importancia son las siguientes:

- Desde el punto de vista de la teoría, todos los ataques analizados son realizables con recursos que son obtenibles en la práctica. Si bien un ataque como *Selfish Mining* requiere un gran poder de cómputo y el riesgo de pérdida es alto, el reciente ataque descubierto en Monacoin muestra que también es una amenaza que debe ser tomada en cuenta por la comunidad de toda criptomoneda.

- A pesar de esta situación, no existen aún registros de que estos ataques se hayan realizado en Bitcoin. Ataques como FAW fueron propuestos en tiempo reciente y los propios autores preveen que el ataque sea llevado a cabo, dado las ventajas que tiene sobre *Selfish Mining* y sobre BWH, ataque sobre el cual sí existe un hecho registrado. Es posible que los incentivos en Bitcoin actualmente cumplan con el cometido de incentivar a los mineros a no correr el riesgo de tomar acciones que pueden afectar la estabilidad de la moneda.

- Los tres ataques son detectables. Quién puede detectarlo depende del ataque como se ve en la tabla 3.6, pero en todos los casos, si el participante es consciente de la existencia del ataque, puede implementar controles que detecten su ocurrencia o la sospecha de su ocurrencia. Para los *pools* en *Selfish Mining* y FAW puede implicar la pérdida de reputación en caso de ser públicos y contar con un tamaño que pueda

afectar a la comunidad entera. Para *BiteCoin*, al afectar a mineros en forma individual y no afectar al *pool*, puede que el ataque ocurra actualmente sin que llame la atención de la comunidad como en los otros casos. Estos pueden ser motivos por los cuales no se han observado ocurrencias en la literatura. Sin embargo, la tecnología y los estudios son recientes y no debe descartarse la posibilidad de que ocurran en el futuro.

- Es difícil aplicar medidas para contrarrestar ataques en una criptomoneda como Bitcoin. Por una parte debido a que la seguridad del sistema se encuentra impuesta en mayor medida por la participación de una comunidad bien distribuida de mineros en el PoW, y no tanto por el uso de la tecnología. Cualquier cambio que no sea compatible con el PoW actual es de riesgo para Bitcoin por la propia descentralización. Por otra parte, cambios que incrementen la seguridad de los *mining pools* se encuentran limitados por la centralización del poder por parte del administrador. Propuestas que se han planteado para solucionar estos problemas pueden verse en el capítulo 5.

Capítulo 5

TRABAJO A FUTURO

5.1 Análisis y evaluación de ataques en Ethereum

Ethereum es una criptomoneda cuyo objetivo es ser una maquina virtual distribuida de propósito general. Es una criptomoneda por el hecho de basarse en la tecnología *blockchain*, y contar con su propia moneda, denominada ether, pero el lenguaje *script* ejecutado durante el procesamiento de las transacciones es de una expresividad mucho mayor, siendo un lenguaje Turing-Completo. Para evitar el problema de parada de un lenguaje de estas características [47], se incorpora un recurso denominado Gas, que debe pagarse por la ejecución de cada sentencia de código. Actualmente Ethereum utiliza un algoritmo PoW al igual que Bitcoin, con algunas diferencias para intentar contrarrestar el avance de los ASIC. Existen varios motivos por los que resulta interesante incursionar en Ethereum para la investigación de los ataques seleccionados. Uno de ellos es una característica del sistema de recompensa, que presenta ventajas para la realización de un ataque que genere *forks* como *Selfish Mining* y FAW. Dado que el tiempo de generación de bloques en Ethereum es tan solo de aproximadamente quince segundos, se pueden generar mayor cantidad de bloques huérfanos como consecuencia de las latencias propias de la red. Para contrarrestar los efectos de este problema sobre la motivación de los mineros, Ethereum implementa como parte de su sistema de recompensa, una adaptación de un protocolo denominado GHOST [48]. Los bloques pueden incluir un listado de bloques huérfanos, tíos del bloque actual. Algunas veces se le denomina coloquialmente bloque padre al bloque predecesor, y un bloque tio corresponde a un bloque que ocupa la misma posición en la cadena que el bloque padre, pero que se encuentra en un *fork*. Los mineros de los bloques tíos, así como los que los incluyen en el listado de un bloque, reciben una recompensa extra en ether. Estudios recientes [49] muestran que la existencia de estas recompensas favorece al *Selfish Mining* de forma que disminuye aún más el poder de cómputo mínimo para que sea rentable. Mientras que para

$\gamma = 0.5$, el mínimo poder de cómputo necesario en Bitcoin es de 25 %, en Ethereum, debido a esta implementación puede llegar a ser de 18.5 %, y se plantea que se puede optimizar la estrategia para reducir el mínimo aún más.

5.2 Análisis y evaluación de ataques en otros tipos de *pool*

Otra motivación para explorar las características de Ethereum es la versatilidad del lenguaje *script*. A estos *scripts* que pueden ser ejecutados durante la transacción se les denomina contratos inteligentes [50] (*Smart Contracts*). En Ethereum es posible desarrollar múltiples aplicaciones distribuidas. Una de las aplicaciones que ha sido que se plantea en la literatura es la de un *mining pool* administrado por un *Smart Contract*. A esta aplicación se la considera un nuevo tipo *pool*, y se lo denomina *Smart-Pool* [51]. Debido a que el código de los *Smart Contracts* es público y no puede modificarse, se espera que un *Smart-Pool* sea más transparente y seguro, además de poder ofrecer una menor varianza a los mineros.

Una última característica de Ethereum que en un futuro puede resultar interesante, es el plan actual de sus desarrolladores de cambiar el algoritmo de consenso. Ethereum piensa abandonar el paradigma de PoW para adoptar un protocolo llamado Casper que utiliza un paradigma diferente denominado prueba de participación [52] (*Proof of Stake* o PoS). En este paradigma, el consenso no se alcanza mediante poder de cómputo, por lo que puede resultar interesante analizar la posibilidad de aplicar los ataques, así como las modificaciones necesarias para que funcionen en dicho contexto. En la actualidad ya hay criptomonedas como Cardano que han experimentado con este paradigma [53], y a su vez, ya existe una alternativa para los *mining pools*, denominadas agrupaciones de participación [54] (*staking pools*).

La última alternativa de *pool* que se propone, es una que ya existe en Bitcoin. Un tipo de *pool* donde se vuelve a aplicar la idea de descentralización, para obtener los denominados P2P-*pools*. El más popular a la fecha se llama P2Pool [55]. Los P2P-*pools* son *pools* en donde no hay administrador, y las tareas del administrador se encuentran distribuidas, removiendo la necesidad de confiar en la entidad central, pero requiriendo un nuevo nivel de consenso.

Todas estas alternativas de *mining pools* ofrecen un espectro más amplio de posibilidades para estudiar los efectos que tienen los ataques y las variantes que pueden emanar de los mismos.

5.3 Mejoras con ataques de red

Durante la clasificación de ataques en la sección 3.1.1 se da a conocer que el *Selfish Mining* cuenta con estudios que proponen mejorar la eficiencia del ataque al combinarlo con un ataque de red llamado *Eclipse* [32]. Entender en qué medida este ataque puede mejorar la efectividad de *Selfish Mining*, al igual que ver si los beneficios son aplicables a los demás ataques, es un punto interesante sobre el cual se puede continuar indagando.

BIBLIOGRAFÍA

[1] Satoshi Nakamoto. Bitcoin: A Peer-To-Peer electronic cash system. Octubre 2008.

[2] Bayu Adhi Tama, Bruno Joachim Kweka, Youngho Park, y Kyung-Hyune Rhee. A Critical Review of Blockchain and Its Current Applications. 2017.

[3] Arthur Gervais, Ghassan O. Karame, Srdjan Capkun, y Vedran Capkun. Is Bitcoin a Decentralized Currency? 2014.

[4] Yoad Lewenberg, Yoram Bachrach, Yonatan Sompolinsky, Aviv Zohar, y Jeffrey S. Rosenschein. Bitcoin Mining Pools: A Cooperative Game Theoretic Analysis. 2015.

[5] Hashrate distribution: An estimation of hashrate distribution amongst the largest mining pools, . URL `https://www.blockchain.com/en/pools`. (Último acceso 30-Nov-2018).

[6] Joseph A. Ritter. The Transition from Barter to Fiat Money. Marzo 1995.

[7] Joachim von zur Gathen. *Cryptoschool*. 2015.

[8] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miler, y Steven Goldfeder. Bitcoin and Cryptocurrency Technologies. página 23, 2016.

[9] James F. Kurose y Keith W. Ross. *Computer Networking: A top down approach*. Pearson, 6ta edición, 2013.

[10] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miler, y Steven Goldfeder. Bitcoin and Cryptocurrency Technologies. páginas 23–36, 2016.

[11] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miler, y Steven Goldfeder. Bitcoin and Cryptocurrency Technologies. páginas 37–42, 2016.

[12] Muhammad Saqib Niaz y Gunter Saake. Merkle Hash Tree based Techniques for Data Integrity of Outsourced Data. 2015.

[13] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miler, y Steven Goldfeder. Bitcoin and Cryptocurrency Technologies. 2016.

[14] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miler, y Steven Goldfeder. Bitcoin and Cryptocurrency Technologies. páginas 101–112, 2016.

[15] James F. Kurose y Keith W. Ross. *Computer Networking: A top down approach*. Pearson, 6ta edición, 2013.

[16] Athanasios Papoulis. *Probability random variables and stochastic processes*. Mcgraw-Hill, 3ra edición, 1991.

[17] Jordi Herrera-Joancomart´ı. Research and Challenges on Bitcoin Anonymity. 2014.

[18] Rahul P. Naik. Optimising the SHA256 Hashing Algorithm for Faster and More Efficient Bitcoin Mining. Septiembre 2013.

[19] Meni Rosenfeld. Analysis of Bitcoin Pooled Mining Reward Systems. Diciembre 2011.

[20] Ben A. Fisch, Rafael Pass, y Abhi Shelat. Socially Optimal Mining Pools. páginas 1–22, Marzo 2017.

[21] Getwork, . URL `https://en.bitcoin.it/wiki/Getwork`. (Último acceso 15-Nov-2018).

[22] Protocolo de minería stratum, . URL `https://slushpool.com/help/manual/stratum-protocol`. (Último acceso 15-Nov-2018).

[23] Ruben Recabarren* y Bogdan Carbunar. Hardening Stratum, the Bitcoin Pool Mining Protocol. páginas 1–18, Marzo 2017.

[24] Arvind Narayanan, Joseph Bonneau, Edward Felten, Andrew Miler, y Steven Goldfeder. Bitcoin and Cryptocurrency Technologies. página 155, 2016.

[25] Mauro Conti, Sandeep Kumar E, Chhagan Lal, y Sushmita Ruj. A Survey on Security and Privacy Issues of Bitcoin. Diciembre 2017.

[26] Joseph Bonneau, Andrew Miller, Jeremy Clark, Arvind Narayanan, Joshua A. Kroll, y Edward W. Felten. Research Perspectives and Challenges for Bitcoin and Cryptocurrencies. 2015.

[27] James F. Kurose y Keith W. Ross. *Computer Networking: A top down approach.* Pearson, 6ta edición, 2013.

[28] Nicolas T. Courtois y Lear Bahack. On Subversive Miner Strategies and Block Withholding Attack in Bitcoin Digital Currency. Diciembre 2014.

[29] Cyril Grunspan y Ricardo Pérez-Marco. ON PROFITABILITY OF SELFISH MINING. Mayo 2018.

[30] Yujin Kwon, Dohyun Kim, Yunmok Son, Eugene Vasserman, y Yongdae Kim. Be Selfish and Avoid Dilemmas: Fork After Withholding (FAW) Attacks on Bitcoin. Agosto 2017.

[31] Maria Apostolaki, Aviv Zohar, y Laurent Vanbever. Hijacking Bitcoin: Routing Attacks on Cryptocurrencies. Mayo 2017.

[32] Ethan Heilman, Alison Kendler, Aviv Zohar, y Sharon Goldberg. Eclipse Attacks on Bitcoin's Peer-to-Peer Network. Agosto 2015.

[33] John R. Douceur. The Sybil Attack. 2002.

[34] Abdullah H. Alqahtani y Mohsin Iftikhar. TCP/IP Attacks, Defenses and Security Tools. 2013.

[35] William Poundstone. *El dilema del prisionero.* Septiembre 2015.

[36] Ittay Eyal. The Miner's Dilemma. Noviembre 2014.

[37] James F. Kurose y Keith W. Ross. *Computer Networking: A top down approach.* Pearson, 6ta edición, 2013. Capítulo 3.

[38] Joachim von zur Gathen. *Cryptoschool.* 2015.

[39] Ittay Eyal y Emin Gün Sirer. Majority is not Enough: Bitcoin Mining is Vulnerable. Noviembre 2013.

[40] Arthur Gervais, Ghassan O Karame, Karl Wust, Vasileios Glykantzis, Hubert Ritzdorf, y Srdjan Capkun. On the Security and Performance of Proof of Work Blockchains. 2016.

[41] Tomaso Aste. The fair cost of Bitcoin proof of work. 2016.

[42] Ittay y Emin Gün Sirer. How to disincentivize large bitcoin mining pools. URL http://hackingdistributed.com/2014/06/18/how-to-disincentivize-large-bitcoin-mining-pools/. (Último acceso 30-Sep-2018).

[43] Loi Luu, Ratul Saha, Inian Parameshwaran, Prateek Saxena, y Aquinas Hobor. On Power Splitting Games in Distributed Computation: The Case of Bitcoin Pooled Mining. 2015.

[44] Joe Lao. A network-dependent rewarding system: proof-of-mining. Septiembre 2014.

[45] Muhammad Saad, Laurent Njilla, Charles Kamhoua, y Aziz Mohaisen. Countering Selfish Mining in Blockchains. Noviembre 2018.

[46] Selfish mining en monacoin, . URL https://www.ccn.com/japanese-cryptocurrency-monacoin-hit-by-selfish-mining-attack/. (Último acceso 03-Dic-2018).

[47] Behrouz A. Forouzan. *Introducción a la ciencia de la computación.* 2003.

[48] Yonatan Sompolinsky y Aviv Zohar. Secure High-Rate Transaction Processing in Bitcoin. Diciembre 2013.

[49] Fabian Ritz y Alf Zugenmaier. The Impact of Uncle Rewards on Selfish Mining in Ethereum. Agosto 2018.

[50] Dr Eliza Mik. Smart Contracts: Terminology, Technical Limitations and Real World Complexity. Octubre 2017.

[51] Loi Luu, Yaron Velner, Jason Teutsch, y Prateek Saxena. SMARTPOOL: Practical Decentralized Pooled Mining. Enero 2017.

[52] Vitalik Buterin y Virgil Griffith. Casper the Friendly Finality Gadget. Noviembre 2017.

[53] Spencer J. Hosack. Use of the Proof-of-Stake Algorithm for Distributed Consensus in Blockchain Protocol for Cryptocurrency. *Honors Scholar Theses*, 2018.

[54] James Tan. A high-throughput cryptocurrency featuring decentralized staking pools. Abril 2018.

[55] P2pool, . URL http://p2pool.org/learn/index.php. (Último acceso 08-Nov-2018).