LibroATMcash

blanco de la presentación técnica dePre-desarrollo.

(Este documento fue escrito antes del desarrollo real, y finalmente se utilizó otro método. Esta es al menos la mayor parte de las matemáticas y explicaciones que eventualmente se utilizaron, y las explicaciones allí. El último diseño elegido sigue todas las reglas mencionadas y es superior a lo que se describe aquí.)

Una criptomoneda basada en pruebas de capacidad

(Sunoo Park, Krzysztof Pietrzak, Albert Kwon, Joel Alwen, Georg Fuchsbauer, Peter Gazi, con adiciones, aclaraciones y formateo por parte del equipo técnico de ATMcash postdesarrollo)

Proponemos una descentralización criptomoneda llamadaATMcash, que se basa en un libro mayor de blockchain similar al hatof Bitcoin, pero donde las pruebas de trabajo derrochadoras se reemplazan por pruebas eficientes de capacidad, recientemente presentado por Dziembowskiet al. En lugar de requerir que la mayoría del poder de cómputo de la red sea controlado por mineros honestos (como en Bitcoin), nuestra moneda requiere que los mineros honestos dediquen másdisco neto
capacidad deque un adversario potencial.

En ATMCash, una vez que un minero ha dedicado e inicializado alguna capacidad, participar en el proceso de minería es muy barato. Se agrega un nuevo bloque a la cadena cada período fijo de tiempo, y
en cada período un minero solo tiene que hacer un pequeño número de búsquedas hasta la capacidad almacenada para verificar si ella "gana", y así puede agregar eficientemente el siguiente bloque a la cadena y obtener la recompensa minera. En este documento, detallamos la construcción de ATMcash, analizamos sus propiedades de seguridad y teoría de juegos, y estudiamos su
rendimiento. Nuestro prototipo muestra que demora aproximadamente 25 segundos para demostrar más de un terabyte de capacidad, y lleva una fracción de segundo verificar la prueba.

I. INTRODUCCIÓN

Bitcoin es una moneda digital descentralizada que se introdujo
en 2009 y ahora es, con mucho, ladigital más exitosa
monedajamás desplegada. La contabilidad descentralizada de la moneda
depende de mantener un libro de contabilidad público que registra todas las
transacciones que ocurren. Este libro de contabilidad se implementa mediante unabloques
cadena de: es decir, una secuencia de bloques, cada uno de los cuales contiene
registros de transacciones y cierta información auxiliar, que
son generados por los participantes en la red. Para alentar a los

participantes a contribuir con bloques, aquellos que agregan un bloque a la cadena son recompensados con un bitcoin recién acuñado.

Una dificultad principal al diseñar una moneda digital es proporcionar seguridad contra el doble gasto: es decir, el propietario de una moneda debe poder gastarla exactamente una vez. Para evitar el doble gasto, se debe hacer cumplir que todas las partes de la red acuerden la misma cadena de bloques (excepto posiblemente para los bloques más recientes). Antes de aceptar una transacción, un destinatario debe esperar hasta que la transacción haya estado en la cadena el tiempo suficiente como para que pueda estar razonablemente segura de que permanecerá allí para siempre (es decir, se ha llegado a un consenso). El protocolo de Bitcoin logra consenso al hacer que sea complicado agregar un bloque a la cadena: actualmente,

1. En consecuencia, un bloque de Bitcoin se considera una prueba de trabajo: es decir, una prueba de queuna cierta cantidad de se invirtiórecursos computacionales. .

La red de Bitcoin explota un bloque aproximadamente cada 10 minutos, por lo que consume recursos computacionales (y recursos naturales asociados, principalmente en forma de electricidad) a gran escala. El uso actual de la red se estima en varios 100 MW de potencia; además, la mayor parte de la minería se realiza actualmente mediante hardware dedicado, quetiene nomás uso que minar Bitcoins. Por estas razones, Bitcoin es considerado un "desastre ambiental" por algunos.

La idea original de basar la minería de Bitcoin en elcómputo poder deera que cualquiera podía participar en la red dedicando sus ciclos de CPU de repuesto, lo que implica un pequeño costo marginal ya que usa el tiempo de inactividad de lasya existentes computadoras personales. Sin embargo, la dinámica de lamoderna de mineríaBitcoin se ha vuelto muy diferente: la mayoría de la minería exitosa es realizada por granjas mineras a gran escala, a menudo en colaboración con productores de electricidad. Sin unespecializado hardware ASIC de minería, no tiene ninguna posibilidad: la minería con sus ciclos de CPU de repuesto perderá dinero debido a los costos de electricidad. La no linealidad de las recompensas mineras en Bitcoin en

relación con los recursos invertidos se ha cuantificado Diseñando una criptomoneda donde la recompensa esperada es más proporcional a los recursos invertidos sería deseable por varias razones, incluyendo que la presencia de "pequeños jugadores" puede ser importante para la estabilidad y la descentralización de la moneda.

Para abordar estos problemas, en este documento proponemos una criptomoneda, ATMcash, que reemplaza las costosas pruebas de trabajo que subyacen a Bitcoin con pruebas de capacidad. En ATMcash, para minar bloques (y por lo tanto monedas de menta), los mineros deben invertir ladisco capacidad del, en lugar de la capacidad de cálculo.

2. Los mineros que dedican más capacidad de disco tienen unaproporcionalmente expectativamayor de minar con éxito un bloque y cosechar la recompensa. En estos días, enormes cantidades de capacidad de disco se encuentran sin usar, y toda esta capacidad de almacenamiento tiene potencial para la minería. Observamos que en un esquema basado en la capacidad, está claro que los mineros se verán incentivados a invertir endisco duro capacidad de, al igual que los mineros de Bitcoin están incentivados para invertir en electricidad. Sin embargo, destacamosdiferencias clave:

algunas1) En ATMcash, la inversión se realiza en forma dede capital gasto, y el proceso de minería después de lalos discos duros compra deimplica un costo indirecto insignificante (tanto en términos monetarios como de recursos naturales). Por el contrario, en Bitcoin, el proceso de minería requiere unenergía perpetuo gasto dede los mineros.

2) En Bitcoin, los recursos son "agotados" por la minería: la electricidad es un recurso que se agotó y eles un recurso hardware de minería de Bitcoinespecializado de un solo propósito que no sirve para nada una vez que se elimina la minería de Bitcoin. . Por el contrario, el recurso consumido por ATMcash es reciclable, ya que se puede usar una y otra vez y es multifuncional, ya que los discos duros tienen un valor intrínseco en su capacidad de almacenar datos útiles.1 La distinción entre gastos de capital yrecurrentes costos generaleses significativo también porque cambia la compensación entre la recompensa esperada y los recursos mineros invertidos. Más concretamente, debido al bajo costo marginal de explotar un bloque, la forma de la curva representada en la Figura 1 será mucho más plana para ATMcash que Bitcoin, y por lo tanto el área rentable se distribuirá más uniformemente sobre el eje horizontal.

A. Antecedentes y desafíos

Una criptomoneda "más verde". La comunidad ha buscado
protocolos de consenso descentralizados alternativos y encontró un
candidato potencialmente prometedor en pruebas de participación (PoStake).
En dichos esquemas, la probabilidad de que una parte mine el siguiente
bloque es proporcional a la fracción de monedas (de todas las monedas que
pertenecen a los mineros participantes) que posee. Esta idea es
muy atractiva ya que norecursos (como energía, hardware, etc.)
se desperdician, pero desafortunadamente, hacer que este enfoque realmente
funcione resulte mucho más delicado que para los esquemas basados en la
prueba de trabajo (PoW).

Al tratar de adaptar Bitcoin de manera directa reemplazando
PoW por PoStake, uno se encuentra con al menos tresprincipales
problemasque se detallan a continuación. Intuitivamente, los primeros dos
problemas están relacionados con el hecho de que la producción de unaPoStake
prueba dees computacionalmente barata y, por lo tanto, abre el potencial de
hacer trampa de formas que no son posibles en Bitcoin. Veremos
que surgen desafíos análogos a partir de la facilidad computacional
de extraer un bloque por medio de una prueba de capacidad; y nuestraATMCash
construcciónpropondrá formas de resolver estos desafíos.

1) Cadenas múltiples: en Bitcoin, un minero racional siempre
trabajará para extender la cadena más larga de la que tenga
conocimiento, ya que trabajar en cualquier otra cadena solo disminuiría la
probabilidad de que su bloque minero termine en labloques
cadena de. Al utilizar PoStake en lugar de PoW, verificando si se
puede preguntar: ¿no impulsará ATMcash el desarrollo de tipos especializados de
almacenamiento que estén diseñados para la minería, y así terminen en la misma
posición
que Bitcoin a este respecto? Argumentamos que esto es poco probable; ver la Sección
IX.
uno puede extender una cadena es muy barato, y por lo tanto los mineros pueden
intentar extender muchas cadenas diferentes en paralelo. Esto impide
encontrar un consenso rápido, a diferencia de Bitcoin, donde todos losracionales se
mineросconcentran en la cadena más larga y por lo tanto siempre
crece más rápido que otros.

• **Minería de cadenas múltiples:**

Nuestro enfoque se basa en penalizar a los
mineros que trabajan en más de una rama. Es
importante desalentar a los mineros no solo de anunciar
bloques en cadenas múltiples, sino también de "probar"
muchas cadenas diferentes y elegir solo la mejor
para anunciar. Sugerimos tres soluciones variantes. En los
primeros dos esquemas, la calidad de bloque de un minero se fija para cualquier
paso de tiempo dado, de modo que probar muchas cadenas no produce
ningún beneficio; y luego penalizamos a los mineros que anuncian
bloques en cadenas múltiples. El tercer esquema tiene un
enfoque diferente e introduce la interacción entre los
mineros durante el proceso de minería, lo que permite la detección
de mineros que prueban muchas cadenas.

Finalmente, realizamos un análisis teórico de juegos de ATMcash
y descubrimos que tiene al menos propiedades de equilibrio tan fuertes
como Bitcoin. Modelamos el protocolo de ATMcash como unextenso
juego, y demostramos que los mineros no están incentivados a desviarse
de las reglas, siempre y cuando haya una mayoría honesta. Más
formalmente, demostramos que el protocolo es unsecuencialmente racional
equilibrio de Nash, que es el concepto de equilibrio estándar
para los juegos que ocurren en muchos pasos de tiempo. Los trabajos previos
relacionados con el equilibrio en Bitcoin han dado solo uninformal
tratamientodel problema: notablemente, [20] presenta unexhaustivo,
análisispero aún informal, de las estrategias de equilibrio en Bitcoin,
y concluye que la minería honesta es un equilibrio de Nash en
Bitcoin (si hay una mayoría honesta).

Contribución. En resumen, nuestra contribución es la siguiente:

• **Criptomoneda a partir de pruebas de capacidad** : ATMcash es una
criptomoneda basada puramente en pruebas de capacidad y, por lo tanto,
evita los principales inconvenientes de losbasados en la prueba de trabajo,
esquemas existentescomo se analiza en esta sección.

• **Abordar el problema de "nada en cuestión"**: proponemos
nuevos enfoques para los problemas conocidos de la molienda
y la explotación de cadenas múltiples enbasados en la no prueba de trabajo
sistemas. Nuestras soluciones también pueden extenderse a lade prueba de participación
configuracióndonde se encontraron estos problemas por primera vez.
• Evaluación de ATMcash y pruebas de capacidad: implementamos
una biblioteca de prueba de capacidad y un prototipo de
ATMcash. Se necesitan menos de 20 segundos para probar más de

1 TB de capacidad y una fracción de segundo para verificar.

• **Teoría de juegos de ATMcash**: nuestrosanálisis teórico de juegos modelos deATMcash como un juego extenso demuestra que la adhesión al protocolo es un equilibrio secuencial de Nash.

II. TRABAJOS RELACIONADOS

Pruebas de almacenamiento / capacidad de recuperación. Otros conceptos similares a las pruebas de capacidad son pruebas de almacenamiento y pruebas de capacidad de recuperación.
Estos son sistemas de prueba donde un verificador envía un archivo a un probador, y luego el probador puede convencer al verificador de que realmente almacenó o recibió el archivo. Demostrar que uno almacena un(aleatorio) archivociertamente muestra que uno dedica capacidad, pero estosprueba sistemas deno son pruebas de capacidad porque el verificador envía el archivo completo al comprobador, mientras que una propiedad importante de POC es que el cómputo del verificador (y así también comunicación) es como máximo poliglotámico en el tamaño de almacenamiento dedicado.

('Pruebas de borrado seguro' es Otro tipo de sistema de prueba que está relacionado con POC.)

III..PROOFS de capacidad

Como se discutió brevemente en la Sección I, el objetivo de la prueba de capacidad es que un probador demuestre a un verificador que está almacenando una cierta cantidad de capacidad. En esta sección, primero discutimos doshombre de paja enfoques deque no funcionan, y luego presentamos nuestra variante de POC para la configuración de criptomonedas.
A. Dos enfoques simples que no funcionan
Almacenar una tabla de funciones. Una "solución" tentadora es tener una función de aspecto aleatorio, ordenada por la salida. ¿Eldel probador desafíosería invertir la función en el valor f (x) para alguna x aleatoria? un prover honesto puede hacer esto en eltiempo registro demediante búsqueda binaria. Desafortunadamente, esto no funciona debido a compensaciones de tiempo / memoria, que permiten que untrampas probador dealmacene áspero aproximadamente ⅔ pares de entrada / salida y aún conecte la función a tiempo.
Almacenamiento de un archivo aleatorio. Otra idea simple sería

enviar bits (pseudo) aleatorios durante la inicialización, y
simplemente consultar de nuevo los subconjuntos aleatorios de estos bits durante la ejecución.
Sin embargo, esto requiere bits de comunicación, mientras que
un POC requiere que la eficiencia del verificador dependa de
algún parámetro de seguridad, pero debe ser básicamente independiente de,
y esta propiedad es crucial para todas las aplicaciones de
POC debatidas.

IV. VISIÓN GENERAL DE ATMcash

A.descripción de protocolo de alto nivel

Transacciones de. Las transacciones se realizan básicamente de manera idéntica
a Bitcoin: cada moneda "pertenece" a alguna clave pública pk.
La cadena de bloques actúa como un libro de contabilidad que realiza un seguimiento de qué monedas
pertenecen a qué claves (pero para evitar la molienda, proponemos
un nuevo diseño para la cadena de bloques en la Sección VI donde las
transacciones se desacoplan de las pruebas). Para transferir una moneda
de pk a pk0, una transacción que especifica esto debe estar firmada
por sk (la clave secreta para pk) y luego debe agregarse al bloque.
El nonce asegura que la misma capacidad no se puede usar para dosdiferentes
pruebas(esto se discutirá más adelante).

También agregamos transacciones especiales para inicializar mineros (trazado)
y un tipo especial de transacción que penaliza a un minero que
extiende dos cadenas diferentes utilizando la misma prueba de capacidad (que
aún no se ha implementado, pero existen métodos para verificar y
eliminar cadenas incorrectas.)

Incentivar la minería. Al igual que en Bitcoin, hay dos formas
de incentivar a los mineros a aportar recursos (capacidad de disco en
ATMcash, potencia de cálculo en Bitcoin).

(1) una recompensa por agregar bloques
(2) tarifas de transacción.

Recompensa: para agregar un bloque a la cadena, un minero recibe
algunas monedas recién acuñadas. El tamaño de la recompensa se especifica como parte
del protocolo y, por lo general, depende del índice del bloque.

En Bitcoin, la recompensa fue inicialmente de 50 Bitcoins, pero se reduce a la mitad cada

4 años, y actualmente está en 25.

En ATMcash, la recompensa es una curva lentamente decreciente que cambia un 10% por
mes, lo que permite un calendario de lanzamiento que no es tan drástico en cortar la recompensa a los mineros. Creemos que este es un enfoque más adecuado que el de Bitcoin y otras monedas con el uso 'reducción a la mitad'.

Tarifas de transacción: al generar una transacción, se puede
dedicar una cantidad (generalmente muy pequeña) de lastransferidas
monedasal minero que agrega la transacción a la cadena de bloques.
Inicializar minero Si un minero quiere aportar N bits de
capacidad al esfuerzo de extracción, ella toma muestras de un par de claves público / secreto (pk, sk) y ejecuta el procedimiento de inicialización de POC. (Al estar
en una configuración no interactiva, no hay un verificador para generar el
nonce único μ, así que simplemente usamos pk para esto.)

(?, S?): = Init (pk, N).

El minero almacena (S ?, sk) y genera una transacción especial
que solo contiene (pk,?). Una vez que esta transacción se encuentre en la
cadena de bloques, el minero puede empezar a extraer como se describe a continuación.
Minería. Los bloques se agregan a la cadena de bloques cadatiempo fijo
período de(por ejemplo, cada minuto), y requerimos que todas las partes tengan
un reloj que esté aproximadamente sincronizado. Para agregar un bloque en eltiempo
período dei, el minero recupera el valor hash del último bloque
en la mejor cadena hasta ahora (esta cadena tiene i - 1 bloques), y
también un desafío c. Esta c se usa como aleatoriedad para muestrearkp
desafíos decp y kcv challenges ccv.

Donde, dependiendo de cuál de las Pocapacity discutidas en la última sección
usamos, kp = O (1), kcv = log (n) o kp = O (1), kcv =? · Registro (n). Estos
desafíos se pueden muestrear utilizando primero c como semilla para generar unasuficiente
cantidadde aleatoriedad (rp, rcv): = hash (c) para que el algoritmo de muestreo de desafío
obtenga cp: = Desafío (n, kp, rp), ccv : = Desafío (n, kcv, rcv).

Cómo derivar el desafío c es la principal dificultad que enfrentamos.
En nuestra solución más simple, asumimos un faro impredecible
que difunde un nuevo valor aleatorio (o al menos impredecible)
del cual el desafío se deriva cada minuto (también
proponemos dos soluciones sin asumir un faro). El minero
luego calcula la respuesta POC desde cp:
a: = Respuesta (pk, S ?, cp).

Para dos pruebas válidas (pk,?, C, a) y (pk0,? 0, c0, a0) denotamos
con (recuerde que N? Es el tamaño de la capacidad cometida por?)
(A0, N? 0)? (a, N?)
que la prueba a es mejor que a0. Posponemos la discusión
sobre la definición precisa de este orden a la Sección VB. Por
ahora, solo mencionamos que el orden debería satisfacer
Pr [(a0, N0)? (a, N)] = NN + N0
.
Es decir, la probabilidad de que un triunfo sea proporcional a su
fracción de la capacidad total. La probabilidad se toma sobre la
elección del oráculo aleatorio utilizado para calcular la calidad de la prueba.
Si la respuesta encontrada por un minero es tan buena que existe una
posibilidad real de que sea la mejor respuesta encontrada por cualquier minero,
el minero crea un bloque y lo envía a la red con la
esperanza de que termine en el cadena. Un bloque debe contener
transacciones, la prueba POC a y también elcompromiso
resultado de verificación decalculado como acv: = Answer (pk, S ?, ccv).
Tenga en cuenta que la verificación del compromiso no necesita ejecutarse a
menos que el minero haya encontrado una prueba excepcionalmente buena: por lo tanto,
el cálculo de la gran mayoría de los mineros en la red
será muy bajo, ya que solo necesitan verificar la calidad de sus
pruebas, y la mayoría de ellos no procederá con la verificación.
Durante el resto del período de un minuto, el minero
no necesita hacer nada. Como la minería solo requiere una pequeña cantidad
de trabajo (computación, comunicación y acceso aleatorio al
almacenamiento) en cada período de tiempo, puede ejecutarse en cualquier
computadora
que tenga alguna capacidad de disco libre y esté conectada a Internet,
sin incurrir en una desaceleración notable. .

Calidad de una prueba de Pocapacity Considere algunas pruebas válidas
(pk1,? 1, c1, a1),. . . , (pkm,? m, cm, am) para la capacidad s del,.
tamaño N1. . , Nm. Queremos asignar una calidad a una prueba
(que solo será una función de ai y Ni), de modo que la
probabilidad (sobre la elección del hash de oráculo aleatorio) de que
la i-prueba tenga la mejor "calidad" corresponda a su fracción
de la capacidad total, es decir
Pr hash [? j 6 = i: (aj, Nj)? (ai, Ni)] = P Ni mj = 1 Nj
.
Observamos que para lograr esto, es suficiente
lograr esto para cualquier par de compromisos, es decir,
Pr hash [(aj, Nj)? (ai, Ni)] = Ni Ni + Nj

Si todos los Ni fueran del mismo tamaño N, podríamos simplemente definir
(aj, N)? (ai, N)? hash (aj) = hash (ai)
Es decir, asignamos cada ai a un valor aleatorio hash (ai), y el
valor que sea mayor gana. Queremos permitir diferentes
valores de Ni, por lo que los mineros que quieran contribuir con la capacidad N0 solo necesitan un compromiso de capacidad, y no tienen que dividirlo en compromisos de capacidad N0 / N de tamaño N, y luego ejecutar una prueba para cada fragmento por separado .

Para esto, definimos una distribución DN, N? N, que se
define al muestrear N valores en [0, 1] al azar, y luego
generar el mayor de ellos.

DN ~ max {r1,. . . , rN: ri? [0, 1], i = 1,. . . , N} (2)
Con DN (t) denotamos una muestra de DN (o más bien, una distribución
que está muy cerca de ella) usando la aleatoriedad t para muestrear.
Ahora decimos que (ai, Ni) es de mayor calidad que (aj, Nj)
if (aj, Nj)? (ai, Ni)? DNj (hash (aj)) = DNi (hash (ai)).

Queda por mostrar cómo muestrear de manera eficiente desde el
DN de distribución para un N. dado. Recuerde que si FX denota
la función de distribución acumulativa (CDF) de algunaaleatoria
variableX sobre [0, 1] y existe el inverso F-1 X, entonces F-1 X (U)
para U uniforme sobre [0, 1] tiene la misma distribución que X. La
variable aleatoria X muestreada según la distribución DN
tiene CDF FX (z) = zN, ya que esta es la probabilidad de que todos los N
los valores ri considerados en (2) terminan por debajo de z (y por lo tanto
también su máximo). Por lo tanto, si queremos muestrear desde
el DN de distribución, podemos simplemente muestrear F-1X (U) para U
uniforme sobre [0, 1], que es U1 / N. En queremos muestrear
DNi usando aleatoriedad hash (ai), y hash salidas de cadenas de bits en
{0, 1} 256 en lugar de valores en [0, 1], entonces tenemos que normalizar:
DNi (hash (ai)): = hash (a) / 2 2561 / N

Tenga en cuenta que esto introduce una pequeña imprecisión debido al hecho de
que hash (a) / 2 256 es uniforme en un conjunto discreto en lugar del
intervalo continuo [0, 1], pero esto puede ser seguro ignorado.
Observación. Observe que la función de calidad descrita anteriormente tiene
la propiedad de que la calidad del bloqueo que un minero dado pk
puede producir en un paso de tiempo determinado es fija, independientemente de la cadena que elija extender. Esta propiedad será importante
para evitar el ataque de "explotación de múltiples cadenas" que se
describió en la Sección I, como se explica en la siguiente sección.

VI. EL FORMATO DE BLOQUEO

Una cadena de bloque es una secuencia de bloques ß0, ß1,. . . Cada bloque ßi = (fi, si, ti) es creado por un minero y consta de tres partes principales, que llamamos "subbloques". Cada subbloque comienza con el índice i que especifica su posición en la cadena de bloques. A continuación, describimos los componentes restantes de las tres sub-bloques de un SSI bloque, i> 0. Los bloquesgénesis ß0necesariamente tiene un formato un tanto diferente, ya que no puede depender de los bloques anteriores:

• El HASH sub-bloque fi contiene:
- Un hash hash de 256 bits (fi-1) del subbloque HASH del bloque anterior de la cadena.
- Una "prueba de capacidad" que contiene la identidad del minero pk (más detalles sobre esto se dan a continuación).

• El subbloque TRANSACTION ti contiene:
- Una lista de transacciones (definida con más detalle a continuación).

• El sub-bloque SIGNATURE si contiene:
- Signo de firma del minero (sk, ti) en elTRANSACTION subbloqueti asociado con este bloque.
- Signo de firma del minero (sk, si-1) en elSIGNATURE subbloquesi-1 asociado con el bloque anterior.

Los enlaces entre bloques consecutivos en el blockchain se ilustran en la Figura 2. También nos referiremos a los subbloques hash como la cadena de prueba, y los subbloques de firma con las transacciones como la cadena de firma. Las flechas sólidas representan hashes y las flechas punteadas representan firmas. Observe que, si bien los subbloques de firma y transacción están todos vinculados entre sí, los subbloques hash solo están vinculados entre sí y no a ningún subbloque de firma o transacción.

A. Solución al problema de la molienda:

Desacoplando pruebas de transacciones conseguimos seguridad contra la molienda: para cualquier compromiso de capacidad (pk,?), El minero pk no puede generar dos (o más) bloques de hash formateados correctamente para ser agregados a la cadena de prueba .

La cadena de firma vincula las transacciones a laprueba
cadena de. Si un minero honesto (honesto para definir más abajo) agrega
el i-ésimo bloque, las transacciones correspondientes a esta prueba
Para demostrar esto, se requiere que sea computacionalmente difícil encontrar dos
transcripciones de aceptación distintas para el mismo desafío. El POC de
satisface esta propiedad (encontrar dos transcripciones de aceptación en sus esquemas
equivale a romper la resistencia a la colisión de la función hash subyacente).
La cadena para bloquear no puede cambiarse más, incluso si un
adversario controla todas las claves secretas de los mineros que agregaron el
primero: 1 bloque. El hecho de que el minero sea honesto significa que
solo firma un solo bloque de transacciones utilizando la clave secreta
sk correspondiente a su identidad pk, y además lo mantiene en
secreto. Para ver esto, tenga en cuenta que si queremos cambiar las
transacciones en el bloque j <i manteniendo lapruebas actual
cadena depara bloquear i, entonces las firmas para los bloques j,. . . ,
debo volver a calcularlo, lo que requiere sk.

B. Transacciones ATMcash se basa en un esquema seguro de 12 firmas

S = (SigParamGen, SigKeyGen, Sign, SigVerify)
y un protocolo POC? = (Init, Challenge, Answer, Verify).

A continuación, especificamos los tres tipos de transacciones
(para pagos, compromisos de capacidad y castigos) que
permitimos en ATMcash. Pagos. Las partes identificadassostienen y transfieren las monedas
por una clave de verificaciónen apoyo de SigKeyGen.

Una transacción transfiere monedas de benefactores
a n beneficiarios y tiene el formctx = (pago, txId, in, ~ out ~).

Para que una transacción se considere válida, selas
deben cumplirsiguientes condiciones:

• txId: un identificador de transacción exclusivo y arbitrario. Es decir,
no hay dos transacciones en una cadena de bloques que puedan tener el mismo
Identificador.

• out ~: una lista de beneficiarios y la cantidad que reciben.
Específicamente, out ~ = (out1, ..., outm) con outi = (pki, vi), donde:
- pki está en el soporte de SigKeyGen y especifica un
beneficiario, y - viis es el número de monedas que pki
debe ser pagado.

• Una lista de monedas de entrada a la transacción. Específicamente,

en ~ = (in1, ..., inn), una lista de n benefactores, cada uno compuesto por un triple: inj = (txIdj, kj, sigj), donde:

- txIdj es el identificador de una transacción pasada

- kj es un índice que especifica un beneficiario particular pkkj de la transacción txIdj

- sigj es una firma de (txId, txIdj, kj, out ~), que verifica bajo clave pkkj probar la propiedad del beneficiario de la transacción txId y vincular la moneda al beneficiarios.

En Bitcoin, la especificación de pagos es más general: en

lugar de especificar beneficiarios a través de sus claves de verificación, los destinatarios se especifican escribiendo una secuencia de comandos en un lenguaje de scripting especial (no Turing-completo) llamado Bitcoin Script. Las monedas de salida de una transacción pueden ser canjeadas por cualquier parte que pueda producir entradas que "satisfagan" la secuencia de comandos. En la práctica, ATMcash se puede modificar de forma directa para acomodar tal scripting; pero en este trabajo, para mayor claridad, suponemos que cada destinatario de pago se especifica con una clave de verificación.

Ese es el kj th beneficiario de la transacción txIdj es el j-ésimo benefactor de la transacción txId.

txId está firmado para evitar la maleabilidad de la transacción https: //en.bitcoin. it / wiki / Transaction Malleability

1) No se hace referencia a ningún benefactor por más de una transacción en el blockchain (para evitar el doble gasto).

2) La suma de los valores de entrada a la transacción (es decir, la suma de las cantidades proporcionadas por cada benefactor) es al menos la suma de los importes pagados a los beneficiarios.

Tenga en cuenta que algunas de las identidades beneficiarias pueden pertenecer al creador de la transacción, que puede transferir dinero a sí mismo como "cambio": por ejemplo, si la suma de los valores de entrada excede el monto total del pago que desea transferir a otras partes.

VII. INSTANTIACIÓN (trazado)

En esta sección describimos los pasos concretos requeridos para
configurar, extraer y pagar en ATMcash.la
Proporcionamosinstanciación para el segundo esquema (desafío del pasado),
delineado en la Sección VC. El primer esquema (baliza aleatoria) es
casi idéntico, excepto que el desafío c se deriva de la
baliza aleatoria (y no mediante el hash de un bloque de la cadena).
16? 0 es un Merkle-hash de todas las etiquetas en un gráfico difícil de guijarro. Podemos
cambiar? 0 a otro valor simplemente cambiando una sola etiqueta, que no
se notará en la fase de ejecución a menos queesta etiqueta particular con sus hijos
se solicite.

¿En la configuración tenemos que arreglar el parámetro de seguridad? para
ser utilizado para la firma y el esquema POC. Además,
debemos especificar parámetros y funciones:

• ¿tiempo? N especifica la longitud de un intervalo de tiempo en minutos.
Debeser suficientemente mayor que elde propagación de la
tiempored,pero por lo demás lo más pequeño posible. tiempo = 1 parece
ser una elección razonable aquí.

• d? N especifica que el desafío para el bloque i es una función
del bloque i - d. Un valor razonable es d = 120.

• La recompensa es una función tal que Reward (i) especifica la
cantidad de monedas que obtiene un minero para extraer el i-
ésimo bloque.

• La calidad es una función que toma como entrada un compromiso de capacidad
(pk,?) Para la capacidad del tamaño N junto con un
par desafío / respuesta (c, a). Si Verify (pk,?, C, a) 6 = 1
(es decir, no es una transcripción de prueba POC válida), la
función de calidad genera -8. De lo contrario, la salida es
(con DN como se define en la Sección VB):

Calidad (pk,?, C, a) = DN (hash (a)).
Para decidir cuál de las dos cadenas de pruebas dadas es la
"mejor", también necesitamos definir la calidad de una cadena de pruebas
f0,. . . , fi, que denotaremos con QualityPC (f0, ..., fi).
Cada bloque hash fj contiene una prueba (pkj,? J, cj, aj), y dejamos

que vj = DNj (aj) denote la calidad de la j-ésima prueba en la
cadena. Para cualquier calidad v? [0, 1], denotamos con N (v) = min {N? N: Pr [v? w | w? DN] = 1/2}
la capacidad requerida para obtener una mejor prueba que v en unaaleatoria
pruebacon probabilidad 1/2.

Tenga en cuenta que N (vj) generalmente estará alrededor del almacenamiento total de todos los mineros que estaban activos
cuando se extrajo el bloque. Con esta definición, unanatural
medidapara la calidad de la cadena sería simplemente la suma17
Pi j = 1 N (vj).

El problema con esta medida es que si un
minero encuentra una prueba extremadamente buena, digamos que N (v) es 1000 veces
más grande que el almacenamiento total (esto sucederá aproximadamente cada
1000 bloques), entonces el minero podría retener su prueba, y
1000 bloques más tarde generará una bifurcación usando esta prueba seguida
de 999 pruebas arbitrariamente malas para los bloques restantes. Para
evitar tales bifurcaciones profundas, tapamos pruebas que son demasiado buenas
diciendo que vj no puede contribuir más a la suma que, digamos 10
veces la mediana de los últimos 101 bloques (la mediana da una
buena aproximación de la capacidad total dedicada a
minería).

Formalmente, supongamos que N (vj) se define recursivamente como
N (vj) = max {N (vj), 10 · mediana (N (vj-101),.., N (vj-1)}

Otra razón por la que se define el calidad simplemente como
Pi j = 1 N (vj) es problemática, es que la capacidad total contribuida
puede aumentar drásticamente con el tiempo. En este caso, para
llegar a una cadena cuya calidad es mejor que la calidad
de la cadena real, es suficiente para dedicar mucho menos que la
capacidad total que actualmente se dedica a la minería. Por esta
razón, solo tomamos los últimos 1000 bloques en cuenta al
calcular la calidad:

QualityPC (f0, ..., fi) = X j = max. {1, i-1000} N (vj)

Comenzamos sumando con j = 1, no j = 0, ya que el bloque de génesis (aún por
definir) no contendrá una prueba.

Finalmente, un bloque de génesis ß0 = (f0, s0, t0) se genera y
publica, tiene un formato diferente de otros bloques. El

bloque de transacciones contiene solo un compromiso de capacidad t0 =
(commit, txId, (pk0,? 0)), el bloque hash f0 contiene solo
una cadena aleatoria , 18 y el bloque de firma s0 contiene
el signo de firma (sk0, t0) del bloque de transacciones (pero no
del bloque de firma anterior, ya que no hay ninguno).

Inicializa la minería. Para dedicar N bits de almacenamiento para la
minería, una parte genera una identidad y un compromiso de capacidad
(pk, sk)? SigKeyGen, (?, S?): = Init (pk, N).

Almacena S? (de tamaño N) y sk localmente. El minero luego genera y
publica una transacción ctx = (commit, txId, (pk,?)). Una vez que ctx se ha
agregado como una transacción a la cadena de hash, el minero puede comenzar a minar como
se describe a continuación.

Mining:

As we enter time slot i, the miner retrieves the
so-far-best blockchain ß0, . . . , ßi-1 (that is, the chain maximizing
Quality PC(f0, . . . , fi-1). We assume that the miner
“honestly” stores capacity S? and the corresponding commitment
(pk, ?) has been added to some transcription block tj , j = i-1
in this chain.

Next, the miner computes the randomness for the
challenge sampling by hashing the hash block that is d blocks
in the past c := hash(pk, fi-d).

From this c we then compute the challenges cp, ccv. The miner
computes the POC answer a := Answer(pk, S?, cp) .

If q := Quality(pk, ?, c, a) is very high, so it has a realistic
chance to end up as the best answer of the entire network, the
miner generates a hash block fi = (i, hash(fi-1), pi), where
Pi is 20 (pk, ?, c, j, q, a, acv), where acv := Answer(pk, S?, ccv) .
is the output of the commitment verification (for efficiency reasons we
only execute commitmentverification at this point).

Then the miner retrieves transactions (typically, giving
priority to the ones paying the highest fees), checks
their corr ectness, and adds the valid ones to a transaction
Block.

. It then computes the signature block si =
(Sign(sk, si-1), Sign(sk, ti)) and publishes block ßi =(fi, si
, ti), hoping that it will end up in the blockchain,earning the
miner Reward(i) coins, plus the transactions fees of the
transactions in the block.

Transaction. Any party can generate a transaction and publish
it. If it is correctly generated, it should ultimately end up
in the blockchain. We have already described the format and
semantics of the three types of transactions.

Or better, some kind of timestamp like a sentence from a newspaper of
the day, as is done in Bitcoin, to show that the genesis block was not generated
before some date “publishes” and “retrieves” we mean that a party sends or downloads
something from the network. Typically, there would be some servers that
organize the data, ie, keep track of the best chains and collect transactions,
so a miner would only interact with one or a few such servers it trusts.

To evaluate ATMcash, we have implemented a prototype
in Go, using SHA3 in 256-bit mode as the hash function. The
prototype uses the graphs from [24], and forces a cheating
prover to store at least ?(N/ log(N)) bits in order to effi-
ciently generate proofs. Given that the network infrastructure
is very similar to Bitcoin, we are mainly interested in three
quantities: time to initialize the capacity (graph), size of the proof,
and time to generate and verify the proof. The experiments
were conducted on a server equipped with an Intel i5-4690K
Haswell CPU and 8GB of memory. We used an off-the-shelf
hard disk drive, with 2TB of capacity and 64 MB of cache.
Our proof-of-capacity library and ATMcash prototype are
available at: https://github.com/kwonalbert/ATMcash.
Time to Initialize.

To start mining ATMcash, the clients must first initialize their capacity.
This involves computing all the hashes of the nodes, and
computing the Merkle tree over the hashes (plotting). In Figure 3,
we show the initialization time for capacity s of size 8 KB to
1.3 TB. As expected the time to initialize grows linearly with
the size of the capacity ; at 1.3 TB, it takes approximately 41
hours to commit the graph. While expensive, we note that this
procedure is done only once when the miner first joins the
ATMcash network, and will use the initialized capacity over and
over again. In fact, we require capacity initialization to non-trivial

time, because an extremely fast capacity initialization would make re-using the same capacity for different commitments a viable strategy (Section VC).

Size of the Proof. A proof (ie, a full solution to the puzzle) in ATMcash consists of the hashes of the challenge nodes and their parents, in the Merkle inclusion proofs

Time to Generate/Verify the Proof. Unlike Bitcoin, generating an answer for a puzzle (ie, generating a proof-of-capacity) takes little time.

In Bitcoin, the miner is expected to work through most of the epoch in an attempt to find a preimage of a hash with sufficient difficulty. In ATMcash, assuming a mineris storing the capacity correctly, the miner needs to only perform (1) lookups in the disk to find their solution which takes fraction of a second.
For instance, it takes < 1 msto read a single hash from the disk.
Only if the miner believes its answer is of very good quality will it generate the full proof, but even this takes seconds, not minutes.

As outlined above, our proofs are substantially bigger than Bitcoin's, and require more than just one hash evaluation to verify. However, for an active currency, we can still expect the size and verification time for the proofs added with every block to be marginal compared to the size of the transaction added with every block, and the time required to verify that the transactions are consistent. This indeed shows that though it may take seconds to generate the proof, verification takes a fraction of a second.

Energy:

Though our prototype was evaluated using a full CPU which wastes a lot of energy, one could in principle run the prover and the verifier on a energy-efficient devices such as Raspberry Pi [3]. An efficient microcontroller consumes less than 10 W of power, and most miners will only open one node per time-step since the quality of their answers will likely be bad. To get an upper bound on the power requirement, let us assume that there are 100,000 miners, each with 1 TB of capacity , and about 1% of the miners mine “good” answers which they will want to generate a full answer. Then we have

$10W \cdot 100000 \cdot 0.01s + 10W \cdot 1000 \cdot 20s = 210000J/block$
which translates to 210 kJ/min if we add one block a minute. In contrast, Bitcoin on average uses 100 MW, so it consumes 6 GJ/min, which is several orders of magnitude larger. We note that this 1% figure is a very conservative bound, so the difference could be even larger in practice.

GAME THEORY OF ATMcash

The miners in a cryptocurrency are strategic agents who seek to maximize the reward that they get for mining blocks. As such, it is a crucial property of a cryptocurrency that "following the rules" is an equilibrium strategy: in other words, it is important that the protocol rules are designed in such a way that miners never find themselves in a situation where "cheating" and deviating from the rules yields more expected profit than mining honestly.

Intuitively, ATMcash mining is modeled by the following n-player strategic game. Game-play occurs over a series of discrete time steps, each of which corresponds to a block being added to the blockchain. At each time step, each player (miner) must choose a strategy, specified by:

• which blocks to extend (if any), which transactions to include in the new blocks, and
• which extended blocks to publish (if any).
We present the details of our game-theoretic analysis in the unpredictable-beacon model, and remark that the analysis can be extended to cover the other models too.

A. Game-theoretic preliminaries

The standard game-theoretic notion for a strategic game which occurs over multiple time steps (rather than in "one shot") is the extensive game. In order to accurately model the probabilistic aspects of the ATMcash protocol (eg the unpredictable beacon), we consider extensive games with chancemoves: this is the standard game-theoretic notion to capture extensive games which involve exogenous uncertainty. The uncertainty is modeled by an additional player called Chance which behaves according to a known probability distribution. In the ATMcash setting, every player (including Chance) makes an action at every time step. A player's action consists

of choosing whether and how to extend the blockchain, and the action of Chance determines the value of the unpredictable beacon for the next time step.

An extensive game is commonly visualized as a game tree, with the root node representing the start of the game. Each node represents a state of the game, and the outward edges from any given node represent the actions that players can take at that node. Leaf nodes represent terminal states: once a leaf is reached, the game is over. In accordance with the literature, we refer to paths in the game tree (starting at the root) as histories; and histories which end at a leaf node are called terminal histories.

Definition X.1 (Extensive game). An extensive game G =hN, H, fC, I~, ~ui is defined by:

- [N], a finite set of players.
- H, the set of all possible histories, which must satisfy the following two properties:

– the empty sequence () is in H, and

– if (a1, . . . , aK) ? H then for all L = K, it holds that (a1, . . . , aL) ? H.

We write Z ? H to denote the subset consisting of all terminal histories. For any history h,A(h) = {a : (h, a) ? H} = ×i?[N]Ai(h) denotes the set of action profiles that can occur at that history, and Ai(h) denotes the set of actions that are available to player i at history h.

- f(·, h) is a probability measure on AC(h), where h ? H and C denotes the Chance player.

- I~ = (I1, . . . , IN), where each Ii is a partition of H into disjoint information sets, such that Ai(h) = Ai(h0) whenever h and h0 are in the same information set I ? Ii. Let Ai(I) denote the set of actions that are available to player i at any history in information set I.

- ~u = (u1, . . . , uN), where each ui: Z ? R is the utility function of player i.

Imperfect information and information sets. An extensive game is said to have perfect information if at any point during game-play, every player is perfectly informed of all actions taken so far by every other player. In the context of ATMcash, players are only aware of each others' announced actions: for example, if Alice tries extending several blocks and then only announces one of them, then Bob does not know about the other blocks that Alice tried to extend. Thus, ATMcash is a game of imperfect information.

The information that players do not know about other players' actions is modeled by the partitions I~ = (I1, . . . , IN) in Definition X.1. Each Ii is a partition of H into disjoint information sets, and for each i ? [N] and any pair of histories h, h0 ? I in a particular information set I ? Ii, player i cannot tell the difference between game-play at h and at h0.

Example X.2 (“Match my number” game).

Consider a simple two-player game in two rounds: in the first round, player 1 chooses a number a ? {0, 1, 2}. In the second round, player 2 chooses a number b ? {0, 1, 2}. Player 2 wins if b = a, and player 1 wins otherwise. Clearly, player 2 can always win if he knows a.

However, we consider a game of imperfect information where player 2 must choose b without knowing a: in particular, suppose player 2 only learns whether a = 0. Then, the histories (a = 1) and (a = 2) are in the same information set in the partition A strategy of a player in an extensive game is defined by specifying how the player decides his next move at any given history. In games of imperfect information, the player may not know which history he is at, so we instead specify how the player decides his next move at any information set.

Definition X.3 (Strategy profile).

A strategy profile ~a =(a1, . . . , aN) of an extensive game G = hN, H, fC, I~, ~ui specifies for each player i ? [N] and each information set I ? Ii a probability distribution ai(I) over the action set Ai(I). We say that ai is the strategy of player i.

Let I(h) denote the information set in which history h lies.

The probability that a history h occurs under strategy profile a is denoted by Pr~a[h], and the probability that a history h0 occurs given that h occurred is denoted by Pr~a[h0 |h].

Recall that the utility functions u1, . . . , uN were originally defined on inputs in Z, the set of terminal histories. For each ? [N], we now define ui(~a) to be the expected utility of player i given the strategy profile ~a. That is,ui(~a) = Xh ?Z ui(h) · Pr ~a[h].

Moreover, we define ui(~a|h) to be the expected utility of player i given ~a and given that history h has already occurred. That is,ui(~a|h) = X h0?Z ui(h 0) · Pr ~a[h 0 |h].

Equilibrium notions. The most widely known equilibrium concept for a strategic game is the Nash equilibrium [23], given in Definition X.4. Intuitively, in a Nash equilibrium, each player's strategy is a best response to the strategies of the other players.

For a strategy profile ~a, we write ~ai to denote (aj)j?N,j6=i, that is, the profile of strategies of all players other than i; and we use (a 0 i, ~ai) to denote the action profile where player i's strategy is a 0 i and all other players' actions are as in ~a.
Definition X.4 (Nash equilibrium of an extensive game). Let G = hN, H, f, I~, ~ui be an extensive game. A strategy profile ~a is a Nash equilibrium of G if for every player i ? [N] and every strategy a 0 i of player i, ui(~a) = ui(a 0 i, ~ai).

The Nash equilibrium concept was originally formulated for one-shot games, and it is known to have some shortcomings in the setting of extensive games. Informally, the Nash equilibrium does not account for the possibility of players changing their strategy partway through the game: in particular, there exist Nash equilibria that are not “stable” in the sense that given the ability to change strategies during the game, no rational player would stick with his equilibrium strategy all the way to the end of the game.

Example X.5 (“Unstable” game). Consider a simple two player game in two rounds: in the first round, player 1 chooses either strategy A or B. In the second round, player 2 chooses

either strategy C or D. The game tree is given below, where the notation (x, y) at the leaves denotes that player 1 gets payoff x and player 2 gets payoff y if that leaf is reached.

To address these shortcomings of the Nash equilibrium concept for extensive games, an alternative (stronger) notion has been proposed: the sequentially rational Nash equilibrium. This stronger concept ensures that players are making the best decision possible at any point during game-play. In a game with imperfect information, it is necessary to consider not only the strategy profile, but the players' beliefs at any point in time about how game-play arrived at the current information set. A strategy profile which takes into account players' beliefs is called an assessment.

Definition X.6 (Assessment). An assessment in an extensive game is a pair (~a, ~μ) where ~a = (a1, . . . , aN) is a strategy profile and ~μ = (μ1, . . . , μN) is a belief system, in which each μi is a function that assigns to every information set in Ii a probability measure on histories in the information set.

In Definition X.6, μi(I)(h) represents the probability that player i assigns to the history h ? I having occurred, conditioned on the information set I ? Ii having been reached. For each i ? [N], we now define ui((~a, ~μ)|I) to be the expected utility of player i at the information set I ? Ii, given the strategy profile ~a and belief system ~μ. That is, ui((~a, ~μ)|I) = X h?I ui(~a|h) · μ(I)(h).

We write ui((~a, ~μ)) to denote ui((~a, ~μ)|{()}), that is, the expected utility for player i at the beginning of the game. An assessment (a, μ) is said to be sequentially rational if for every i ? [N] and every information set I ? Ii, the strategy of player i is a best response to the other players' strategies, given i's beliefs at I. A formal definition follows.

Definition X.7 (Sequentially rational assessment). Let G = hN, H, f, I~, ~ui be an extensive game. An assessment (~a, ~μ) is sequentially rational if for every i ? [N] and every strategy A 0 i of player i, for every information set I ? Ii , it holds that ui((~a, ~μ)|I) = ui(((a 0i, ~ai), ~μ)|I).

Definition X.7 almost fully captures the idea players should be making the best decision possible given their beliefs at any

point during game-play. To fully characterize a sequentially rational Nash equilibrium, we require additionally that the beliefs of the players be consistent with ~a. For example, if an event occurs with zero probability in ~a, then we require that the players also believe that it will occur with zero probability.

Definition X.8 (Consistent assessment).

Let G = hN, H, f, I~, ~ui be an extensive game. A strategy profile ~a is said to be completely mixed if it assigns positive probability to every action at every information set. An assessment (~a, ~μ) is consistent if there is a sequence ((~an, ~μn))n?N of assignments that converges to (~a, ~μ) in Euclidean capacity , where each ~an is completely mixed and each belief system ~μ n is derived from ~a
n using Bayes' rule.

Finally, we arrive at the definition of a sequentially rational Nash equilibrium.

Definition X.9 (Sequentially rational Nash equilibrium). An assessment is a sequentially rational Nash equilibrium if it is sequentially rational and consistent.

B. Game-theoretic analysis of ATMcash

In order to analyze the game-theoretic properties of ATMcash mining, we define an extensive game, ATMcashGame, which models the actions that miners can take, and the associated payoffs. To facilitate analysis, we simplify the action capacity of the game as much as possible while still accurately modeling the incentives of ATMcash miners. Concretely:

• We do not include the action of creating a capacity commitment because (as discussed in Section VA under “Mining”) we can assume that rational miners will commit to all the capacity they have, and nothing else.

• We do not include the action of creating transactions because such actions do not affect the rewards that players receive from mining blocks, except in the case of punishment transactions. To deal with the case of punishment transactions, we define the payoff of a player

who mines multiple blocks in the same time step to be zero. This payoff function exactly captures that of a miner in the actual ATMcash protocol, because it is a dominant strategy for each other miner to create a punishment transaction (including a positive transaction fee) if she sees that a cheating player has mined multiple blocks in a time step, and hence we can assume that the cheating player will surely be punished at a later point in the protocol. Since the punishment penalizes the cheating player by the amount of the mining reward, it follows that the cheater's overall utility for the time step in which he cheated is zero.

• We do not explicitly model the amount of capacity that each player has. Instead, we study the two critical cases: in our initial analysis, we assume that no miner controls more than 50% of the capacity committed by active miners. Then, we discuss potential issues that arise if a miner does control a majority of the capacity.

Let B denote set of all blocks as defined in Section VI. For any number of players N ? N, any number of time steps K ? N, and any reward function ?: N ? N, we define the extensive game

ATMcashGame?,K,? = hN, H, fC, I~, ~ui as follows:

• The set H of histories is defined inductively as follows:

– The action set of the Chance player AC(h) ={0, 1}
m is the same for every history h.
– The empty sequence () is in H, and Ai(()) =
{(Ø, Ø)} for each i ? [N].

– Let h = (h 0, a) be any non-terminal history where the latest action profile a = (a1, . . . , aN , aC) consists of the actions of each player in [N] ? {C} at Later in this section, we address what happens if a miner gains additional capacity (or loses some capacity) during the game.

We remark that the standard way to model this would be to assign a type to each player, representing how much capacity he has.

history h 0, and for each player i ? [N], the action ai = (Si, Ti)
is a pair of sets. Then for any i ? [N], the action set Ai(h) of player i at h is

Ai(h) = P(T) × B where T =[i?[N] Ti An action ai = (Si, Ti) can be interpreted as Follows:

The set of blocks from the previous time step which player i attempts to extend in this time step, and Tiis the set of extended blocks which player i announces in this time step.

• The probability measure f(·, h) is uniform over {0, 1} m.
• For each i ? [N], we define the partition Ii by an equivalence relation ~i

. The equivalence relation ~i is defined inductively as follows (we write [h]i to denote the equivalence class of h under ~i):

– [()]i = {()}, that is, the empty sequence is equivalent only to itself.

– [(h,((S1, T1), . . . ,(SN , TN), aC))]i = {(h 0,((S 0 1, T0 1), . . . ,(S 0 N , T0 N), a0 C)) ? H : h ~ih 0 ? Si = S 0 i ? Ti = T 0i ? aC = a 0 C ??j 6= i, Tj = T 0 j

- where h and h 0 are histories and the pairs (Sj , Tj) and (S 0 j , T0 j) are actions of player j. That is, two histories are equivalent under ~i if they are identical except in the "first components" Sj of the actions (Sj , Tj) taken by the players other than i.

• ~u = (u1, . . . , uN), where each ui : Z ? R is defined as described below. For a history h, let beac(h) denote the sequence of actions taken by the Chance player in h, and let beacj (h) denote the jth action taken by the Chance player in h. For a block B, let Bc denote the challenge
We define Quality(B, c) = (Quality(B) if Bc = c0 otherwise.

Similarly, let QualityPC((B1, . . . , BL),(c1, . . . , cL)) = (QualityPC((B1, . . . , BL)) if ?i ? [L], Bi .c = ci 0 otherwise.

Let blocks(h) denote the sequence of "winning blocks" at each time step in the game, defined inductively:

– blocks(()) = () – blocks(h = (h 0 ,((S1, T1), . . . ,(SN , TN), aC))) = arg maxB?T (Quality(B, beac|h|(h))), where T = ?i?[N]Ti

Let blocksj (h) denote the jth block in the blockchain.

We assume that the winning block is unique at each time
Let winners(h) denote the sequence of players who announce

the winning block at each time step in the game, defined inductively as follows:

This can be achieved by breaking ties between blocks in an arbitrary way.

Note that it is not possible for two different players to announce exactly the same (valid) block, because each block contains the miner's identity.

– winners(()) = () – winners(h = (h 0 ,((S1, T1), . . . ,(SN , TN), aC))) = arg maxi?[N] maxB?Ti (Quality(B, beac|h|(h))).

Let winnersj (h) denote the jth winner in the sequence winners(h). Let only one j(i, h) be an indicator variable for the event that player i's jth action (Si, Ti) in the history h does not mine multiple blocks, ie |Ti | = 1.

Finally, the players' utility functions are defined as follows: for a terminal history h of length K, ui(h) = X j?[K] di,winnersj (h) ·onlyonej (i, h)·?(blocksj (h)), where di,j is the Kronecker delta function26. That is, a player's utility is the sum of the rewards he has received for announcing a winning block (in the time steps where he has announced at most one block).

By Definition X.10, for any i ? [N], for any histories h, h0 in the same information set I ? Ii, it holds that blocks(h) = blocks(h 0).
Thus, we can associate a unique blockchain with each information set: we define blocks(I) to be equal to blocks(h) for any h ? I. Similarly, beac(h) = beac(h 0) for any h, h0 ? I in the same information set I, so we define beac(I) to be equal to beac(h) for any h ?For a block B ? B and a challenge c ? Challenge, wedefine Extendi(B, c) to be the block generated by player i when mining the next block after B using the POC challenge c (see Section VII for exact block format).

Theorem X.11.

For any number of players N, any number of time steps K ? N, and any reward function ? : N ? N, let ~a = (a1, . . . , an) be a pure strategy profile of ATMcashGame?,K,?, defined as follows: for each i ? [N], for any information set I ? Ii such that I 6= {()}, ai(I) (({blocksj (I)} , {Extendi(blocksj (I), beacj (I))})) = 1, where j = 1 is the length of the histories in information set

27. That is, player i's next action at information set I is
aˆi = ({blocksj (I)} , {Extendi(blocksj (I), beacj (I))}).

Then ~a is a Nash equilibrium of ATMcashGame?,K,?.
Proof. Take any player i ? [N]. By the definition of Extend,
for any information set I ? Ii with I 6= {()}, the quality v of the extended blockchain
v = QualityPC((blocks(I), Extendi(B, beacj (I))), beac(I))
is the same for any block B which was announced at time step
j. Therefore, no utility can be gained by choosing any block B
over any other block B0 to extend: that is, ui(~a) = ui(a 0 i, ~ai)
for any strategy a 0 i which distributes probability over actions
of the form (S, T) where |S| = 1.

Moreover, not extending any block or extending multiple
blocks precludes a player from being the “winner” and receiving
the reward in this time step, so extending a block is preferable
to not extending any block. That is, ui(~a) = ui(a
0 i , ~ai) for any strategy a 0 i which assigns non-zero probability to any
action of the form (S, T) where |S| 6= 1.

Kronecker delta function: di,j = 1 if i = j, and 0 otherwise.
27All histories in an information set must be of the same length.
We have shown that ui(~a) = ui(a0 i , ~ai) for all strategies A 0 i of player i.
The theorem follows.Theorem X.12. Let ? = {Init, Challenge, Answer, Verify} be
a proof of capacity . For any number of players N, any number
of time steps K ? N, and any reward function ? : N ? N, let
(~a, ~µ) be an assessment of ATMcashGame?,K,? Where:

• ~a and aˆi are defined as in Theorem X.11, and for
each n ? N, we define ~an to be the completely mixed
strategy profile which (at history h) assigns probability 1/|Ai(h)| n to every action except aˆi
, and assigns allremaining probability to aˆi.

• ~µ is derived from ~a using Bayes' rule in the following
way: ~µ = limn?8 ~µ n, where for each n ? N, ~µ n is
derived from ~an using Bayes' rule.

Then (~a, ~µ) is a sequentially rational Nash equilibrium of
ATMcashGame?,K,?.

Proof. Let I ? Ii be any information set of player i in
ATMcashGame?,K,?, and let L be the length of histories in
I. It follows from Definition X.10 that the expected utility of

player i at I is ui((~a, ~μ)|I) = X j?[L] di,winners · only one j
(i, h) · ?(blocksj (h)) + u 0 I ((~a, ~μ)),where u 0 i is the utility function
of player i in the game ATMcashGame?,KL,?. Since winners,
onlyone, and blocks are invariant over histories within any given information set,
the summation term can be computed explicitly by player
i at I. Hence, in order to maximize his expected utility
at I, the player needs simply to maximize u 0 I ((~a, ~μ)). Let (~a|KL, ~μ|KL) denote the
assessment (~a, ~μ) for the first K - L time steps of the game. By Theorem X.11, ~a|KL
is a Nash equilibrium of ATMcashGame?,KL,?. Since ~μ is
derived from ~a by Bayes' rule, it follows that ui((~a, ~μ)|I) =
ui(((a 0 i, ~ai), ~μ)|I) for any strategy a 0 i of player i.

Applying this argument for every I, we conclude that (~a, ~μ) is sequentially
rational in ATMcashGame?,K,?.

By construction, limn?8 ~an = ~a and ~μ = limn?8 ~μ n, so
(~a, ~μ) is consistent. The theorem follows.

Parameters:.

The ATMcash Game is parametrized by N and
K. It is natural to ask: do we require that the number of
miners N is fixed in advance, or that the blockchain will end
after a certain number K of time-steps? The answer is no.

Theorem X.12 gives a sequentially rational Nash equilibrium
in which each player's strategy is independent of N, and so
it makes sense for each miner to play this strategy even if N
is unknown or changes over time. In light of this, from each
rational player's point of view, K can be considered to be
the number of time-steps that he intends to participate in the
Buying of capacity . Players' strategies in equilibrium do not
depend on the amount of capacity that (they believe) other players
possess. Also, we showed above that the equilibrium strategies
are robust to changes in N. Hence, if a player's amount of
capacity changes (eg he buys/sells a hard disk), then he can
simply create a new capacity commitment, and then behave as a
“new player” with the new amount of capacity.

The “51% Attack”.

If a player P controls more than half of the total capacity that
belongs to active miners, then following

the protocol rules is no longer a Nash equilibrium, because whichever branch of the blockchain P chooses to mine on will eventually become the highest-quality chain. Thus, P can decide arbitrary rules about which blocks to extend, and the other players will be incentivized to adapt their strategies accordingly. Moreover, P can prevent certain transactions from ever getting into the blockchain, by refusing to extend blocks which contain these transactions – as a consequence, P can mine multiple blocks per time-step without ever being punished. This attack was first analyzed by [20] in the context of Bitcoin, which suffers from the same problem (with respect to computing power rather than capacity).

It may seem unrealistic that a single party would control more than half of the total capacity that belongs to active miners in a widely adopted currency. A more realistic concern could be that a large group of miners (in a mining pool) may acquire more half of the total capacity . However, under the assumption that each miner is an individual strategic agent, we consider it unlikely that such a mining pool could do much damage: for this, a large group of self-interested and relatively anonymous agents would have to coordinate and trust each other throughout the duration of an attack. In particular, each rational miner in the pool must be convinced that he will get his share of the attack profits, and it seems highly unlikely that a large group of anonymous people would all trust each other so. The improbability of a 51% attack by a mining pool is supported by recent events: when a large mining pool (ghash.io) was nearing 50% of Bitcoin computing power in 2014, self-interested miners started leaving the mining pool in order to avoid destabilizing the currency

CONCLUSION

We have presented ATMcash, a cryptocurrency that uses efficient proofs of capacity instead of energy-intensive proofs of work to maintain a public ledger of all transactions. We have described a variant of a proof-of-capacity protocol that is more suitable for cryptocurrencies, and modified the structure of the hash chain and transactions to address some of the issues of other cryptocurrencies. We have also demonstrated the feasibility of ATMcash through a prototype, and show that maintaining a public ledger could be much more efficient

with proof-of-capacity . Finally, we do a game-theoretic analysis of ATMcash modeled as an extensive game, and prove that it satisfies strong equilibrium properties.

A. Proof-of-capacity Parameters

The two Pocapacity constructed in have the following efficiency/security properties. Below thash denotes the time 17required to evaluate the underlying hash function hash :

{0, 1}* ? {0, 1} L on inputs of length 2L (to hash an input of length m · L takes time m · thash by using Merkle-Damgard), ° For a given n, the number of nodes of the underlying graph, an honest prover P must dedicateN = 2 · n · L bits of storage (L · n for the labels, and almost the samefor the values required to efficiently open the Merkle tree commitment).

ATMcash uses the Shabal256 hash function, which below we will denote with H(·). To mine ATMcash, a miner first initialises his disk capacity as follows: he picks a nonce μ and an account identifier (which is a hash of a public key) Id, and then computes iteratively 4096 values x0, x1, . . . ? {0, 1} 256 As x0 = H(Id, μ) and (4) xi+1 = H(xikxi-1k . . . kx0) for i = 0, . . . , 4095 . (5) The miner then stores s0, . . . , s4095 where si = xi ? x4096. Each block is called a “scoop”, and the 4096 scoops together are called a “plot”. The miner is supposed to store as many plots as he can (using different nonces) until all the dedicated capacity is filled. To compute a plot, one must hash 4096·1+40962 ˜ 8 million 256-bit blocks. In the following we assume for simplicity that there is just one plot s0, . . . , s4095.
Efficiency. Once every few minutes, a new block gets added to the hash-chain. At this point the miner can compute a designated (public) index i ? {0, . . . , 4095} and must look up the value.

This then determines if the miner “wins” and thus can add the next block to the block chain30. Note that this requires accessing a constant fraction of the entire dedicated disk capacity (ie one block per plot, or 0.024%), every time a new block gets mined. Moreover, in order to verify that a miner “won” and can add a block, it is necessary to recompute the entire plot from the initial inputs (Id, μ), which, as mentioned above, involves hashing over 8 · 106 blocks.

Time-memory trade-offs. We observe that ATMcash allows for a simple time-memory trade-off: instead of storing an entire plot s0, . . . , s4095, a miner can initially compute and store only the value x 4096. The miner then re-computes the required scoop si at a given time-step, but only if i is sufficiently small (say, i = 10). This would require hashing only at most 50 blocks. Thus, the miner will get a shot at adding a block only at 10/4095 ˜ 0.25% of the time slots, but now also only requires a 1/4095 ˜ 0.025% fraction of the
Note that in equation (4), a freshly computed block xi is prepended to the previous input. This is important as Shabal256 is an iterated hash function: appending instead of prepending would bring the number of hashes required to compute a plot down to linear (instead of quadratic) in the length of the plot, but at the same time would allow for much more dramatic time-memory trade-offs than the ones outlined below.

To be precise, the miner computes x0, . . . , xi and sets si = xi ?x4096.

REFERENCES

[1] Bitcoin network graphs. http://bitcoin.sipa.be/index.html.
[2] ATMcash. http://ATMcash.info.
[3] Raspberry pi. www.raspberrypi.org.
[4] Slasher: A punitive proof-of-stake algorithm. https://blog.ethereum.org/2014/01/15/slasher-a-punitive-proof-of-stake-algorithm.
[5] N. Anderson. Mining Bitcoins takes power, but is it an “environmental disaster”?, April 2013. http://tinyurl.com/cdh95at.
[6] G. Ateniese, I. Bonacina, A. Faonio, and N. Galesi. Proofs of capacity : When capacity is of the essence. In M. Abdalla and RD Prisco, editors, SCN 14, volume 8642 of LNCS, pages 538–557. Springer, Sept. 2014.
[7] G. Ateniese, RC Burns, R. Curtmola, J. Herring, L. Kissner, ZNJ Peterson, and D. Song. Provable data possession at untrusted stores. In P. Ning, SDC di Vimercati, and PF Syverson, editors, ACM CCS 07, pages 598–609. ACM Press, Oct. 2007.
[8] KD Bowers, A. Juels, and A. Oprea. Proofs of retrievability: theory and implementation. In CCSW, pages 43–54, 2009.
[9] H. Buhrman, R. Cleve, M. Koucky, B. Loff, and F. Speelman. Com- ´ puting with a full memory: catalytic capacity . In Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014, pages 857–866, 2014.
[10] R. Di Pietro, L. Mancini, YW Law, S. Etalle, and P. Havinga. Lkhw:

a directed diffusion-based secure multicast scheme for wireless sensor networks. In Parallel Processing Workshops, 2003. Proceedings. 2003 International Conference on, pages 397–406, 2003.
[11] C. Dwork and M. Naor. Pricing via processing or combatting junk mail. In EF Brickell, editor, CRYPTO'92, volume 740 of LNCS, pages 139–147. Springer, Aug. 1993.
[12] S. Dziembowski. Proofs of capacity and a greener bitcoin, June 2013. Presentation at Workshop on Leakage, Tampering and Viruses, Warsaw. https://sites.google.com/site/warsawcryptoworkshop2013/abstracts.
[13] S. Dziembowski, S. Faust, V. Kolmogorov, and K. Pietrzak. Proofs of capacity . In CRYPTO 2015, 2015.
[14] S. Dziembowski, T. Kazana, and D. Wichs. One-time computable selferasing functions. In Y. Ishai, editor, TCC 2011, volume 6597 of LNCS, pages 125–143. Springer, Mar. 2011.
[15] P. Golle, S. Jarecki, and I. Mironov. Cryptographic primitives enforcing communication and storage complexity. In M. Blaze, editor, FC 2002, volume 2357 of LNCS, pages 120–135. Springer, Mar. 2003.
[16] ME Hellman. A cryptanalytic time-memory trade-off. IEEE Transactions on Information Theory, 26(4):401–406, 1980.
[17] A. Juels and BS Kaliski Jr. Pors: proofs of retrievability for large files. In P. Ning, SDC di Vimercati, and PF Syverson, editors, ACM CCS 07, pages 584–597. ACM Press, Oct. 2007.
[18] NP Karvelas and A. Kiayias. Efficient proofs of secure erasure. In Security and Cryptography for Networks - 9th International Conference, SCN 2014, Amalfi, Italy, September 3-5, 2014. Proceedings, pages 520–537, 2014.
[19] S. King and S. Nadal. Ppcoin: Peer-to-peer cryptocurrency with Proof-Of-Stake.
[20] JA Kroll, IC Davey, and EW Felten. The economics of Bitcoin mining, or Bitcoin in the presence of adversaries. In Workshop on the Economics of Information Security, June 2013.
[21] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz. Permacoin: Repurposing bitcoin work for data preservation. In 2014 IEEE Symposium on Security and Privacy, pages 475–490. IEEE Computer Society Press, May 2014.
[22] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system, 2009. http://bitcoin.org/bitcoin.pdf.
[23] JF Nash. Equilibrium points in n-person games. Proceedings of the National Academy of Sciences, 36(1):48–49, 1950.
[24] WJ Paul, RE Tarjan, and JR Celoni. capacity bounds for a game on graphs. Mathematical systems theory, 10(1):239–251, 1976–1977.
[25] D. Perito and G. Tsudik. Secure code update for embedded devices via proofs of secure erasure. In D. Gritzalis, B. Preneel, and M. Theoharidou, editors, ESORICS 2010, volume 6345 of LNCS, pages 643–662.

Springer, Sept. 2010.
[26] S. Valfells and JH Egilsson. Minting money with megawatts: How to mine bitcoin profitably, September 2015. http://www.researchgate.net/publication/278027487 Minting Money With Megawatts - How to Mine Bitcoin Profitably.
APPENDIX