Quantum Resistant Ledger (QRL)

peterwaterland@gmail.com

Noviembre de 2016

Resumen

Las monedas digitales privadas deben ser seguras frente los avances computacionales con el fin de perdurar en el tiempo. Se presenta el diseño y la emisión de un libro de contabilidad para criptomonedas usando firmas basadas en fórmulas (hash), las cuales son resistentes ante ataques computacionales con ordenadores clásicos y cuánticos.

1 Introducción

El concepto de un libro de contabilidad en pares por internet, registrado como una cadena de bloques y asegurado con prueba de trabajo fue reportado por primera vez en el año 2008 [11]. Bitcoin sigue a la cabeza como la criptomoneda más ampliamente usada a la fecha. Cientos de registros de criptomonedas han sido creados subsecuentemente, pero con algunas excepciones, ellos se basan en la misma criptografía de curva elíptica con claves públicas (ECDSA, por sus siglas en inglés) para generar firmas digitales que permiten que las transacciones sean verificadas de una forma segura. Los esquemas de firmas más ampliamente usados en la actualidad, como ECDSA, DSA y RSA son teóricamente vulnerables ante ataques con computadoras cuánticas. Resulta valioso explorar el diseño y la construcción de un libro de contabilidad de cadenas de bloque que sea resistente a ataques cuánticos para contrarrestar los potenciales riesgos de avances repentinos y no lineales de la informática cuántica.

2 Seguridad en las transacciones de Bitcoin

Actualmente sólo es posible utilizar esta moneda (salidas de transacciones no utilizadas) desde una dirección de Bitcoin al crear una transacción que contiene una firma de curva elíptica válida (secp256k1) desde la clave privada ($x \in N \mid x < 2^{256}$) para dicha dirección de Bitcoin específica. Si la clave privada realmente generada de forma aleatoria se mantiene secreta o se pierde, es razonable esperar que no haya movimiento de los fondos de esa dirección en ningún momento.

La probabilidad de que exista una colisión para una clave privada específica de Bitcoin es de 1 en 2^{256}. La probabilidad de cualquier colisión de claves de direcciones bitcoin puede estimarse utilizando la Paradoja del Cumpleaños. La cantidad de direcciones de bitcoin que deben ser generadas para obtener un 0,1% de posibilidades de una colisión es $5{,}4 \times 10^{23}$ [14].

Sin embargo, cuando una transacción es firmada ocurre que la clave pública de ECDSA de la dirección que envía los datos es revelada y almacenada en la cadena de bloques. La mejor práctica sería que las direcciones no fuesen reutilizadas, pero para noviembre de 2016 ya el 49,58% del saldo completo del registro contable de bitcoin se había retenido en direcciones cuyas claves públicas habían sido expuestas [1].

3 Vectores de ataque de informática cuántica

RSA, DSA y ECDSA permanecen seguras con base en la dificultad computacional de la factorización de números muy grandes, el problema del logaritmo discreto y el problema del logaritmo discreto de curva elíptica. El algoritmo cuántico de Shor (1994) resuelve la factorización de números grandes y logaritmos discretos en un tiempo polinómico. Por consiguiente, una computadora cuántica podría, en teoría, reconstituir la clave privada otorgada a una clave pública ECDSA. Es por medio de ECDSA que hay mayor vulnerabilidad ante ataques cuánticos, en comparación con RSA, debido a la utilización de tamaños de claves más pequeños, siendo una computadora cuántica de 1300 y 1600 qubit (2^{11}) capaz de descifrar las claves de 228 bit de ECDSA.

El desarrollo público de informática cuántica no ha sobrepasado los 2^5 qubits para la factorización de números pequeños (15 o 21). Sin embargo, en agosto de 2015 la Agencia de Seguridad Nacional de Estados Unidos (NSA, por sus siglas en inglés) desacreditó ostensiblemente la criptografía de curva elíptica con base en inquietudes acerca de la informática cuántica. No está claro cómo la computación cuántica avanzada podría surgir o que cualquier avance en este campo sea publicado para que los protocolos criptográficos de uso común en internet deban desarrollar una seguridad postcuántica. Con un cierto origen contrario al orden establecido, bitcoin podría convertirse en un primer blanco para un adversario con una computadora cuántica.

En caso de que ocurriese un avance significativo en la computación cuántica de forma pública, los desarrolladores de nodos podrían implementar esquemas de firmas criptográficas resistentes a ataques cuánticos dentro de la estructura de bitcoin, y alentar a todos sus usuarios a mover sus saldos de direcciones basadas en ECDSA a nuevas direcciones seguras ante riesgos cuánticos. Para mitigar la proporción de direcciones afectadas, sería razonable desactivar el reciclado de claves públicas a nivel del protocolo. Una actualización con tal planificación podría también resultar en el posible movimiento del millón de monedas que pertenecen a Satoshi Nakamoto – con la volatilidad del precio asociada.

Un escenario menos favorable sería un avance silencioso y no lineal en la computación cuántica seguido por un ataque matizado con un ordenador cuántico hacia las direcciones de bitcoin con claves públicas expuestas. Tales robos podrían tener un efecto devastador en el precio de cotización de bitcoin debido a una nueva presión fuerte en las ventas y una pérdida completa de confianza en el sistema a medida de que la escala de los robos se hace conocida. El rol de bitcoin como una tienda de valores ("oro digital") sería gravemente agraviado con consecuencias extremas a nivel mundial. En este contexto, los autores creen que es razonable experimentar con firmas criptográficas resistentes a ataques cuánticos en un registro de criptomonedas, y crear de forma potencial una tienda de valores de respaldo en caso de un Cisne Negro.

4 Firmas resistentes a ataques cuánticos

Existen varios sistemas criptográficos importantes que se creen son resistentes a ataques cuánticos: la criptografía basada en fórmula (hash), la criptografía basada en código, la criptografía basada en redes (lattice), criptografía con ecuaciones cuadráticas multivariable y criptografía de claves secretas. Todos estos esquemas se han diseñado para resistir tanto ataques computacionales clásicos como cuánticos, con tamaños de clave lo suficientemente grandes.

Existen esquemas de firmas digitales con seguridad adelantada basados en fórmulas (hash) con requisitos de seguridad mínimos que dependen solamente en la resistencia a la colisión de una función criptográfica hash. Al cambiar la función hash seleccionada se produce un nuevo esquema de firma digital basado en hash. Las

firmas digitales basadas en hash están bien estudiadas y representan el candidato principal para las firmas postcuánticas del futuro. De tal forma, son la clase elegida de firmas postcuánticas para el QRL.

5 Firmas digitales basadas en hash

Las firmas basadas en has resistentes a ataques cuánticos dependen de la seguridad de una función criptográfica hash de un solo sentido, la cual toma un mensaje *m* y genera como resultados la conversión con hash *h* con una longitud definida *n*, es decir, SHA-256, SHA-512. Utilizando una función criptográfica hash, debería ser computacionalmente imposible obtener a la fuerza *m* a partir de *h* (resistencia pre-imagen), u obtener a la fuerza *h* a partir de h_2, donde $h_2 = hash(h)$ (segunda resistencia pre-imagen), mientras que debería ser muy difícil encontrar dos mensajes ($m_1 \neq m_2$) que produzcan el mismo valor *h* (resistencia a colisión).

El algoritmo cuántico de Grover podría utilizarse para intentar encontrar una colisión hash o realizar un ataque pre-imagen para hallar *m*, lo cual requeriría $O(2^{n/2})$ operaciones. Por consiguiente, para mantener una seguridad de 128 bits, debe seleccionarse una longitud del resultado hash transformado, *n*, de al menos 256 bits – asumiendo una función criptográfica hash perfecta.

Las firmas digitales basadas en hash requieren una clave pública, *pk* , para la verificación, y una clave privada, *sk*, para firmar un mensaje. Se discutirán varias firmas de un solo uso (OTS, por sus siglas en inglés *One-Time Signatures*) basadas en hash considerando su idoneidad para ser incluidas como un registro contable de cadena de bloques.

5.1 OTS de Lamport-Diffie

En 1979, Lamport describió una firma de un solo uso basada en hash para un mensaje de *m* bits de longitud (generalmente la salida de una función hash resistente a colisión). La generación de pares de claves produce *m* pares de claves secretas aleatorias, $sk_j^m \epsilon \{0,1\}^n$ donde $j \epsilon \{0,1\}$, es decir, la clave privada es: $sk = \left((sk_0^1, sk_1^1), \dots, \dots, (sk_0^m, sk_1^m)\right)$. Siendo f una función hash de un solo sentido $\{0,1\}^n \rightarrow \{0,1\}^n$ con *m* pares de claves públicas generadas $pk_j^m = f(sk_j^m)$, es decir, la clave pública es: $pk = \left((pk_0^1, pk_1^1), \dots, \dots, (pk_0^m, pk_1^m)\right)$. Firmar involucra una inspección del hash del mensaje a nivel de bits para seleccionar sk_j (es decir, si *bit* = 0, sk_0 = 0; *bit* = 1, $sk_j = sk_1$), creando la firma: $s = (sk_j^1, \dots, sk_j^m)$ la cual revela la mitad de la clave privada. Para verificar una firma, la inspección a nivel de bits ($j \epsilon \{0,1\}$) del hash del mensaje verifica que $(pk_j = f(sk_j))^m$.

Asumiendo que luego del algoritmo de Grover se desee una seguridad de 128 bit, donde la longitud del mensaje sea un resultado hash definido de SHA256, *m* = 256 y *n* = 256, resultando en una *pk* = *sk* = 16kb, y una firma de 8kb para cada OTS utilizada. Una firma de Lamport debería utilizarse una sola vez, puede generarse muy rápidamente, pero por consiguiente tendrá tamaños grandes de claves, de firmas y, por ende, de transacciones, haciéndola impráctica para un registro público de cadena de bloques.

5.2 OTS de Winternitz

Para la transformación de un mensaje, M, de *m* bits de longitud con claves secreta y pública de *n* bits de longitud, una función de un solo sentido $f: \{0,1\}^n \rightarrow \{0,1\}^n$ y un parámetro de Winternitz, $w \epsilon N \mid w > 1$, la idea general de la firma de un solo uso de Winternitz es aplicar una función hash iterativa en una lista de

claves secretas aleatorias, $sk \epsilon \{0,1\}^n, sk = (sk_1, \dots, sk_{m/w})$, creando cadenas de hashes de longitud w – 1, que finalizan con claves públicas $(pk \epsilon N \mid \{0,1\}^n), pk_x = f^{2^{w-1}}(sk_x), pk = (pk_1, \dots, pk_{m/w})$.

A diferencia de la inspección a nivel de bits del mensaje transformado en la firma de Lamport, en este caso el mensaje se analiza w bits a la vez para extraer un número $i \epsilon N, i < 2^w - 1$, del cual se genera la firma $s_x = f^i(sk_x), s = (s_1, \dots, s_{m/w})$. Con un valor creciente de w proporcionando una compensación entre claves más pequeñas y firmas para un esfuerzo computacional mayor [10].

La verificación involucra la generación simple de $pk_x = f^{2^{w-1}-i}(s_x)$ a partir de M, s y confirmar la coincidencia de las claves públicas.

Utilizando SHA-2 (SHA-256) como una función criptográfica hash de un solo sentido, f: m = 256 y n = 256, con w = 8 resulta en una pk = sk = s de un tamaño igual a $\frac{(m/w)n}{8}$ bytes = 1 kb.

Para generar pk se requieren f^i iteraciones hash, donde $i = \frac{m}{w}2^{w-1} = 8160$ por cada generación de par de claves OTS. A w = 16, las claves y firmas tienen la mitad del tamaño, pero i = 1048560 se convierte en un valor impráctico.

5.3 OTS+ de Winternitz (W-OTS+)

Buchmann introdujo una variante de la OTS original de Winternitz al cambiar la función de iteración de un solo sentido a que se aplicara a un número aleatorio, x, repetidamente, pero esta vez parametrizado por una clave, k, la cual se genera a partir de una iteración previa de $f_k(x)$. Esto es fuertemente infalsificable bajo ataques adaptativos a mensajes seleccionados al utilizar una función pseudo-aleatoria (PRF, por sus siglas en inglés) y una prueba de seguridad puede calcularse para parámetros dados [3]. Con ella se elimina la necesidad de una familia de funciones hash resistente a colisión al realizar un recorrido aleatorio a través de la función en vez de una simple iteración. Huelsing presentó una variante adicional, W-OTS+, permitiendo la creación de firmas más pequeñas con una seguridad de *bits* similar a través de la adición de una máscara de bit XOR en la función de concatenación iterativa [6]. Otra diferencia entre W-OTS (variante de 2011)/W-OTS+ y W-OTS es que el mensaje se analiza a razón de $\log_2(w)$ bits a la vez, en vez de w bits a la vez, disminuyendo las iteraciones de la función hash pero aumentando los tamaños de las claves y firmas.

W-OTS+ será brevemente descrita a continuación. Con el parámetro de seguridad $n \epsilon N$, correspondiente a la longitud del mensaje (m), las claves y firmas en bits, que son determinadas por la función criptográfica hash elegida y por el parámetro de Winternitz, $w \epsilon N \mid w > 1$ (por lo general {4, 16}), l es calculado. l es el número de elementos de la cadena de n *bits* en una clave o firma WOTS+, donde $l = l_1 + l_2$.

$$l_1 = \left\lceil \frac{m}{log_2(w)} \right\rceil, l_2 = \left\lfloor \frac{log_2(l_1(w-1))}{log_2(w)} \right\rfloor + 1$$

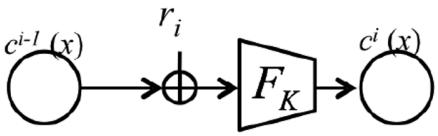
Una función hash con clave se utiliza, $f_k: \{0,1\}^n \rightarrow \{0,1\}^n \mid k \epsilon \{0,1\}^n$. En pseudocódigo:

$$f_k(M) = Hash(Pad(K)||Pad(M))$$

Donde $Pad(x) = (x||10^{b|x|1})$ para $|x| < b$

La función de concatenación $c_k^i(x,r)$: a la entrada de $x \in \{0,1\}^n$, el contador de iteración i, la clave $k \in K$ y los elementos de aleatorización $r = (r_1, \ldots, r_j) \in \{0,1\}^{n*j}$ con $j \geq i$, se definen de la siguiente forma.

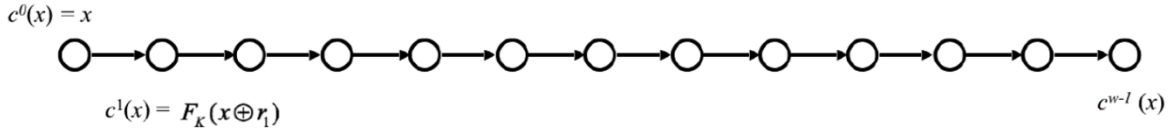
Figura 1. Función de concatenación W-OTS+



Donde:

$$c^i(x,r) = \begin{cases} x & si\ i = 0 \\ f_k\ (c_k^{i-1}(x,r) \oplus r_i & si\ i > 0 \end{cases}$$

Figura 2. Ejemplo de generación de cadena hash



Que sea un XOR a nivel de bit para la iteración previa de c_k y el elemento de aleatorización seguido por f_k en el resultado, el cual luego se utiliza como valor inicial de la próxima iteración c_k.

5.3.1 Clave de firma

Para crear la clave secreta *sk*, se eligen de forma uniforme y aleatoria cadenas de $l + w - 1n$ bits (con PRF), en las cuales la primera *l* forma la clave secreta, $sk = (sk_1, \ldots, sk_l)$ y las cadenas $w\text{-}1n$ restantes se convierten en $r = (r_1, \ldots, r_{w-1})$. Una clave función, *k*, es elegida de forma uniforme y aleatoria.

5.3.2 Clave de verificación

La clave pública es:

$$pk = (pk_0, pk_1, \ldots, pk_l) = ((r,k), c_k^{w-1}(sk_1, r), c_k^{w-1}(sk_2, r), \ldots, c_k^{w-1}(sk_l, r))$$

Hay que destacar que pk_0 contiene *r* y *k*.

5.3.3 Firma

Para realizar una firma: un mensaje M de longitud *m* es analizado de forma tal que $M = (M_1, \ldots, M_{l_1}), M_i \in \{0, w-1\}$ (creando una representación de M con base en *w*).

Luego la suma de control, C, de longitud l_2 se calcula y se anexa:

$$C = \sum_{i=1}^{l_1} (w - 1 - M_i)$$

De forma tal que: M + C = $b = (b_0, \ldots, b_l)$
La firma es:

$$s = (s_1, \ldots, s_l) = (c_k^{b_i}(sk_1, r), \ldots, c_k^{b_l}(sk_l, r))$$

5.3.4 Verificación

Para verificar una firma, $b = (b_1, \ldots, b_l)$ se reconstruye a partir de M.

Si $pk = c_k^{w-1-b_1}(s_1), \ldots, c_k^{w-1-b_l}(s_l)$ entonces la firma se considera válida.

W-OTS+ proporciona un nivel de seguridad de al menos $n - w - 1 - 2log(l\ w)$ bits [3]. Una firma típica donde $w = 16$ utilizando SHA-256 ($n = m = 256$) es $l\ n$ bits o 2,1 kb.

6 Esquemas de firmas de Árbol de Merkle

Mientras que las firmas de un solo uso proporcionan una seguridad criptográfica satisfactoria para firmar y verificar transacciones, tienen una desventaja mayor porque sólo pueden utilizarse de forma segura en una sola ocasión. Si la dirección de un registro de contabilidad se basa en alguna transformación de la clave pública de un par de claves OTS particular, entonces esto podría llevar a un registro de cadena de bloques extremadamente restrictivo, donde todos los fondos de una dirección emisora requerirían ser movidos en cada transacción realizada – o tales fondos tendrían un riesgo de robos.

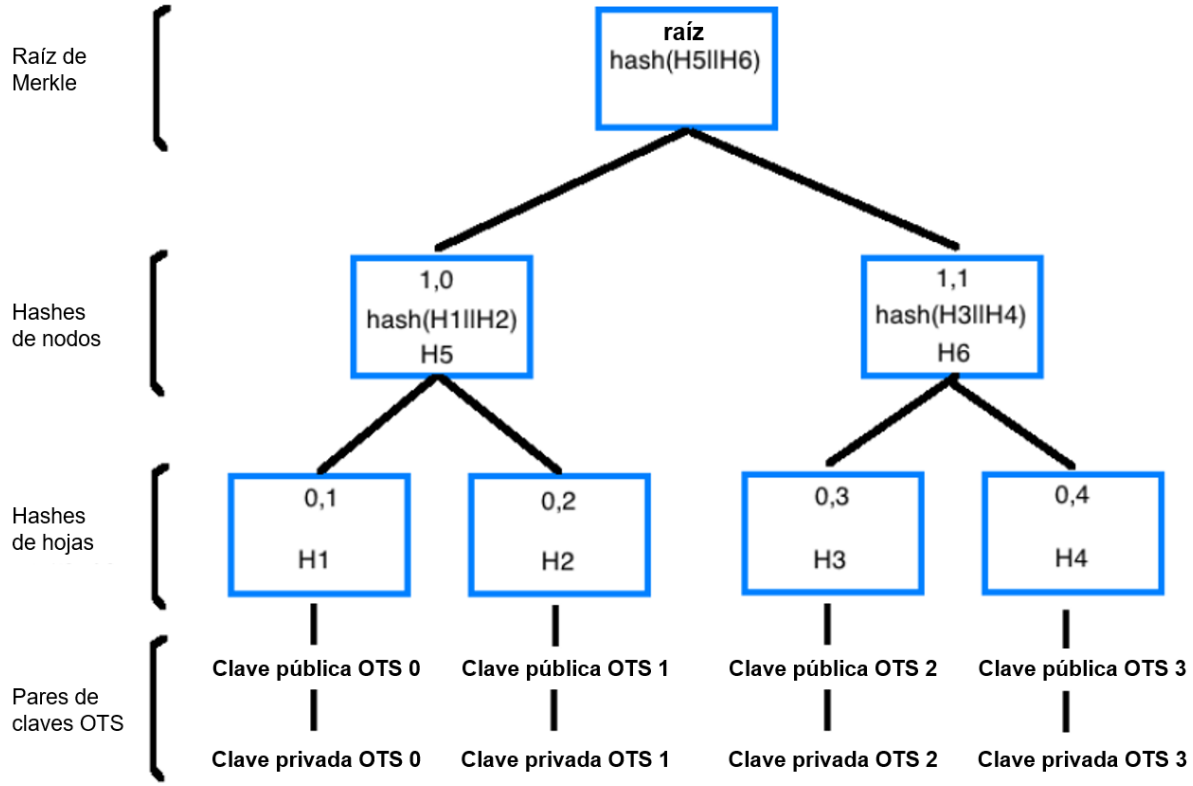
Una solución es extender el esquema de firma para incorporar más de una firma OTS válida para cada dirección de registro, permitiendo así tantas firmas como pares de claves OTS sean pre-generados. Un árbol hash binario conocido como el árbol de Merkle es la forma lógica de conseguir esto.

6.1 Árbol hash binario

La idea general tras un árbol de Merkle es un árbol inverso compuesto por nodos matrices calculados mediante la aplicación de fórmula hash de nodos hijos, de abajo hacia arriba, en capas y hasta la raíz. La existencia de cualquier nodo u hoja puede ser criptográficamente demostrada con cálculos computacionales de la raíz.

Un árbol de Merkle se forma a partir de n cantidad de hojas base y tiene la altura hasta la raíz de Merkle, h ($n = 2^h$) – comenzando desde los hashes hoja (capa 0) y contando hacia arriba con cada capa de nodos. Cada nodo hoja es creado en nuestro caso de utilización hipotético del registro de contabilidad calculando con una fórmula hash una clave pública OTS aleatoriamente pre-generada. Desde la base del árbol se puede apreciar que el nodo que se encuentra encima de cada par de hojas hash se ha creado aplicando una fórmula hash a la concatenación de hashes hijos.

Figura 3. Ejemplo de esquema de firmas de Árbol de Merkle



Esto continúa hacia arriba a través de capas del árbol hasta llegar a una confluencia en el hash raíz del árbol, conocido como la raíz de Merkle.

Del árbol presentado como ejemplo en la figura anterior, al tomar la raíz de Merkle como la clave pública, se pueden utilizar cuatro pares de claves OTS pre-calculados para generar cuatro firmas criptográficamente válidas y seguras de un solo uso. La raíz de Merkle del árbol hash binario puede transformarse en la dirección del registro de contabilidad (posiblemente aplicando fórmulas hash de forma iterativa con una suma de verificación anexa). Una firma completa, S, de un mensaje M para un par de claves OTS dado incluye: la firma, s, el número de clave OTS, n, y la ruta de autenticación de Merkle, es decir, para el par de claves OTS 0 (entonces $n = 0$):

$$S = s, n, clave\ pública\ OTS\ 0, H1, H2, H5, H6, raíz$$

Considerando que la clave pública OTS y un hash hoja pueden ser deducidos a partir de s, y los nodos matrices pueden calcularse a partir de sus nodos hijos, esto en realidad puede reducirse a:

$$S = s, n, H2, H6, raiz$$

Donde S es válido al verificar la clave pública OTS a partir de s y M, y luego al verificar los hashes de la ruta de autenticación de Merkle se obtiene una raíz de Merkle coincidente (clave pública).

6.2 Estado

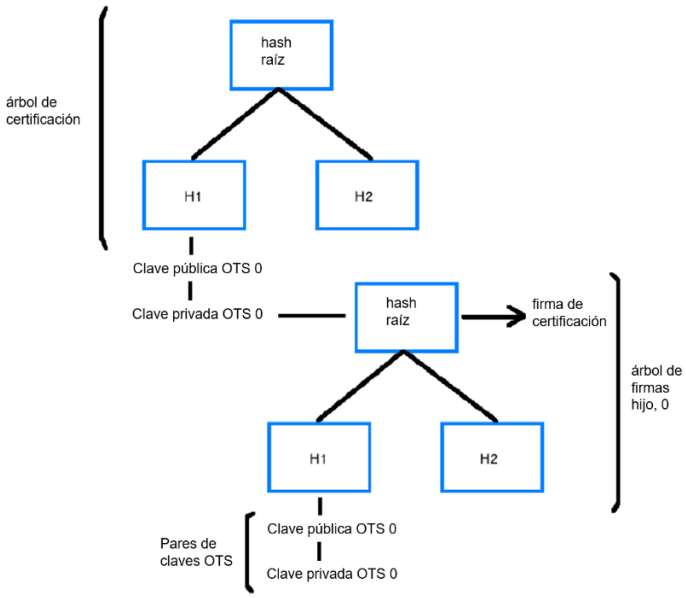
Usar el esquema de firma de Merkle (MSS; por sus siglas en inglés) anteriormente mostrado de forma segura dependerá de no reutilizar las claves OTS. Por consiguiente, depende del estado de las firmas o de las transacciones firmadas que son registradas. Con el uso ordinario que se le da en el mundo real, esto podría convertirse en un problema potencial, pero un registro público inmutable de cadena de bloques es el medio de almacenamiento ideal para un esquema criptográfico de firmas. Un esquema criptográfico de firmas basado en hash más nuevo llamado SPHINCS, el cual ofrece firmas prácticas sin estado con una seguridad de 2^{128}, bits fue reportado en 2015 [2].

6.3 Hiperárboles

Un problema con el MSS básico es que la cantidad de firmas posibles está limitada y todos los pares de claves OTS deben ser pre-generados antes del cálculo del árbol de Merkle. La generación de claves y el tiempo de firma crecen exponencialmente con la altura del árbol, h, lo que implica que la generación de árboles mayores a 256 pares de claves OTS se vuelve temporal y computacionalmente muy costosa.

Una estrategia para deferir la computación durante la generación de claves y árboles, y además de extender la cantidad de pares de claves OTS disponibles es utilizar un árbol que a su vez esté compuesto por árboles de Merkle, lo cual se conoce como un hiperárbol. La idea general es firmar la raíz de Merkle de un árbol hijo con una clave OTS del hash hoja de un árbol de Merkle matriz conocido como el árbol de certificación.

Figura 4. Vinculación de árboles de Merkle



En la forma más simple (altura, h = 2), un árbol de certificación está pre-computado con 2^1 pares de claves OTS, y cuando se requiere la primera firma se procede a calcular un nuevo árbol de Merkle de firmas (árbol de firmas 0), el cual es firmado por uno de los pares de claves OTS del árbol de certificación. El árbol de firmas está compuesto por n hashes hoja con sus correspondientes pares de claves OTS, y éstos sirven para firmar mensajes según se requiera. Cuando cada par de claves OTS en el árbol de firmas ha sido utilizado, entonces el siguiente árbol de firmas (árbol de firmas 1) es firmado por el segundo par de claves OTS del árbol de certificación, haciendo posible el siguiente lote de firmas.

Una firma, S, de tal construcción de tipo hiperárbol se convierte ligeramente más complicada e incluiría:
1. Del árbol de firmas: *s*, *n*, *ruta de Merkle*, *raíz*.
2. De cada árbol de certificación: *s* (de la raíz de Merkle del árbol hijo), *n*, *ruta de Merkle*, *raíz*.

Es teóricamente posible anidar capas de árboles desde el árbol de certificación para extender el MSS original de forma infinita. El tamaño de la firma crece linealmente para cada árbol adicional que es firmado, mientras que la capacidad de firma del hiperárbol aumenta exponencialmente.

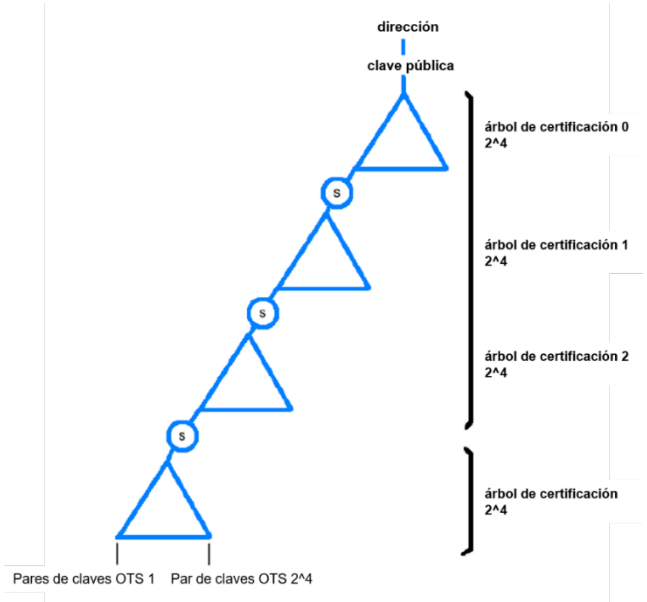
6.3.1 Ejemplos de hiperárboles

Para demostrar cuán fácil el MSS puede extenderse con la construcción de un hiperárbol, considere un árbol de certificación inicial de altura h_1 = 5, con 2^5 hashes hoja y con pares de claves OTS asociados. La raíz de Merkle de este árbol se transforma para generar la dirección del registro de contabilidad. Otro árbol de Merkle, un árbol de firmas de tamaño idéntico (h_2 = 5, 2^5 hashes hojas y pares de claves OTS) es instanciado. Son posibles 32 firmas antes de que deba crearse el próximo árbol de firmas. La cantidad total de firmas posibles es $2^{h_1+h_2}$, que en este caso es 2^{10} = 1024.

En una Macbook Pro de 2.7Ghz i5, con 8 Gb de RAM creando pares de claves OTS y un árbol de certificación de Merkle para varios tamaños, el rendimiento resultante fue el siguiente (código Phyton no optimizado, OTS de Winternitz): 2^4 = 0,5 s; 2^5 = 1,2 s; 2^6 = 3,5 s; 2^8 = 15,5 s. Un hiperárbol que consista en la generación inicial de dos árboles de 2^4 toma cerca de 1 s, en comparación con los 15.5 s para el árbol estándar del MSS de 2^8 con la misma capacidad de firma.

Al aumentar la profundidad (o la altura) de un hiperárbol se mantiene esta tendencia. Un hiperárbol compuesto por cuatro árboles de certificación de 2^4 encadenados y un árbol de firmas de tamaño 2^4 es capaz de realizar 2^{20} = 1.048.576 firmas, con un costo añadido al tamaño de la firma, pero con un tiempo de creación de apenas 2,5 s.

Figura 5. Construcción de un hiperárbol

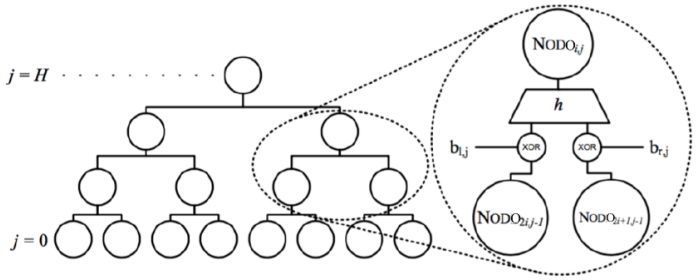


No hay requisitos que obliguen a un hiperárbol a que sea simétrico, y si está compuesto inicialmente por dos árboles, luego podría extenderse firmando capas adicionales de árboles. Las firmas de una dirección de registro podrían entonces iniciar siendo pequeñas y eventualmente aumentar a medida que incrementa la profundidad del hiperárbol. Utilizando un hiperárbol de Merkle para crear y firmar transacciones de una dirección de registro nunca existirá la necesidad de requerir $>2^{12}$ transacciones. Por consiguiente, la capacidad de firmar con facilidad computacional 2^{20} veces de forma segura con un hiperárbol de $h = 5$ de profundidad será más que suficiente.

6.4 XMSS

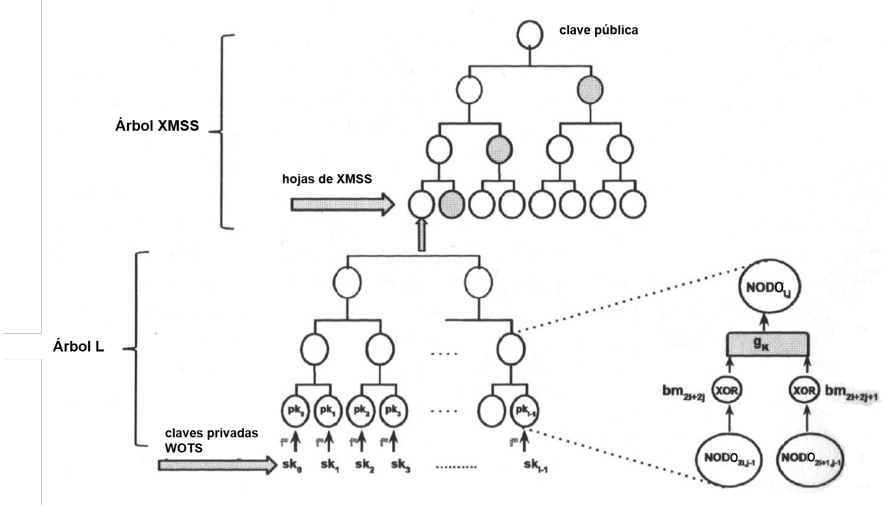
El esquema extendido de firma de Merkle (XMSS, por sus siglas en inglés) fue reportado por primera vez por Buchmann et. al en 2011, y fue publicado como un boceto de IETF el año pasado [4][7]. Probablemente tenga un esquema de seguridad pensado hacia el futuro y existencialmente infalsificable bajo ataques de mensajes seleccionados con requisitos mínimos de seguridad: una PRF y una segunda función hash resistente a pre-imagen. El esquema permite la extensión de firmas de un solo uso a través de un árbol de Merkle, siendo la mayor diferencia el uso de la máscara de bits XOR de los nodos hijos antes de la concatenación de los hashes hacia el nodo matriz. El uso de la máscara de bits XOR permite que la familia de funciones hash resistente a colisión sea reemplazada.

Fig. 1. La construcción del árbol XMSS



Las hojas del árbol no son hashes de pares de claves OTS, sino la raíz de árboles L hijos que contienen las claves públicas OTS, con *l* cantidad de piezas que forman las hojas base. El OTS+ de Winternitz se utiliza para las firmas de un solo uso (sin considerar lo descrito antes para la variante de 2011).

Figura 7. Construcción de XMSS [8]



La longitud en bits de la clave pública de XMSS es $(2(H + \lceil \log l \rceil) + 1)n$, una firma XMSS tiene una longitud $(1 + H)n$, y la longitud de la clave secreta de firma XMSS es < 2n.

Buchmann reporta un rendimiento con un procesador Intel® i5 de 2,5 Ghz para un árbol XMSS de altura h = 20, donde w = 16 y la función criptográfica hash utilizada es SHA-256 (m = 256) de hasta alrededor de un millón de firmas. Con los mismos parámetros y hardware, la firma tomó 7ms, la verificación 0,52ms y la generación de claves 466 segundos. El nivel de seguridad alcanzado con tales parámetros fue 196 bits para una clave pública de 1,7 kb de tamaño, de 250 bits para una clave privada y una firma de 2,8 kb. XMSS es un esquema atractivo, con la mayor desventaja de tener un tiempo de generación de claves extremadamente largo.

6.5 Rendimiento del árbol XMSS

Usando una biblioteca de Python no optimizada construida para la formación de nodos QRL de prueba de un árbol XMSS de 4096 hojas (h=12) con todas las claves y máscaras de bits generadas a partir de una PRF basada en hash transcurrieron 32 segundos en el mismo hardware descrito anteriormente (una Macbook Pro de 2.7 Ghz i5, 8 Gb de RAM). Esto incluyó la generación a través de una PRF de más de 8000 máscaras de bits y más de 300.000 fragmentos sk. Un algoritmo de árbol de Merkle transversal más eficiente, y la necesidad de solamente tener que realizar w – 1 hashes por función de cadena de clave secreta en WOTS+ en vez de 2^{w-1} con WOTS contribuye a la demarcada mejora en el rendimiento sobre un MSS convencional.

Se consiguió un tamaño de firma completo cerca de 5,75 kb con esta estructura (codificación de cadena hexadecimal de 1,75 kb) incluyendo: el par de claves OTS n, la firma, la ruta de autorización XMSS, la clave pública OTS y la clave pública del árbol XMSS (incluyendo también la semilla de la clave pública PRF y la raíz del árbol XMSS).

Para árboles de varias capacidades de firma generados usando una PRF y una semilla aleatoria, se obtuvo el siguiente rendimiento: (h = 9) 512 4,2 s; (h = 10) 1024 8,2s; (h = 11) 2048 16,02 s.

7 Esquema de firma propuesto

7.1 Requisitos de seguridad

En el diseño del QRL es importante que la seguridad criptográfica del esquema de firma sea segura contra ataques computacionales clásicos y cuánticos, tanto en el presente como en las décadas futuras. Un esquema XMSS usando SHA-256, donde w = 166, ofrece una seguridad de 196 bits con resguardo previsto contra ataques computacionales de fuerza bruta hasta el año 2164 [9].

7.2 Firmas QRL

Se propone un esquema de hiperárbol de firma extensible, provisto de estado y asimétrico, compuesto de árboles XMSS encadenados. Éste tiene el doble beneficio de utilizar un esquema de firma validado y de permitir la generación de direcciones del registro de contabilidad con la capacidad de firmar transacciones, evitando un retraso prolongado en el pre-cálculo generalmente observado en construcciones XMSS gigantes. Se ha elegido la firma basada en hash W-OTS+ de un solo uso para este esquema, tanto por motivos de seguridad como por su rendimiento.

7.3 Construcción del hiperárbol

7.3.1 Tamaños de claves y firmas

A medida de que el número de árboles dentro de un hiperárbol aumenta, el tamaño de las claves y de las firmas incrementa linealmente – pero la capacidad de la firma aumenta exponencialmente. Los tamaños de varias claves públicas y firmas derivadas de un árbol XMSS (con base en la descripción hecha en 2011), donde w = 16, m = 256, h es la altura del árbol y SHA-256 se ha elegido como el algoritmo criptográfico hash, son:

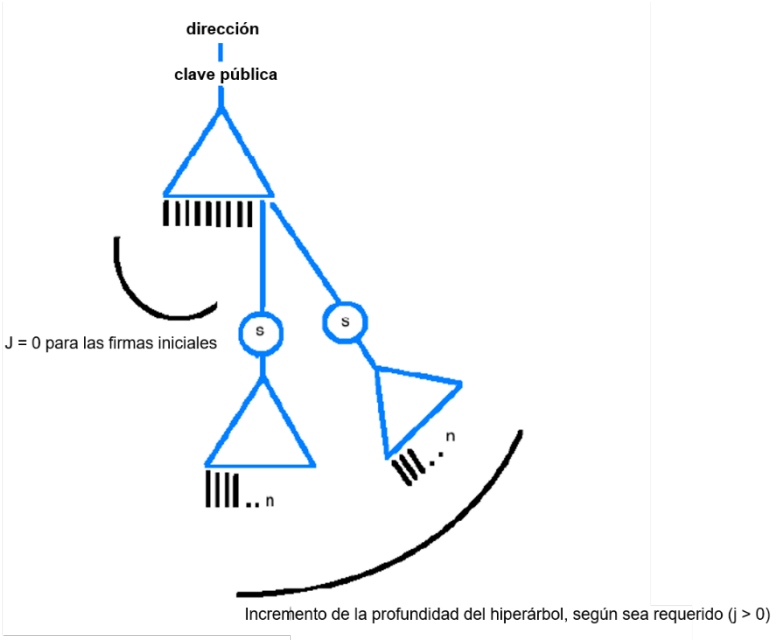
- h = 2, 2^2 firmas: clave pública 0,59 kb, firma 2,12 kb (0,4 s).
- h = 5, 2^5 firmas: clave pública 0,78 kb, firma 2,21 kb (0,6 s).
- h = 12, 2^{12} firmas: clave pública 1,23 kb, firma 2,43 kb (32 s).
- h = 20, 2^{20} firmas: clave pública 1,7 kb, firma 2,69 kb (466 s [3]).

La compensación obtenida al crear un hiperárbol XMSS (4 árboles, j = 3, h = 5) con una capacidad de firma eventual de 2^{20} en menos de 3 s, en comparación con los 466 s antes mostrados, para una firma de 8,84 kb en vez de 2,69 kb puede ser aceptable.

7.3.2 Asimetría

Crear un árbol asimétrico permite que las primeras firmas surjan con una construcción única de árbol XMSS, la cual se va extendiendo a medida que sea necesario para firmas posteriores con un costo en la capacidad global de firmas. El fundamento para esta afirmación es que es muy probable que no haya consecuencias para una aplicación de registro de cadena de bloques y la cartera puede otorgar al usuario la opción de la capacidad de firma frente a los tamaños de firmas/claves. Una profundidad máxima de árbol de j = 2 debería bastar para todas las circunstancias.

Figura 8. Hiperárbol asimétrico



7.4 Especificación del hiperárbol QRL

Los siguientes parámetros por defecto deberán ser adoptados para la construcción de un hiperárbol estándar:

- $j = 0$ $(j \epsilon \{0 \leq x \leq 2\}), h = 12$ $(h \epsilon \{1 \leq x \leq 14\})$, límite superior de firmas posibles: 2^{36}, tamaño mínimo de firmas: 2,21 kb, tamaño máximo de firmas: 7,65 kb.

Es decir, un solo árbol XMSS de h = 12 con 4096 firmas disponibles, el cual puede extenderse con árboles adicionales de hasta h = 14, según sea necesario. Para la mayoría de los usuarios, es poco probable que se requieran árboles adicionales.

7.4.1 Ejemplo de firma QRL

Asumiendo la construcción de hiperárbol más complicada donde j = 2 y h = 14, una firma para un mensaje de transacción, m, donde n es la posición del par de claves OTS para cada árbol XMSS, requeriría:

- Árbol de firmas, j = 2: firma OTS de m, n, prueba de autenticación de Merkle, raíz de Merkle del árbol de firmas.
- Árbol de certificación, j = 1: firma OTS de la raíz de Merkle del árbol de firmas (j = 2), n, prueba de autenticación de Merkle, raíz de Merkle.
- Árbol XMSS original, j = 0: firma OTS de la raíz de Merkle (j = 1), prueba de autenticación de Merkle, raíz de Merkle.

La verificación involucra generar la clave pública OTS a partir de m y la firma, luego confirmar la prueba de autenticación de Merkle suministrada para generar la raíz de Merkle del árbol de firmas. Este se convierte en el mensaje para la siguiente firma OTS, y a partir de este valor se genera la siguiente clave pública OTS, la prueba de autenticación de Merkle suministrada utilizada para recrear la raíz de Merkle del árbol de certificación, la cual se convierte en el mensaje para la firma OTS del siguiente árbol de certificación, y así sucesivamente. Una firma sólo es válida si la raíz de Merkle del árbol más grande, el árbol XMSS original, (j = 0) es generada correctamente.

Es importante destacar que las claves públicas OTS no son necesarias para la verificación de la firma del árbol XMSS. De hecho, la raíz de Merkle para cada árbol puede también deducirse y, por consiguiente, omitirse con la verificación de la firma del hiperárbol si se conoce la dirección del registro emisor (ya que esto es un cálculo derivado de la raíz de Merkle del árbol XMSS de certificación de mayor jerarquía (j = 0) dentro de la firma QRL – ver la sección Cuentas más abajo).

Mientras que el esquema de firmas tenga estado, la implementación de la cartera debe retener y actualizar n para cada árbol XMSS generado en el hiperárbol para una dirección dada.

7.5 PRF

PRF a partir de la semilla. HMAC_DRBG.

7.6 Cartera determinística

Utilizando una única semilla SEED es posible generar un árbol XMSS muy amplio que debería bastar para la mayoría de los usuarios durante un período prolongado. Se utiliza una fuente segura de entropía para generar esta SEED, la cual atraviesa una función PRF segura para generar un conjunto de claves pseudo-aleatorias con las cuales se genera el árbol. Una desventaja de utilizar el mismo árbol XMSS es que el usuario está confinado a una dirección única (aunque la exposición de la clave pública no es problema con un MSS).

Una dirección Bitcoin o Ethereum se deriva de la clave pública asociada, y de esta forma una clave única privada o pública sólo puede crear una dirección única. Sin embargo, una dirección XMSS se deriva de la clave pública, PK, la cual contiene la raíz de Merkle y la SEED pública. Si la SEED permanece constante pero la cantidad de pares de claves OTS para calcular el árbol varía, entonces la raíz de Merkle cambiará para cada variación. Por consiguiente, para cada adición o sustracción única de un par de claves OTS en particular, la dirección derivada cambiará.

Esta característica puede ser utilizada por un software de cartera/nodo para generar numerosas variaciones del árbol XMSS (extendiéndolo/contrayéndolo según sea necesario utilizando la misma SEED inicial), permitiendo la generación de tantas direcciones únicas como sean necesarias. Registrar esta información de forma segura, compacta y con estado definido es computacionalmente trivial.

8 Parámetros de diseño de las Criptomonedas

Las secciones remanentes de este artículo definirán los parámetros del diseño propuestos para el registro de contabilidad QRL. El objetivo del registro es que sea una cadena de bloques pública altamente segura contra vectores de ataques computacionales clásicos y cuánticos. Se trata de un borrador inicial, por lo cual cada aspecto está sujeto a cambios potenciales.

8.1 Prueba de participación (*Proof-of-Stake*)

QRL busca ser un registro de contabilidad de cadena de bloques público y abierto, asegurado con un algoritmo de prueba de participación (proof-of-stake). Una época tiene una duración de 10.000 bloques. Los validadores de participación son determinados a partir de transacciones de participación en la época anterior. La idea general es que cada validador de participación firme una transacción que contiene el hash final de una cadena iterativa de 10.000 hashes de longitud (una máscara de bits XOR puede aplicarse durante cada iteración para reducir los requisitos de seguridad de la función hash). Con la transacción de participación confirmada en la cadena, cada nodo en la red ahora puede enlazar la identidad criptográfica de la dirección de participación hacia la cadena de hash para la siguiente época.

8.1.1 Diseño y aleatoriedad

Para cada bloque, cada nodo de validación que está participando en la época actual revela el siguiente hash consecutivo al anterior en la cadena para demostrar criptográficamente la participación, y votar para convertirse en el selector de bloques ganador.

HMAC_DRBG se utiliza para generar salidas como secuencias de números pseudo-aleatorias de 32 bytes a partir de datos semilla tomados de la cadena de bloques (el bloque generador inicialmente, luego la entropía añadida tomada de hashes de encabezado de bloques recientes para cada época subsecuente).

Por tanto, cada bloque que ha elegido el validador de participación para convertirlo en el selector de bloque se determina siendo el hash de revelación más cercano numéricamente a la salida PRF para dicho bloque. Esto es difícil de obtener ya que la PRF es desconocida por los participantes al momento de la creación de la cadena hash. Adicionalmente, una cadena de hash iterativa (provista de claves) es esencialmente una secuencia numérica aleatoria. Por último, incluso si los validadores de participación de alguna forma colisionan, no hay manera de que conozcan los contenidos de la cadena de hash del otro validador de participación, pues aún no han sido revelados.

Para prevenir que un bloque retenga un ataque, se asocia la falla al producir un bloque válido luego de la presentación de un hash de revelación válido con la pérdida de la compensación del bloque entero de dicha dirección, y luego se le prohíbe la participación durante un período de castigo.

Para mitigar el ataque de un bloque vacío de nodos sybil o direcciones del validador de participación con bajos balances de cuentas, se utiliza un límite de lista flexible para el validador de participación. La compensación del bloque se paga de forma ponderada, con base en el saldo de la dirección participante. Con el mensaje hash revelado cada nodo también divulga un hash de la raíz del árbol de Merkle de una lista ordenada de txhashes en sus conjuntos de transacción, conjuntamente con el número de transacciones que esperan un bloque. Cada nodo representa un porcentaje respecto al tope y al fondo para saber cuántas transacciones son de esperar para el próximo bloque. Si el bloque está vacío o tiene una cantidad menor a la esperada, los validadores de participación quedan habilitados para contratos de participación hacia arriba (excluyendo validadores de participación de pobres a ricos) entre los bloques. Si los nodos del selector de bloques se comportan de forma honesta, entonces lo contrario es verdadero y la cantidad permitida de validadores de participación involucrados aumenta. Los fondos no pueden moverse durante la época en la que participan – esto previene los intentos de manipular la selección de bloques por la creación sybil de numerosas direcciones de validadores de participación.

8.2 Tarifas

Los tamaños más grandes de transacciones en comparación con otros registros de contabilidad requieren que una tarifa sea cancelada con cada transacción. El autor opina que los mercados artificiales de tarifas (ver Bitcoin) son innecesarios y se desplazan en dirección opuesta al ideal de un registro de cadena de bloques público y abierto. Cada transacción, si se acompaña con el pago de una tarifa mínima, debería ser tan válida como cualquier otra. Las tarifas mínimas que los mineros de criptomonedas están dispuestos a aceptar deberían ser flotantes y definirse según el mercado. Es decir, los nodos/mineros fijan de forma competitiva el límite inferior de las tarifas entre ellos mismos. Un valor mínimo absoluto se impondrá a nivel del protocolo. Por consiguiente, los mineros ordenarán transacciones del conjunto mempool para la inclusión en un bloque a su discreción.

8.3 Bloques

8.3.1 Tiempos de bloques

Bitcoin tiene un tiempo entre bloques cercano a 10 minutos, pero con una varianza natural esto puede, en ocasiones, conllevar a períodos considerablemente largos antes de que el siguiente bloque sea minado. Nuevos diseños de registros como Ethereum han mejorado esto, y se benefician al tener tiempos de bloques mucho menores (15 segundos) sin perder seguridad o la centralización de mineros de altas tasas de bloques

huérfanos/caducados. Ethereum usa una versión modificada del protocolo GHOST (siglas de *Greedy Heaviest Observed Subtree*), la cual permite que los bloques huérfanos/caducados sean incluidos en la cadena de bloques y sean compensados [13, 5].

Como el QRL plantea utilizar un algoritmo tipo *proof-of-stake* desde el principio, esperamos utilizar de forma segura un tiempo de bloques entre 15 y 30 segundos.

8.3.2 Compensación de bloques

Cada nuevo bloque creado incluirá una transacción "coinbase" inicial que contendrá una dirección de minado en la cual existirá una compensación igual a la suma de la compensación coinbase y la suma combinada de las tarifas de transacción dentro del bloque. La compensación del bloque está ponderada con base en el saldo de la dirección del validador de participación seleccionada como selector de bloques.

La compensación de bloques es recalculada por el nodo de minería cada bloque y sigue el programa de emisión de monedas.

8.3.3 Tamaño de bloques

Para evitar controversias, se utilizará una solución adaptativa y práctica modelada de acuerdo a la propuesta de Bitpay para incrementar el tamaño de bloques con base en un múltiplo, x, de la mediana del tamaño, y, de los últimos z bloques [12]. El uso de la mediana previene que las transacciones de los mineros incluyan bien sea bloques vacíos o sobrecargados que alteren el tamaño promedio de bloques, x y z se convertirían entonces en las reglas obligatorias consensuadas que debe obedecer la red.

Por consiguiente, un tamaño de bloques máximo, b, podría ser calculado simplemente como:

$$b = xy$$

8.4 Unidad de divisa y denominaciones

El QRL utilizará una ficha monetaria, el *quantum* (plural *quanta*), como la unidad monetaria base. Cada *quantum* es divisible a un elemento más pequeño de la siguiente manera:

- 1: Shor
- 10^3: Nakamoto
- 10^6: Buterin
- 10^{10}: Merkle
- 10^{13}: Lamport
- 10^{16}: Quantum

De esta forma, cada transacción que involucre una fracción de un *quantum* en realidad es un número muy grande de unidades *Shor*. Las tarifas de la transacción son pagadas y calculadas en unidades *Shor*.

8.5 Cuentas

Los saldos de los usuarios se mantienen en cuentas. Cada cuenta es simplemente una dirección única y reutilizable del registro de contabilidad denotada por una cadena que comienza con "Q".

Se crea una dirección realizando un SHA-256 sobre la raíz de Merkle del árbol de certificación XMSS más grande. A esto se anexa una suma de verificación de cuatro bytes (formada a partir de los cuatro primeros bytes de un hash SHA-256 doble de la raíz de Merkle) y la letra "Q" como prefijo. Es decir, en pseudocódigo de Phyton:

$$Q + sha256(raíz\_merkle) + sha256\big(sha256(raíz\_merkle)\big)[:4]$$

Una dirección típica de cuenta sería:

Qcea29b1402248d53469e352de662923986f3a94cf0f51522bedd08fb5e64948af479

Cada cuenta tiene un saldo en términos de *quanta*, divisible hasta una unidad *Shor*.

Las direcciones tienen estado con cada transacción utilizando un par de claves OTS fresco y el QRL que almacena toda clave pública anteriormente utilizada (esto podría ser reducido dado que podría regenerarse sobre la marcha a partir de la firma de transacción y el mensaje, pero con un costo operativo intenso) para cada cuenta. Un contador de transacciones denominado nonce incrementa con cada transacción enviada desde una cuenta. Esto le permite a las carteras que no almacenan la cadena de bloques completa poder mantener un registro de sus ubicaciones en el esquema de firma con estado de hiperárbol de Merkle.

8.6 Emisión de monedas

8.6.1 Consideraciones históricas

Bitcoin fue la primera criptomoneda descentralizada e inicialmente experimental sin valor monetario, así que era apropiado distribuir la divisa completamente a través de minería. Recientemente Zcash ha elegido el mismo proceso, con un porcentaje de la base monetaria minada como recompensa en el período inicial de la emisión enviada al proyecto de código abierto – resultando en una increíble volatilidad del precio.

Otros registros de contabilidad como Etherium, en su lugar han vendido un amplio porcentaje del suministro monetario final como parte de una oferta inicial de monedas (ICO, por sus siglas en inglés). Este enfoque tiene el beneficio de que los primeros inversores tienen una ganancia potencial a través del soporte del proyecto, pero además el proyecto como tal es capaz de generar fondos para continuar su desarrollo, arrancando y desarrollándose a sí mismo desde sus orígenes. El enfoque ICO también le permite al mercado desarrollarse fácilmente, pues una mayor flota de monedas está disponible a los inversionistas para que compren y vendan a partir del bloque génesis.

Auroracoin (2014) tomó un enfoque diferente, ofreciendo a todos los participantes en Islandia un porcentaje igual de ICO, mientras que los desarrolladores mantuvieron el 50% de todo el suministro monetario para ellos mismos.

Otras criptomonedas han simplemente clonado a Bitcoin completamente, o empezaron desde cero con una nueva cadena pero una base de código diferente.

8.6.2 Transferencia de saldo intercadena

Es posible emitir el QRL con base en una captura del estado del registro de bitcoin actual en el bloque génesis de QRL. La idea general sería permitirle a los usuarios crear una transacción de "importación" de un solo uso que contenga un mensaje y una firma únicos (es decir, una dirección de cartera QRL generada aleatoriamente, firmada con una clave privada de bitcoin desde una dirección con un saldo bitcoin al momento de la captura). Esta característica podría permanecer activa hasta una cierta altura de bloque, y luego el remanente del suministro de moneda sería minado de la forma normal. El crecimiento inicial en el bloque génesis podría recortarse a la misma altura de bloque. Una desventaja con esto es que, a pesar de ser justo, penaliza a los propietarios de otras criptomonedas diferentes a bitcoin y técnicamente puede ser potencialmente difícil que nuevos usuarios lo hagan. Esto podría exponer permanentemente claves públicas hacia direcciones bitcoin utilizadas en el proceso, y sería importante mover los fondos después hacia una dirección bitcoin nueva generada aleatoriamente para mitigar esto.

(¿Podría permitirse la misma característica para los usuarios de Ethereum?)

8.6.3 Emisión propuesta – Borrador

La emisión inicial de QRL sería la siguiente:

- ICO de un millón de *quanta* (4,7% del suministro final de monedas) antes del lanzamiento.
- Una captura del estado de todos los saldos de direcciones bitcoin por encima de 0,01 btc usados para formar el bloque génesis QRL inicial. Cualquiera que decida transferir monedas directamente en una proporción 1:1 desde el registro contable de bitcoin al registro QRL podría hacerlo hasta que la altura del bloque alcance 518400 (3 – 6 meses) por medio de la cartera del nodo.
- Un monto adicional de 1 millón de *quanta* será retenido en una dirección de bloque génesis para el uso de la fundación.
- El suministro remanente será minado (21.000.000 – (2.000.000 + balances de btc importados cuando la altura de bloque alcance 518400)).

8.7 Cronograma de emisión de monedas

Una característica determinante de bitcoin es su escasez y su límite superior fijo de emisión de la ficha monetaria subyacente. QRL seguirá los pasos de bitcoin en este aspecto, con un límite superior fijo del suministro monetario de $21x10^6$ *quanta*. Una decaída exponencial suave en la compensación de bloques es favorecida hasta el tope definido de suministro de monedas. Esto eliminará la volatilidad asociada con el fenómeno de "disminución a la mitad" de bitcoin.

El suministro monetario total, $x = 21x10^6$, menos las monedas creadas en el bloque génesis, y, se reducirá exponencialmente desde Z_0 hacia abajo. La curva de decaída se calcula para distribuir las recompensas de minado por aproximadamente 200 años (hasta el año 2217 D.C., 420480000 bloques con tiempos de bloques de 15 segundos), hasta que un único *quanta* quede sin minar (aunque el minado podría continuar después de esta fecha).

El suministro remanente de monedas en el bloque *t*, Z_t, podría calcularse de la siguiente manera:

$$Z_t = Z_0 e^{-\lambda t}$$

El coeficiente λ es calculado con la expresión: $\lambda = \frac{\ln Z_0}{t}$. Donde *t* es el número total de bloques en el cronograma de emisión hasta el *quanta* final. Hasta el bloque 518400, λ = 3,98590885111x10^{-08}. La recompensa de bloque, *b*, es calculada para cada bloque con:

$$b = Z_{t-1} - Z_t$$

Entre el bloque génesis y el bloque número 518400, los saldos de bitcoin pueden ser transferidos al registro de contabilidad a través de transacciones de importación. En el bloque 518401, el cronograma de emisión fijará un nuevo objetivo y bloqueará los nuevos saldos importados, reduciendo así Z_t y ajustando las recompensas de bloques correspondientemente.

Referencias

[1] http://oxt.me/charts

[2] D. Bernstein. Sphincs: practical stateless hash-based signatures (*Sphincs: firmas prácticas sin estado basadas en hash*). 2015.

[3] J Buchmann. On the security of the winternitz one-time signature scheme (*Acerca de la seguridad del esquema de firmas de Winternitz de un solo uso*).

[4] J. Buchmann. Xmss - a practical forward secure signature scheme based on minimal security assumptions (*XMSS – un esquema de firmas práctico, seguro y hacia adelante basado en suposiciones mínimas de seguridad*). 2011.

[5] V Buterin. Ethereum whitepaper (*Artículo técnico de Ethereum*). 2013.

[6] A. Hulsing. W-ots+ - shorter signatures for hash-based signature schemes (*W-OTS+ - Firmas más cortas para esquemas de firma basados en hash*). 2013.

[7] A. Hulsing. Xmss: Extended hash-based signatures (*XMSS: firmas extendidas basadas en hash*). 2015.

[8] A Karina. An effocient software implementation of the hash-based signature scheme mss and its variants (*Una implementación eficiente en software del esquema de firma basado en hash MSS y sus variantes*). 2015.

[9] A. Lenstra. Selecting cryptographic key sizes (*Selección de tamaños de claves criptográficas*). 2001.

[10] R. Merkle. A certified digital signature. (*Una firma digital certificada*). CRYPTO, 435, 1989.

[11] S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system (*Bitcoin: un sistema de divisas electrónicas de pares*). 2008.

[12] S Pair. A simple, adaptive blocksize limit (*Un límite de tamaño de bloques simple y adaptativo*). 2016.

[13] Yonatan Sompolinsky. Accelerating bitcoin's transaction processing fast money grows on trees, not chains (*Aceleración del procesamiento de transacciones de bitcoin. El dinero rápido crece en los árboles, no en cadenas*). 2014.

[14] A. Toshi. The birthday paradox (*La Paradoja del Cumpleaños*). 2013.