2019

BITCOIN

Firmado por TITOS PAYAN JORGE - 75719231B el día 20/01/2019

Firmado por SANCHEZ SANCHEZ JESUS - 74739544D el día 20/01/2019

Miguel Santiago Cervilla

Firmado digitalmente por Miguel Santiago Cervilla
Fecha: 2019.01.20 20:13:42 +01'00'

CRIPTOGRAFÍA

MIGUEL SANTIAGO CERVILLA

JESUS SANCHEZ SANCHEZ

JORGE TITOS PAYAN

TEORIA DE CODIGOS Y CRIPTOGRAFIA | Justo Peralta López

Índice

BITCOIN

Un poco de historia:

El bitcoin es una moneda virtual, la cual es independiente ya que no está controlada por ningún banco, empresa o Estado de Gobierno. Se puede utilizar como medio de pago, al igual que el dinero físico, pero esta moneda es virtual, por lo que no es visible.
Esta moneda virtual tiene su origen en 2009, a través de Satoshi (aunque este autor no está confirmado ya que puede estar bajo una o varias personas), y su uso está controlado para la utilización de la criptografía, por lo que podemos decir que es una red criptográfica.

Antecedentes:

Antes del nacimiento de bitcoin, lo que se pretendía era impedir el "doble gasto" que significa que una misma moneda se utilizara para dos transacciones distintas al mismo tiempo. Esto se ha conseguido gracias a la implementación del protocolo bitcoin, el cual consiste en que los mineros tengan que realizar durante 10 minutos una prueba de trabajo (proof-of-work) que consiste en conseguir durante ese tiempo un hash único con el que se consiga el bloque actual.

Además, Adam Back había desarrollado hashcash, un sistema de prueba de trabajo para el control de spam.

Antes del bitcoin, había otras monedas que fueron creadas:

BMoney:
ECash: falló debido al crecimiento de uso de la tarjeta de crédito en el comercio electrónico.
E-Gold: sin el uso de la tarjeta de crédito, se implementó las transacciones instantáneas en una API.
Flooz: se hizo famosa por utilizarse para delitos con tarjetas de crédito y lavado de dinero.
Bitgold: fue una plataforma que se considera como la iniciadora al bitcoin, con un mecanismo de control inflación basada en el mercado.

Descripción de los distintos protocolos utilizados para Bitcoin

Los protocolos de Bitcoin se pueden englobar en una serie de protocolos TCP/IP construyendo una red de nodos superpuestos o lo que sería lo mismo red de pares P2P todos estos nodos serán los que consumen o proveen los servicios mientras mientras se colabora en un servicio de consenso.

¿Como funcionarían las transacciones de Bitcoin?

Se entiende por transacción el envío de bitcoins, todas estas transacciones serían simples registros guardados en cadena de bloques(BlockChain).

Una transacción está compuesta por una entrada, una cantidad y una salida.
La entrada y salida serían las direcciones para mantener la seguridad de Bitcoin, todo ello firmado con claves criptográficas para certificar su validez.

¿Como se envian Bitcoins?

Para enviar bitcoins necesitamos una dirección de Bitcoin y una clave privada.
La dirección de Bitcoin es la clave pública de la clave privada del destinatario.
La clave privada es generada a partir de una semilla que es generada según automáticamente según el software que utilices. Se trata de semillas complejas para evitar la coincidencia de claves.

¿Cuál sería la unidad más pequeña de un bitcoin y cuánto son sus equivalencias con monedas reales?

La unidad más pequeña de un bitcoin sería un “satoshi” en honor al creador de Bitcoin, el desconocido “Satoshi Nakamoto”.

Las equivalencias con las monedas más utilizadas a nivel mundial son las siguientes:

Bitcoin	Euros(EUR)	Dolares(USD)	Yenes Japoneses(JPY)	Libra de Gibraltar(GIP)
1	3281.86	3741.98	409185.04	2907.53

Como podemos observar 1 bitcoin se trata de una unidad muy elevada en comparación a cualquier tipo de moneda física.

Si lo comparamos al contrario con nuestra moneda local un euro equivaldría a 0,00031 Bitcoins actualmente.

Redes superpuestas de Bitcoin

Las redes superpuestas de Bitcoin son redes P2P (peer-to-peer), también conocida como red entre pares o red entre iguales es una red en las que casi todos o todos los aspectos funcionan sin que existan servidores o clientes fijos, sino que se sustenta a través de una serie de nodos que se relacionan entre sí como iguales. Cada nodo es servidor y cliente a la vez con respecto a los demás, por lo cual P2P es una red descentralizada. Este tipo de conexión hace posible que se realicen intercambios directos de información de cualquier tipo entre los usuarios interconectados. La propia Internet es una red pública y descentralizada sobre la cual se pueden implementar otras redes de distintos tipos (redes superpuestas), como es el caso de la P2P.

Las redes peer-to-peer obtienen un mayor rendimiento que las centralizadas, puesto que optimizan el ancho de banda de los usuarios por medio de la conectividad entre los mismos. El tráfico de información ocurre de una forma mucho más fluida que en las redes centralizadas, donde una cantidad relativamente pequeña de servidores provee de la totalidad del ancho de banda que utilizan todos los clientes de la red para una aplicación o servicio. Por esta razón, es posible que se realicen envíos de información a altas velocidades, lo cual ha resultado sumamente útil a compañías como Bitcoin.

Bitcoin se basa en una red peer-to-peer descentralizada, que permite que los usuarios posean dinero virtual y manejen las transacciones que realizan con el mismo de forma directa con cualquiera de los demás usuarios de la red, sin que intervenga una autoridad central, de hecho, es en este aspecto donde radica la gran diferencia entre las transacciones que se realizan con esta moneda virtual y el resto de los servicios de transferencia y pago online que tradicionalmente han sustentado la compra-venta en Internet.

Cuando se habla de Bitcoin, normalmente se hace referencia no solo a la moneda en sí, la cual es absolutamente virtual e intangible, sino también al protocolo y a la red P2P en la que se sustenta, la cual posibilita que la divisa sea regulada indirectamente por los propios usuarios a través de las transacciones que estos realizan, sin que exista un control sobre su valor y su producción.

Blockchain(Cadena de bloques)

Se trata de la tecnología en la que se basa Bitcoin para proporcionar transacciones seguras y confiables. Para ponerlo en términos sencillos, este registro de transacciones es una especie de libro de contabilidad virtual que sirve para anotar las transferencias de bitcoins y que todos pueden consultar. Para que una transacción sea incluida, esta debe cumplir con unos estándares que han de verificar los mineros.

Vamos a imaginar un fichero de texto con dos columnas, donde en una columna pone un identificador (por ejemplo “xyz”) y en la otra un número (por ejemplo “23”). Es decir “xyz” le corresponden “23”. Ahora imagina que ese fichero pudiera estar en miles de ordenadores duplicado, con la seguridad de que nadie lo puede alterar a traición pero cuando legítimamente se debe alterar algo, en cuestión de segundos, todos se sincronizan. Aunque uno de los miles de ordenadores desapareciese de la red no pasaría nada. Esto es lo que consigue Blockchain y aunque su magia es mucho más compleja y compuesta de más piezas como la criptografía, en esencia eso busca: un registro distribuido resistente a la sincronización y sin necesidad de confianza entre los miembros que la conforman.

En este ejemplo, los individuos que se ponen de acuerdo al explicar lo que han visto, vienen a ser nodos, estando geográfica y computacionalmente aislados los unos de los otros.

Al “detector de mentiras”, le enseñan un “proof of work”, es decir, un proceso criptográfico que prueba que un ordenador/chip y no otro ha resuelto un problema de forma correcta.

Falsificar una entrada en la cadena de bloques equivaldría a conseguir que más de la mitad de la gente se pusiese de acuerdo en mentir acerca de los detalles del aterrizaje del suricato de la misma manera, todos al mismo tiempo y sin tener la posibilidad de coordinar esa mentira previamente. Es decir, un “ataque” complicado de realizar.

Lo que se plasma en el blockchain no puede desaparecer jamás. Blockchain es un registro inmutable y permanente. Se trata de una base de datos que solo permite escritura. No se puede modificar ni borrar nada de ello, solo añadir, y todo ello bajo consenso.

¿Cómo se construye?
La cadena de bloques es un registro de todas las transacciones que tienen lugar “empaquetadas” en bloques que los mineros se encargan de verificar. Posteriormente serán incluidas en la cadena una vez validadas y distribuidas a todos los nodos que forman la red.

Bloques

Un bloque es un conjunto de transacciones confirmadas e información adicional que se ha incluido en la cadena de bloques. Cada bloque que forma parte de la cadena (excepto el bloque generatriz, que inicia la cadena) está formado por:

- Un código alfanumérico que enlaza con el bloque anterior.
- El "paquete" de transacciones que incluye.
- Otro código alfanumérico que enlazará con el siguiente bloque.

El bloque en progreso lo que intenta es averiguar con cálculos el punto anterior. Un código que sigue unas determinadas reglas para ser válido y sólo puede sacarse probando sin parar.

Mineros

Los mineros son ordenadores/chips dedicados que aportan poder computacional a la red de bitcoin para verificar las transacciones que se llevan a cabo.

Cada vez que alguien completa un bloque recibe una recompensa en forma de bitcoins (actualmente 25) y/o por cada transacción que se realiza.

Nodos

Un nodo es un ordenador/chip conectado a la red bitcoin utilizando un software que almacena y distribuye una copia actualizada en tiempo real de la cadena de bloques.

Cada vez que un bloque se confirma y se añade a la cadena se comunica a todos los nodos y este se añade a la copia que cada uno almacena.

Protocolo Proof-of-work:

Proof of Work (PoW, Prueba de trabajo en español), es un proceso que nace con el objetivo de eliminar los ataques a redes informáticas a través de la realización de una prueba moderadamente difícil (por ejemplo, un problema matemático) antes de permitir la realización de otra acción.

Actualmente es una pieza fundamental en las criptomonedas como Bitcoin, permitiendo alcanzar el consenso dentro de una red Blockchain.

Lo que se proponía desde un principio con este protocolo era el de crear un proceso por el cual se pudiese reducir el correo no deseado de Internet y los ataques DDOS, de forma que el origen tuviese que hacer un trabajo computacional previo que individualmente no tendría apenas coste eléctrico, pero que si intentabas replicarlo miles o millones de veces fuese económicamente inviable en relación coste / beneficio.

Gracias a la implementación de RPoW: Prueba de trabajo reutilizable, Bitcoin pudo ser descentralizado pudiendo tener un algoritmo seguro para llegar al consenso de Blockchain y evitar situaciones como el Doble Gasto o la manipulación de las cuentas, siendo posible confirmar las transacciones y organizar los bloques de forma ordenada.

Descripción de los criptosistemas o algoritmos de cifrado usados

SHA256: Primer algoritmo de minado en aparecer. Algunas transacciones resultan más lentas , pero trabaja de manera más concienzuda dejando poco margen para el error. Se usa para aquellos que desean una alta seguridad de los datos. El algoritmo SHA-256 utilizado por Bitcoin genera un hash casi único, con un tamaño fijo de 256 bits (32 bytes).El proceso de creación de una dirección pública de Bitcoin se inicia con un par de claves privada y pública de una curva elíptica ECDSA. Las direcciones públicas (por ejemplo, 31uEbMgunupShBVTewXjtqbBv5MndwfXhb) que vemos cuando utilizamos un cliente monedero/wallet Bitcoin han pasado un proceso de creación de clave pública y hashing en el que se utilizan las funciones hash SHA-256 y RIPEMD-160 para maximizar su seguridad.

Scrypt: Requiere menos energía que el SHA256 y procesa las transacciones de manera más rápida, ya que analiza los datos superficialmente.Una de las ventajas de minar con scrypt es que hay más cripto monedas que usan este algoritmo. También si no podemos comprar mineros Bitcoin por falta de stock, es una buena idea cambiar a Scrypt. Puede haber más unidades disponibles en venta de mineros en otras fases de producción. Su consumo es menor que SHA-256. Podemos transformar nuestra potencia scrypt y venderla para convertirla en Bitcoin. LiteCoin está subiendo de precio y por supuesto la potencia y demanda de mineros.

ScryptNf: Es una variable del algoritmo Scrypt.

X11: Consigue reducir el consumo de energía hasta en un 50 %, reduciendo así la temperatura que llegan a alcanzar los equipos. El algoritmo X11 funciona combinando 11 funciones *hash* individuales. Cuando se envía un valor, la primera función produce un *hash* que luego se envía a la siguiente función para producir otro hash. El algoritmo X11 se compone de las siguientes funciones *hash*:

- BLAKE
- DESEO DE MEDIA NOCHE AZUL (BMW)
- Grøstl
- J H
- Keccak
- Madeja
- Luffa
- CubeHash
- SHAvite-3
- SIMD
- ECO

ECDSA: Este algoritmo emplea operaciones sobre puntos de curvas elípticas en lugar de las exponenciaciones com el **DSA.Esto** trae como consecuencia dos grandes ventajas:

- Crea claves muy pequeñas pero con un altísimo nivel de seguridad.
- Criptoanálisis más complejo y por tanto más difícil de romper.

La criptografía de curva elíptica es extremadamente segura desde el punto de vista matemático. ECDSA utiliza claves más pequeñas y en general es una implementación más eficiente. ECDSA es por el momento el esquema utilizado por **Bitcoin**, **Ethereum** y la mayor parte de los protocolos blockchain.La magia de **ECDSA** es la que nos permitirá obtener una clave pública a partir de una clave privada. Con ambas claves ya podremos firmar transacciones y verificar luego las firmas.

CRYPTONIGHT: Una característica importante de CryptoNight es que es más difícil aumentar el poder de minería utilizando dispositivos **ASIC.** Desde hace un tiempo, los mineros se enfrentan a tal desafío y este acuerdo ayudará a mantener la minería distribuida, en lugar de un sistema casi centralizado.

La tecnología CryptoNote que utiliza el algoritmo hash CryptoNight ayuda a evitar el doble gasto. También asegura que los parámetros son adaptativos y resistentes al análisis del Blockchain. En esencia, el objetivo de CryptoNight es desarrollar la próxima generación de criptomonedas que son aún más 'cripto' y 'críptico' que las existentes.

Buscar en java dichas funciones y comentar su uso. Si no existiera en java buscar una en java que pueda realizar la misma función

Algoritmo SHA512:

```
 public class Sha512Hash implements Serializable, Comparable {
    private byte[] bytes;
    public static final Sha512Hash ZERO_HASH = new Sha512Hash(new
byte[64]);

    /**
     * Creates a Sha512Hash by wrapping the given byte array. It must be 64 bytes
long.
     */
    public Sha512Hash(byte[] rawHashBytes) {
        checkArgument(rawHashBytes.length == 64);
        this.bytes = rawHashBytes;

    }

    /**
     * Creates a Sha512Hash by decoding the given hex string. It must be 64
characters long.
     */
    public Sha512Hash(String hexString) {
        checkArgument(hexString.length() == 64);
        this.bytes = Hex.decode(hexString);
    }

    /**
     * Calculates the (one-time) hash of contents and returns it as a new wrapped
hash.
     */
    public static Sha512Hash create(byte[] contents) {
        try {
            MessageDigest digest = MessageDigest.getInstance("SHA-256");
            return new Sha512Hash(digest.digest(contents));
        } catch (NoSuchAlgorithmException e) {
            throw new RuntimeException(e);  // Cannot happen.
        }
    }

    /**
```

```
     * Calculates the hash of the hash of the contents. This is a standard operation in
Bitcoin.
     */
    public static Sha512Hash createDouble(byte[] contents) {
        return new Sha512Hash(Utils.doubleDigest(contents));
    }

    /**
     * Returns a hash of the given files contents. Reads the file fully into memory
before hashing so only use with
     * small files.
     * @throws java.io.IOException
     */
    public static Sha512Hash hashFileContents(File f) throws IOException {
        FileInputStream in = new FileInputStream(f);
        try {
            return create(ByteStreams.toByteArray(in));
        } finally {
            in.close();
        }
    }

    /**
     * Returns true if the hashes are equal.
     */
    @Override
    public boolean equals(Object other) {
        if (!(other instanceof Sha512Hash)) return false;
        return Arrays.equals(bytes, ((Sha512Hash) other).bytes);
    }

    /**
     * Hash code of the byte array as calculated by {@link
java.util.Arrays#hashCode()}. Note the difference between a SHA256
     * secure bytes and the type of quick/dirty bytes used by the Java hashCode
method which is designed for use in
     * bytes tables.
     */
    @Override
    public int hashCode() {
        // Use the last 4 bytes, not the first 4 which are often zeros in Bitcoin.
        return (bytes[63] & 0xFF) | ((bytes[62] & 0xFF) << 8) | ((bytes[61] & 0xFF) <<
16) | ((bytes[60] & 0xFF) << 24);
    }
```

```
@Override
public String toString() {
    return Utils.bytesToHexString(bytes);
}

/**
 * Returns the bytes interpreted as a positive integer.
 */
public BigInteger toBigInteger() {
    return new BigInteger(1, bytes);
}

public byte[] getBytes() {
    return bytes;
}

public Sha512Hash duplicate() {
    return new Sha512Hash(bytes);
}

@Override
public int compareTo(Object o) {
    checkArgument(o instanceof Sha512Hash);
    int thisCode = this.hashCode();
    int oCode = ((Sha512Hash)o).hashCode();
    return thisCode > oCode ? 1 : (thisCode == oCode ? 0 : -1);
}

public Sha256Hash trim256()
{
    byte [] result = new byte[32];
    for (int i = 0; i < 32; i++){
        result[i] = bytes[i];
    }
    return new Sha256Hash(result);
}
```

Algoritmo X11:

```
public class X11 {

    private static final Logger log = LoggerFactory.getLogger(X11.class);
    private static boolean native_library_loaded = false;

    static {

        try {
            System.loadLibrary("x11");
            native_library_loaded = true;
        }
        catch(UnsatisfiedLinkError x)
        {

        }
        catch(Exception e)
        {
            native_library_loaded = false;
        }
    }

    public static byte[] x11Digest(byte[] input, int offset, int length)
    {
        byte [] buf = new byte[length];
        for(int i = 0; i < length; ++i)
        {
            buf[i] = input[offset + i];
        }
        return x11Digest(buf);
    }

    public static byte[] x11Digest(byte[] input) {
        //long start = System.currentTimeMillis();
        try {
            return native_library_loaded ? x11_native(input) : x11(input);
            /*long start = System.currentTimeMillis();
            byte [] result = x11_native(input);
            long end1 = System.currentTimeMillis();
            byte [] result2 = x11(input);
            long end2 = System.currentTimeMillis();
            log.info("x11: native {} / java {}", end1-start, end2-end1);
            return result;*/
        } catch (Exception e) {
```

```
            return null;
        }
        finally {
            //long time = System.currentTimeMillis()-start;
            //log.info("X11 Hash time: {} ms per block", time);
        }
    }

    static native byte [] x11_native(byte [] input);


    static byte [] x11(byte header[])
    {
        //Initialize
        Sha512Hash[] hash = new Sha512Hash[11];

        //Run the chain of algorithms
        BLAKE512 blake512 = new BLAKE512();
        hash[0] = new Sha512Hash(blake512.digest(header));

        BMW512 bmw = new BMW512();
        hash[1] = new Sha512Hash(bmw.digest(hash[0].getBytes()));

        Groestl512 groestl = new Groestl512();
        hash[2] = new Sha512Hash(groestl.digest(hash[1].getBytes()));

        Skein512 skein = new Skein512();
        hash[3] = new Sha512Hash(skein.digest(hash[2].getBytes()));

        JH512 jh = new JH512();
        hash[4] = new Sha512Hash(jh.digest(hash[3].getBytes()));

        Keccak512 keccak = new Keccak512();
        hash[5] = new Sha512Hash(keccak.digest(hash[4].getBytes()));

        Luffa512 luffa = new Luffa512();
        hash[6] = new Sha512Hash(luffa.digest(hash[5].getBytes()));

        CubeHash512 cubehash = new CubeHash512();
        hash[7] = new Sha512Hash(cubehash.digest(hash[6].getBytes()));

        SHAvite512 shavite = new SHAvite512();
        hash[8] = new Sha512Hash(shavite.digest(hash[7].getBytes()));
```

```
        SIMD512 simd = new SIMD512();
        hash[9] = new Sha512Hash(simd.digest(hash[8].getBytes()));

        ECHO512 echo = new ECHO512();
        hash[10] = new Sha512Hash(echo.digest(hash[9].getBytes()));

        return hash[10].trim256().getBytes();
    }
}
```

Algoritmo ECDSA:

```
ECParameterSpec ecSpec = ECNamedCurveTable.getParameterSpec("B-571");
KeyPairGenerator g = KeyPairGenerator.getInstance("ECDSA", "BC");
g.initialize(ecSpec, new SecureRandom());
KeyPair keypair = g.generateKeyPair();
String publicKey = keypair.getPublic();
String privateKey = keypair.getPrivate();

//at sender's end
Signature ecdsaSign = Signature.getInstance("SHA256withECDSA", "BC");
ecdsaSign.initSign(privateKey);
ecdsaSign.update(plaintext.getBytes("UTF-8"));
byte[] signature = ecdsaSign.sign();

JSONObject obj = ...;

// at receiver's end
Signature ecdsaVerify = Signature.getInstance("SHA256withECDSA", "BC");
ecdsaVerify.initVerify(obj.getString("publicKey"));
ecdsaVerify.update(obj.getString("message").getBytes("UTF-8"));
boolean result = ecdsaVerify.verify(obj.getString("signature"));
```

Referencias:

http://www.finanzasparatodos.es/es/secciones/actualidad/bitcoin.html
https://rewisor.com/historia-bitcoin/
https://btcmarket.es/tag/antecedentes-de-bitcoin/
https://bitcoin.es/noticias/bitcoin-no-fue-la-primera-descubre-las-7-criptomonedas-anteriores/
http://www.metamercados.com/Content/peer-to-peer-explicado.html
https://blog.bitcoinvenezuela.com/que-es-bitcoin-y-como-funciona-el-protocolo-blockchain/
https://academy.bit2me.com/transacciones-bitcoin/
https://es.coinmill.com/BTC_calculator.html
https://www.goliat-mining.com/tipos-de-algoritmos-de-minado/
https://www.criptotendencias.com/base-de-conocimiento/eli5-algoritmos-criptograficos-en-la-blockchain/
https://bitcoiner.today/es/scrypt-el-algoritmo-con-eficiencia-y-potencia-que-puedes-minar/
https://libroblockchain.com/ecdsa/
https://www.programcreek.com/java-api-examples/index.php?source_dir=darkcoinj-master/core/src/main/java/com/hashengineering/crypto/X11.java
https://metamug.com/article/sign-verify-digital-signature-ecdsa-java.html