Simulación de una Blockchain utilizando la API Bouncy Castle

Álvaro Zavala, Member, IEEE
Unidad de Tecnología Informática
Universidad de Sonsonate
Sonsonate, El Salvador
alvarohz@gmail.com

Leonel Maye
Unidad de Informática
Ministerio de la Defensa Nacional
Sonsonate, El Salvador
maye@mdn.mil.sv

***Resumen*—*Blockchain* es una bitácora de acontecimientos digitales descentralizada, asegurada mediante criptografía y que solo puede ser actualizada por consenso de la mayoría de participantes en el sistema en el que está aplicado. Se garantiza que esta bitácora no pueda ser alterada a favor de unos pocos por medio de técnicas criptográficas. En este artículo se reporta cómo se puede simular una *blockchain* haciendo uso de la API Bouncy Castle mediante la cual se proveen las siguientes primitivas criptográficas: funciones hash y firma digital. Además se hace uso de una base de datos en MySQL en donde se generan todas las transacciones y un programa desarrollado en C# para minería, es decir, que por medio de éste se verifican las transacciones para que sean adheridas a la *blockchain*. Finalmente, otra aportación es una plataforma web que se encuentra escrita en C# donde se pueden efectuar transacciones entre usuarios y consultar la *blockchain*.**

***Index Terms*—Blockchain, Criptomoneda, SHA, Cartera, Transacción, Minería**

I. Introducción

Una cadena de bloques (*blockchain*) es una lista de registros en continuo crecimiento, llamados bloques, que están vinculados y asegurados mediante criptografía. Cada bloque contiene típicamente un hash criptográfico del bloque anterior, una marca de tiempo y datos de las transacciones. Por diseño, una cadena de bloques es resistente a la modificación de los datos. Es una bitácora abierta y distribuida que puede registrar transacciones entre dos partes de manera eficiente y de manera verificable y permanente [1]. Actualmente una *blockchain* está ampliamente utilizada en criptomonedas y ésta es solamente una de las aplicaciones que puede proporcionarnos una cadena de bloques. Porque su potencial es mayor, puede ser usada en el campo de bases de datos, en sistemas de gestión de activos digitales, notarios distribuidos, contratos inteligentes, al ser *blockchain* una red inalterable y fiable de datos, se podría utilizar para llevar la gestión contable de las empresas y gobiernos, para que se pueda ver si cumplen con las normas vigentes [2]. Uno de los sectores que estaban en contra de las criptomonedas es la banca, sin embargo, ahora se encuentran invirtiendo para conseguir cadenas de bloques propias con el fin de mejorar la fiabilidad de sus transacciones [3]. En este artículo se propone la simulación de una *blockchain* como la que utiliza Bitcoin a partir del uso de la API Bouncy Castle, la cual es una librería que tiene todos los algoritmos criptográficos necesarios para el proyecto, se ha optado por generar una simulación para comprender cómo funciona disminuyendo la complejidad. Los componentes desarrollados son: a) una base de datos para almacenar la *blockchain* con todas las operaciones, b) un programa que permite realizar la minería, comprobando las operaciones y así poder insertarlas en la *blockchain*, c) una plataforma web, un sitio dónde los usuarios puedan crear sus propias carteras y hacer operaciones entre ellos.

II. Preliminares

La seguridad de las criptomonedas está basada en criptografía de llave pública (Curvas Elípticas) e implementan la *blockchain* para resolver el problema del doble gasto y prevenir la modificación de transacciones previas, a continuación, se describen los bloques más importantes para esta tecnología.

II-A. Función Hash

Una hash es una función computacionalmente eficiente que relaciona cadenas binarias de longitud arbitraria a cadenas binarias de longitud fija, denominadas digestos [4]. Algunos de los usos criptográficos de las funciones hash son las firmas digitales y la integridad de datos.

II-B. Firma Digital

La firma digital es el resultado de una transformación criptográfica de datos que, cuando se implementa correctamente, proporciona un mecanismo para verificar la autenticación de origen, la integridad de los datos y el no repudio del signatario. Un algoritmo de firma digital incluye un algoritmo de generación de llaves, un algoritmo de firma y un algoritmo de verificación de firma. Un signatario usa el algoritmo de generación de firma y su llave privada para generar una firma digital sobre un mensaje; un verificador utiliza el proceso de verificación y la llave pública del signatario para verificar la autenticidad de la firma [4].

II-C. Cartera (Wallet)

Una Billetera o Wallet es un repositorio de direcciones de depósito, cada dirección tiene asociada un par de llaves privada y pública [5]. Las direcciones se derivan a partir de los digestos de las llaves públicas.

ISSN: 3061-8991

II-D. Transacciones

Se define una transacción como una transferencia de valor de un ente a otro, siendo este último conocido como beneficiario, entiéndase por ente a una dirección. El modelo se basa en un libro de contabilidad, la transferencia da como resultado saldos en las salidas [6]. El problema, por supuesto, es que el beneficiario no puede verificar que el ente que le transfiere no haya gastado dos veces la misma moneda. La *blockchain* es usada para solucionar el problema del doble gasto y prevenir la modificación de transacciones previas.

II-E. Bloques

Cada bloque representa un conjunto de transacciones confirmadas, que se va uniendo a la cadena, está formado por: Un código hash que enlaza el bloque anterior, las transacciones y otro hash que enlazará el siguiente bloque [6]. Cada bloque debe de ser validado por el resto de computadoras de la red, entre ellos los mineros.

II-F. Servidor de estampa de tiempo

Un servidor de estampa de tiempo trabaja tomando el hash de un bloque de ítems para sellarlos en el tiempo y notificar públicamente su hash. Cada estampa de tiempo incluye la estampa de tiempo previa en su hash, formando una cadena, con cada estampa de tiempo adicional reforzando al que estaba antes [5]. Básicamente es la *blockchain*

II-G. Mineros

El trabajo de los mineros consiste en que estos bloques sean validados, crear el hash correspondiente y unirlo al resto de la cadena de bloques. Cuando se realizan varias transferencias, estos mineros (computadoras) las van confirmando y las van añadiendo a lo que sería el siguiente bloque de la cadena, una vez tienen un bloque completo, toman el hash del último bloque para generar el hash del siguiente, de esta forma se asegura que el hash que se añadirá será único e intransferible (si se intentara insertar un bloque falsificado, el hash que produciría sería diferente del que debería de ir almacenado a la cadena y sería identificado como falso) [7]. En Bitcoin, cada vez que un minero crea un hash con éxito, se le recompensa con monedas virtuales.

II-H. Red (nodos)

Son las diferentes computadoras conectadas en la red de *Blockchain*. Cualquiera puede descargarse la cartera de *Blockchain* (por ejemplo, de Bitcoins) y contribuir a que la red sea más segura. Cada vez que se confirma un bloque nuevo en la red, se comunica a todos los nodos para que vuelvan a actualizar su cartera [7]. En el caso de que algún nodo quiera realizar una transacción, su *blockchain* local deberá estar actualizado y sincronizado con el resto de nodos.

Se debe mencionar que los conceptos anteriores son independientes de la moneda criptográfica a la que se haga referencia, por lo general estas se diferencian por los algoritmos criptográficos empleados para la generación de llaves, la firma, el cálculo de hashes.

III. Estado del Arte

Existen muchos frameworks para implementar *blockchains* a partir de las aplicaciones que esta tiene, entre ellos los siguientes:

- **Hyperledge** (o proyecto Hyperledger) es una plataforma código abierto para *blockchain*, iniciado en diciembre de 2015 por la Fundación Linux, para apoyar a los ledgers distribuidos basados en la *blockchain*. Los objetivos del proyecto son aunar un número de esfuerzos independientes para desarrollar estándares y protocolos abiertos, así como proporcionar un marco modular que soporte componentes diferentes para usos diferentes [8].
- **Ethereum** es una plataforma open source, descentralizada que permite la creación de acuerdos de contratos inteligentes entre pares, basada en el modelo *blockchain*. Cualquier desarrollador puede crear y publicar aplicaciones distribuidas que realicen contratos inteligentes. Ethereum también provee una criptomoneda que se llama 'ether' [9].
- **Bouncy Castle** es una colección de APIs utilizados en criptografía. Tiene versiones para los lenguajes Java y C#. La API de bajo nivel está optimizada para gestionar eficientemente los algoritmos criptográficos, de forma que se puedan usar en entornos de bajos recursos y provee todas las herramientas criptográficas involucradas para crear una *blockchain* [10].

IV. Metodología

Para el desarrollo de la aplicación y la determinación de requerimientos se utiliza como referencia la metodología ágil SCRUM, y basándonos en algunos de sus artefactos como: Pila del Producto, Pila del sprint y Sprint. [11]

IV-A. Visión del producto

La aplicación debe permitir a los usuarios el manejo de sus carteras electrónicas o wallets:

- Generar las carteras electrónicas
- Generación de usuarios.
- Puesta a disposición de los mineros, una cola de transacciones que permitan a los mineros cálculos los valores válidos para las nuevas transacciones.
- Guardado de la *blockchain*, como mecanismo de seguridad de las transacciones.
- Evitar el doble gasto.

V. Desarrollo del proyecto

La aplicación se divide en dos partes: aplicación de escritorio para la minería y aplicación web para el manejo de las carteras electrónicas, ambas programadas con el API Bouncy Castle, tanto para el cálculo de los picadillos y el manejo de los números grandes, la funcionalidad de cada una se describe a continuación.

V-A. *Descripción técnica*

La aplicación de minería permite conectarse al servidor, solicitar una transacción de la cola para la determinación de un picadillo que tenga un valor mayor en 10,000 que el valor anterior. Se usó este valor, porque en las pruebas realizadas con dos o tres mineros se necesitaban más de 24 horas para la determinación de un valor válido de nuevo bloque de la *blockchain* según Fig. 1. Los mineros no necesitan autenticación, solo identificarse con una cartera electrónica válida para conectarse al servidor. Los "centicoin" ganados serán acumulados en las correspondientes carteras electrónicas.

En cuanto a la web, los servicios ofrecidos son los siguientes:

- Crear usuarios.
- Un usuario puede crear diferentes direcciones.
- Un usuario puede enviar monedas a otros usuarios mediante su dirección pública.
- Un usuario puede vender monedas a un precio en concreto.
- Un usuario puede comprar monedas.

Cuadro I
ESPECIFICACIONES TÉCNICAS PARA LA WALLET ELECTRÓNICA

Algoritmo de firma	Elliptic Curve Digital Signature Algorithm (ECDSA)
Curva utilizada	Secp256k1
Función Hash para identificador privado	SHA3-256
Función Hash para identificador público	SHA3-256

De acuerdo a los tamaños de llaves y algoritmos sugeridos por el NIST [12][13][14][15] las especificaciones anteriores superan la seguridad mínima requerida en cada uno de ellos siendo como objetivo mínimo una seguridad de 128 bits.

V-B. *Diagrama de funcionamiento*

Cada semilla para generar el nuevo bloque debe asegurar que el hash resultante sea 10000 mayor que el bloque anterior. Encontrar este Hash es el trabajo que deben realizar los mineros

V-C. *Flujo de trabajo*

- Un usuario se registra y se le generan sus identificadores públicos y privados
- Luego de registrado se le asigna una cantidad de 100 sonsocoins
- El usuario puede comenzar a realizar transacciones en la plataforma que funciona como un banco, cada transacción es firmada digitalmente por el emisor
- Los bloques son de 10 transacciones y se confirman cada 5 minutos por los mineros
- Una vez hay transacciones sin confirmar los mineros comienzan a realizar cálculos
- Cuando un bloque es confirmado pasa a formar parte de la base de datos como parte de la *blockchain*.
- Todas las transacciones son consultables desde la plataforma y es posible comerciar con las monedas virtuales

Figura 1. Encontrar el valor válido para el nuevo bloque de la *blockchain*.

V-D. *Implementacíon con Bouncy Castle API*

La API Bouncy Castle cuenta con versiones en C# y Java, la versión en Java permite desarrollo multiplataforma, y entre las primitivas criptográficas ofrecidas están las siguientes: criptografía de llave pública, criptografía de llave privada, funciones hash, códigos de autenticación de mensajes (MAC), firma digital y estampa de tiempo.

Algunos puntos importantes de la librería son: énfasis en el cumplimiento de estándares y normas, amplia documentación disponible en internet e integración transparente con la librería nativa security de java.

Con respecto a las funciones hash y firmas digitales, primitivas usadas en el proyecto, se destaca lo siguiente:

- Entre las funciones hash esta SHA3, pero requirió hacer adecuaciones debido a que la entrada es de cierto tipo y tamaño, así mismo la salida debe ser codificada de acuerdo a los requerimientos de la aplicación.
- La clase que permite implementar firmas digitales cuenta con distintos algoritmos estándares como DSA y ECDSA, para el proyecto se utilizó el segundo. Entre las curvas disponibles están las señaladas por los estándares NIST y SEC, pero fue necesario escribir una implementación para procesar las entradas y salidas del algoritmo al formato necesario y de la misma forma para el algoritmo de verificación.
- La generación del par de llaves privada-pública fue acondicionada para integrarse con el algoritmo de firma y se utilizó el generador de números aleatorios que viene por defecto, además se codificó su salida.

En general para las implementaciones disponibles se requiere escribir código para acondicionarlo al caso específico de aplicación debido a que las entradas y salidas por defecto esperan cierto tipo y tamaño.

V-E. Emulación minería

Para la emulación de los mineros, se diseñó una aplicación de consola en C# que se puede ejecutar en cualquier PC que abre un hilo (thread) de procesador que se comunica por el puerto 3333 al servidor, consistente en una aplicación que se conecta a la base de datos, que del pool de transacciones pendientes obtiene la más antigua, con la finalidad de calcular un hash tal que al convertirlo en decimal sea 10000 mayor que el bloque anterior, usando una cadena codificada en formato JSON con los valores indicados en la Figura 1.

Es de hacer notar que cuando se ejecutan más mineros que números de hilos soportados por el procesador, el sistema completo se ralentiza.

Cada minero accede al último bloque confirmado de la *blockchain*. Este último bloque se usa como parámetro, junto a los datos de la nueva transacción para el cálculo del siguiente bloque.

VI. Experimentos y resultados

De acuerdo a las herramientas creadas el minero se ejecuta en cualquier computadora, este programa se conecta mediante TCP/IP a la plataforma web que almacena la *blockchain* y notifica la confirmación de bloques tanto a la plataforma como a los demás mineros como se muestra en Figura 2.

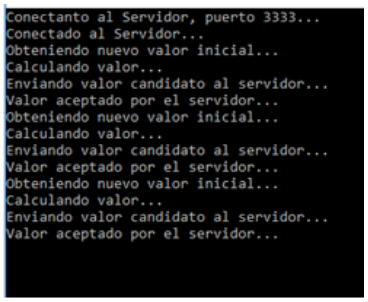

Figura 2. Minero ejecutándose por el puerto 3333.

En la base de datos se generan dos valores según Cuadro III debido a que obtiene la información tanto del que transfiere las monedas como del que las recibe.

Cuadro II
Tabla de transacciones

ID	*Dirección*	*Monto*
1	98db6b79acb71383b5a83e0bbc1cadd4	20
2	d94d81a75c0e8c0aef4e46a08206426b	20

También se cuenta con una ventana donde podemos vender las monedas. Vamos a la pestaña de Venta y seleccionamos nuestra dirección, la cantidad de monedas puestas a la venta y el precio por el cual queremos venderlas.

VII. Conclusiones

- Utilizar la API Bouncy Castle, dada su implementación a bajo nivel para manejo eficiente de los algoritmos criptográficos, requiere más líneas programación y por ende más tiempo, pero permite una mayor personalización de los procesos involucrados en una cadena de bloques, cuestión que en otras librerías como Ethereum o Hiperledge por su alto nivel de abstracción oculta más detalles de la implementación.
- La cadena de bloques propuesta proporciona un ejemplo de la implementación de una criptomoneda dónde se pueden apreciar las ventajas que la tecnología ofrece como la invariación en el tiempo, anonimato, la independencia de un ente central de confianza, por lo anterior las herramientas y código desarrollado pueden servir como base aplicable a futuras implementaciones, dado que no tiene dependencias de bibliotecas externas además de Bouncy Castle.
- En las pruebas realizadas se tuvo que calibrar los tiempos y cantidad de mineros necesarios según la capacidad del procesador y memoria disponible para encontrar los valores óptimos de funcionamiento de la cadena de bloques, comprobando que los recursos de cómputo disponibles y su optimización inciden en la implementación de la minería y rendimiento de la propuesta.

VIII. Recomendaciones

- Se recomienda que en un futuro proyecto se implemente un mecanismo para configurar carteras electrónicas descentralizadas.
- Implementar un mecanismo de consenso para evitar el doble gasto, desafío que se vuelve más difícil con carteras electrónicas descentralizadas.

Agradecimientos

Los autores agradecen enormemente a la universidad de Sonsonate y al Ministerio de la Defensa Nacional, por propiciar la investigación en temas de seguridad informática y especial agradecimiento a la Doctora Lil María Rodríguez Henríquez por su apoyo y asesoría en la investigación.

Referencias

[1] M. Iansiti, K. Lakhani, ”The Truth About Blockchain”, Harvard Business Review, 2017.
[2] CBInsights Research Portal, ”Banking Is Only The Beginning: 42 Big Industries Blockchain Could Transform”[online], disponible en: https://www.cbinsights.com/research/industries-disrupted-blockchain/.
[3] Reuters, Banks adopting blockchain ’dramatically faster’ than expected: IBM [online], disponible en: https://www.reuters.com/article/us-tech-blockchain-ibm-idUSKCN11Y28.
[4] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, CRC Press, 1996.
[5] S. Nakamoto, Bitcoin: un sistema de dinero en efectivo electrónico peer-to-peer [online], disponible en: https://bitcoin.org/files/bitcoin-paper/bitcoin_es.pdf.
[6] A. Narayanan, J. Boneau, E. Felten, A. Miller y S. Goldfeder, Bitcoin and Cryptocurrency Technologies, Princeton: Princeton University Press, 2016.
[7] A. Antonopoulos, Mastering Bitcoin: Unlocking Digital Cryptocurrencies, CA: O’Reilly, 2014.
[8] The Linux Foundation, Hyperledger [online], 2015, disponible en https://www.hyperledger.org/.
[9] Ethereum Foundation, Ethereum blockchain app platform [online], 2016, disponible en https://www.ethereum.org/.
[10] Legion of the Bouncy Castle, The Bouncy Castle Crypto API [online], 2015, disponible en https://www.bouncycastle.org/.

[11] Scrum Manager Body of Knowledge [online], 2018, disponible en: http://www.scrummanager.net/bok/index.php?title=Scrum_Manager_BoK.
[12] NIST, Digital Signature Standard (DSS) [online], disponible en: https://nvlpubs.nist.gov/nistpubs/fips/nist.fips.186-4.pdf.
[13] NIST, Secure Hash Standars (SHS) [online], disponible en: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.180-4.pdf
[14] Copia de fuentes de las aplicaciones web y de escritorio se encuentra en: http://sonsocoin.mil.sv/content/dist.zip.
[15] NIST, Recommendation for key management [online], disponible en: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-57pt1r4.pdf