Formación profesional en CePETel 2023

Desde la Secretaria Técnica del Sindicato CePETel convocamos a participar en el siguiente curso de formación profesional:

Introducción a blockchain y billeteras virtuales

Clases: 5 clases de 3 hs c/u de 18:00 a 21:00 hs.
Días que se cursa: jueves 18 de mayo; 1, 8, 15, y 22 de junio.
Modalidad: a distancia (requiere conectarse a la plataforma Zoom en los días y horarios indicados precedentemente).
Docente: NAHUEL DARÍO SÁNCHEZ

La capacitación es:

- **Sin cargo para afiliados y su grupo familiar directo.**
- **Sin cargo para encuadrados con convenio CePETel.**
- **Con cargo al universo no contemplado en los anteriores.**

Informes: enviar correo a tecnico@cepetel.org.ar

Inscripción (hasta el 15 de mayo 12:00 hs): ingresar al formulario (se recomienda utilizar una cuenta de correo personal)

https://forms.gle/1ZmzZYez6KqtPdxK7

Objetivos

Hace más de una década que la implementación Blockchain, junto con otras ciencias y tecnologías, están cambiando los paradigmas de procesamiento colaborativo y descentralizado de la información. Algunos se atreven a decir que Blockchain es una revolución tecnológica similar a la que se vivió con la llegada de Internet y la World Wide Web.

Las aplicaciones basadas en Blockchain van ocupando cada vez más espacios en todos los sectores y actividades, por ello es necesario entender de qué se trata.

En este curso introductorio se abordará el universo Blockchain actual; un confuso escenario lleno de conceptos como NFT, criptomonedas, smart contracts, DAOs, DAPPs, y demás, para tratar de organizarlos en un mapa mental y comprender qué lugar y función
cumple cada uno de ellos.

Adicionalmente se abordará el concepto de Billeteras Virtuales, también conocidas cómo “e-wallets”, las cuales durante los últimos años han crecido en su utilización de manera exponencial.

Ing. Daniel Herrero – Secretario Técnico – CDC

Temario

Módulo 1: Introducción
- Historia
- Ventajas y desventajas
- Tipos de Blockchains
- Trilema
- Redes p2p
- Conceptos principales
- Introducción a la criptografía
- Métodos de consenso

Módulo 2: Bitcoin
- ¿Qué es Bitcoin?
- Historia
- ¿Cómo funciona Bitcoin?
- Claves, direcciones y billeteras
- Transacciones
- La red de Bitcoin
- La cadena de Bloques (Estructura de la Blockchain)
- Minado y consenso

Módulo 3: Ethereum
- ¿Qué es Ethereum?
- Conceptos básicos
- Clientes
- Wallets y Transacciones
- Smart Contracts, Solidity
- Tokens
- Oráculos
- Aplicaciones descentralizadas (DAPPs)

Módulo 4: Otros aspectos
- Exchanges
- DeFi
- DAOs
- Los Tokens “más conocidos”
- “Shitcoins” y la explosión de ICOs
- NFTs
- Otras Blockchains

Módulo 5: Billeteras Virtuales
- Introducción
- Tipos
- Usos y aplicaciones
- Implementación

Ing. Daniel Herrero – Secretario Técnico – CDC

http://www.cepetel.org.ar ✉ tecnico@cepetel.org.ar Rocamora 4029 (CABA) (+54 11)35323201

- Aspectos de seguridad

Ing. Daniel Herrero – Secretario Técnico – CDC

http://www.cepetel.org.ar tecnico@cepetel.org.ar Rocamora 4029 (CABA) (+54 11)35323201

Introducción a blockchain y billeteras virtuales - Módulo 1

Docente: Nahuel Sánchez

*Este documento fue realizado en concepto de capacitación en Formación Profesional y dictada para el **Sindicato CePETel** a contar del mes de mayo del año 2023.*

Sindicato CEPETEL

Introducción a “Blockchain”

Módulo 1: Introducción

Introducción general al curso

Temario:

- Módulo 1 (18/05): **Introducción general a la tecnología Blockchain**
- Módulo 2 (01/06): **Blockchain y Bitcoin**
- Módulo 3 (08/06): **Aplicaciones Distribuidas y Ethereum**
- Módulo 4 (15/06): **Otros aspectos**
- Módulo 5 (22/06): **Introducción a billeteras virtuales**

Horarios y modalidad:

- 18Hs a 21Hs
- Teoría con algunos ejercicios y participación

Aclaraciones importantes

¿ Qué cosas **NO** trata este curso?

- Aspectos económicos específicos
- Trading
- Como “ganar dinero” con cosas relacionadas a Blockchain*
- Aspectos legales
- Inversiones (Comprar Bitcoin?, minar criptomonedas?, etc.)

Conocimientos **básicos** previos:

- Redes, programación

Introducción al módulo 1

Introducción general a la tecnología Blockchain:

- Historia
- Ventajas y Desventajas
- Tipos de Blockchains
- Trilema (Escalabilidad/Seguridad/Descentralización)
- Redes P2P
- Conceptos principales
- Introducción a la criptografía
- Métodos de consenso

En la escala de Elon. Cómo estás para Arrancar hoy?

Conociéndonos un poco mejor

Acerca de mi:

- Experiencia previa en investigación y seguridad informática
- Apasionado por la tecnología Blockchain
- Actualmente trabajando como Ingeniero de seguridad en Blockchain

Conociéndonos un poco mejor

Acerca de ustedes

- Ideas para presentarse:
  - Nombre
  - Ocupación / Cargo
  - ¿Por qué decidiste asistir al curso?

Introducción

Introducción

¿Qué es una Blockchain?

¿Qué es una Blockchain? - ¿Para qué sirve?

- Podemos definir “Blockchain” como un conjunto de datos (blocks) relacionados entre sí de forma “segura” que generan una “cadena” (chain).

- Permite a distintas partes que no confían entre sí (nodos) ponerse de acuerdo sobre el estado de cierta información.

Fuentes:

https://en.wikipedia.org/wiki/Blockchain
https://arxiv.org/pdf/1810.06130.pdf

Historia previa

- En 1979 Ralph Merkle publica la idea de “Merkle Tree”.
- En el paper de 1979 - “Computer systems established, maintained and trusted by mutually suspicious groups” aparecen muchas de las ideas fundamentales.
- En 1990 Haber y Stornetta aplican muchas de estas ideas para aplicar “timestamps” a documentos.
- En 1994 Haber y Stornetta crean la compañía “Surety”.
- En 2004 Hal Finney describe un prototipo de dinero digital (Reusable Proof of Work - RPoW)
- En 2008 se publica el famoso paper “Bitcoin: A Peer-to-Peer Electronic Cash System”
- En 2015 se lanza el proyecto de la blockchain Ethereum

Fuentes:
http://www.ralphmerkle.com/papers/Certified1979.pdf
https://arxiv.org/pdf/1810.06130.pdf
https://es.wikipedia.org/wiki/Reusable_Proof_Of_Work

Propiedades

- Normalmente usan almacenamiento distribuido
- Pueden usarse para almacenar cualquier tipo de información
  - Propiedad intelectual
  - Registros médicos
  - Información de stock / Cadenas de suministros
- Existen distintos tipos según por lo que se las catalogue
  - Públicas / Privadas (Acceso a la información)
  - Sin permisos / Con permisos (Posibilidad de agregar información)
  - Mezclas de las opciones anteriores

Fuentes:

https://supplain.io/news/blockchain-applications-use-cases

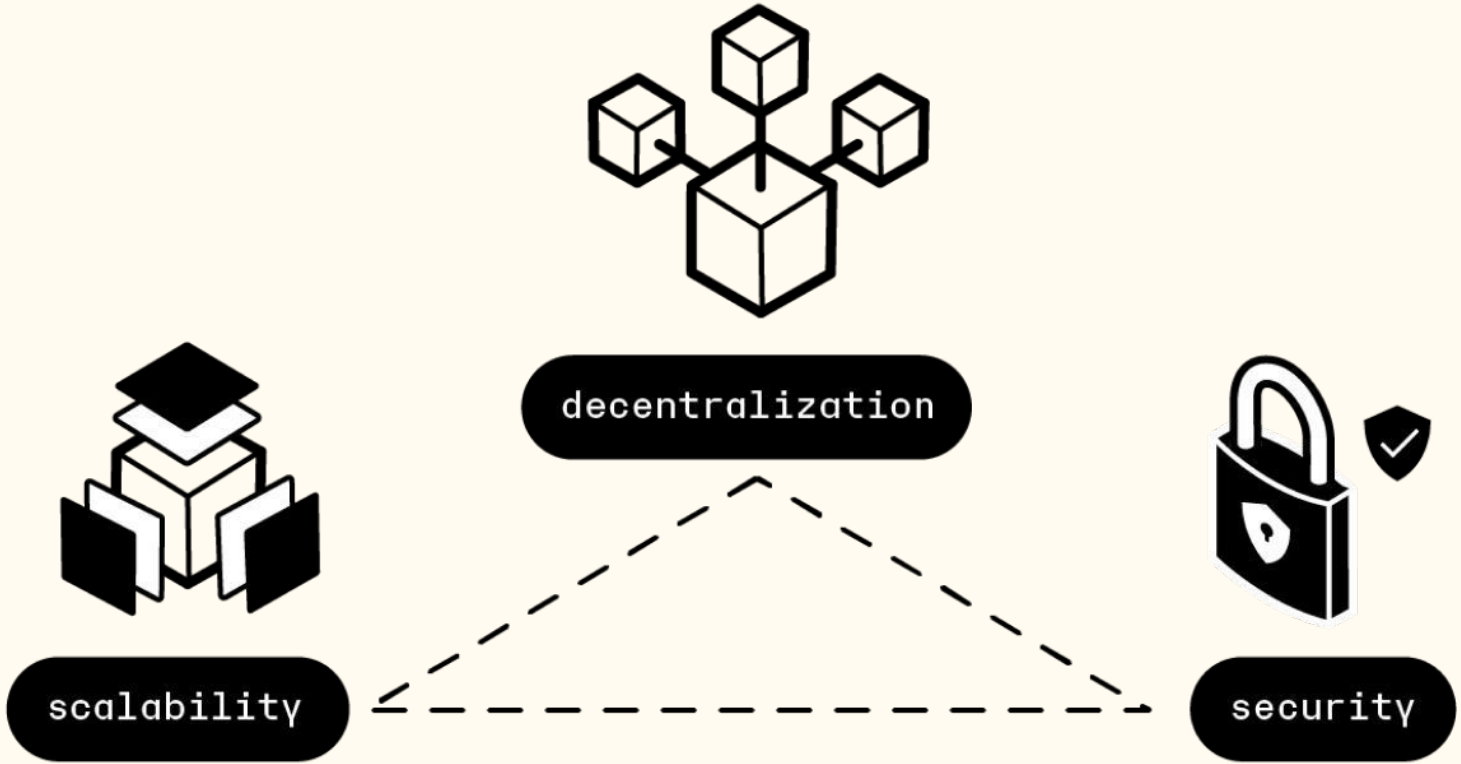
¿Qué ventajas tiene la tecnología Blockchain?

- Descentralización
- Confianza en la integridad de la información
- Transparencia
- Posibilidad de verificar toda la información

¿Qué desventajas tiene la tecnología Blockchain?

- Alto consumo de energía *
- Dificultad en la escalabilidad
- Falta de confianza del público general
- Regulación

Trilema Blockchain



https://www.ledger.com/wp-content/uploads/2021/11/cover-14.png

Tipos de Blockchains

Tipos de Blockchains - Públicas

- Cualquier participante puede (potencialmente) agregar bloques
- La política de consenso determina que estado es válido
- Normalmente son más lentas que las Blockchains permisionadas
- Son más seguras en comparación con sus contrapartidas
  - Descentralizadas *
  - Escalables
  - No se requiere confiar en una entidad
  - Alta tolerancia a fallas
- Ejemplos:
  - Bitcoin
  - Ethereum
  - LiteCoin

Tipos de Blockchains - Privadas

- Solamente nodos autorizados pueden participar
- Normalmente funcionan en redes privadas
- Generalmente usadas por empresas
- Ventajas:
  - Velocidad
  - Privacidad de la información
- Desventajas:
  - Centralización
  - Baja tolerancia a fallos

Tipos de Blockchains - Híbridas

- Partes permisionadas y partes no, por ejemplo, libre acceso a leer información pero restringido el agregado de nuevos bloques
- Normalmente controladas por una sola entidad/empresa
- Ejemplos
  - IBM Food Trust

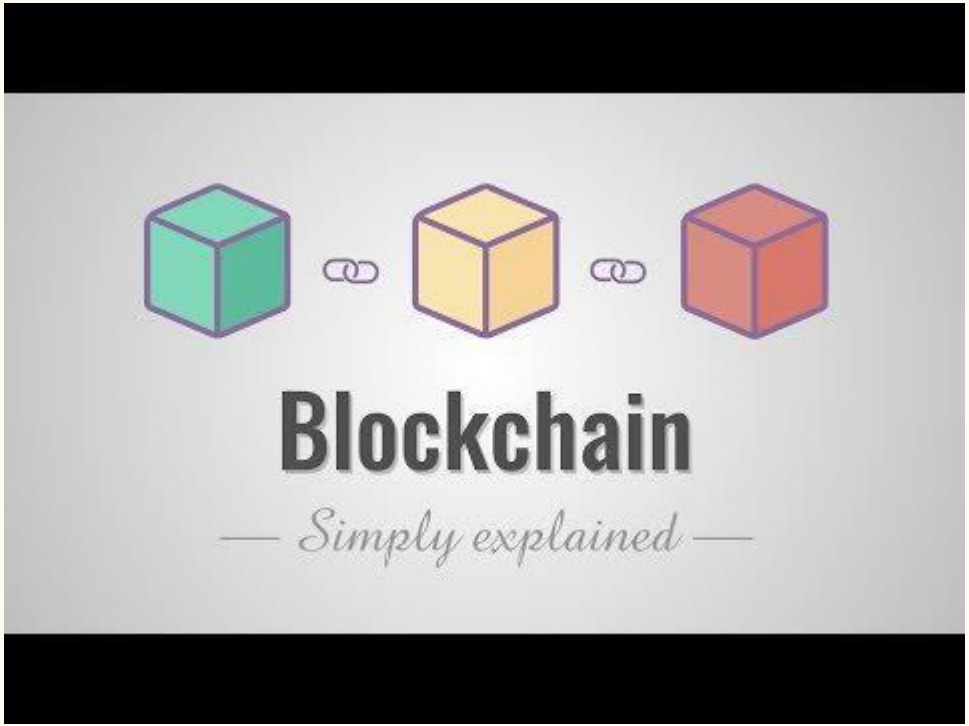
Tipos de Blockchains - Consórcio/Federadas

- Blockchains controladas por más de una entidad/compañía
- Ventajas
  - Participantes conocidos y validados
  - Control sobre quién puede acceder a la red
  - Bajo costo (en comparación con otros tipos de Blockchain)
- Ejemplos de uso:
  - Logística
  - Seguros / Medicina
  - Bancos / Finanzas

Preguntas

- ¿Qué tipos de Blockchain recordás?
- ¿Qué ventajas tiene el uso de la tecnología Blockchain?
- ¿Qué características compiten en el trilema de las Blockchain?

¿Cómo funciona una Blockchain?

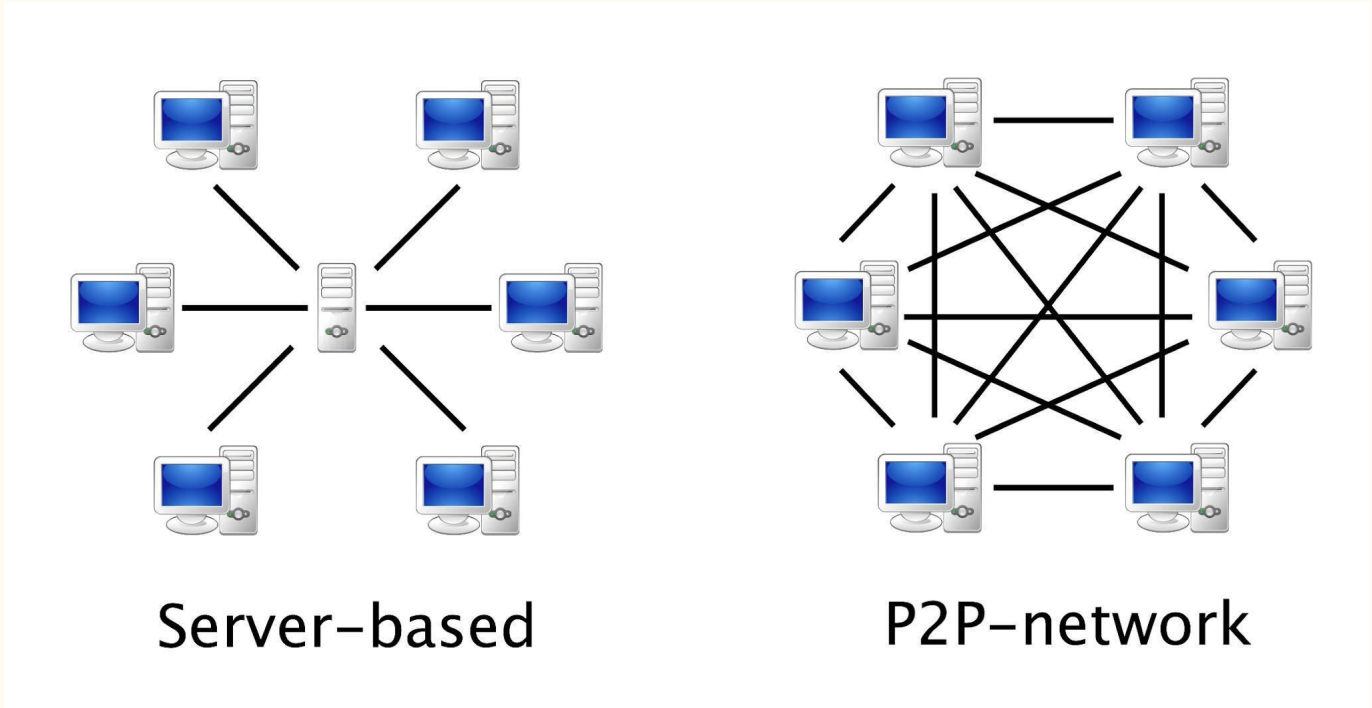


Redes P2P

Redes: Cliente-Servidor / Peer To Peer

- La mayoría de Blockchains se basan en la idea de red P2P
  - Basadas en la descentralización
  - La red se constituye de “nodos” (peers)
  - Todos los nodos son iguales en relación a sus privilegios
- Otras tecnologías previamente usaron redes de este tipo
  - Intercambio de archivos
  - Almacenamiento distribuido
  - Procesamiento distribuido
- La arquitectura de la red y su protocolo tendrá peso en:
  - La seguridad de la red: Algoritmo de consenso
  - Privacidad de las transacciones
- Cada tipo de Blockchain decide qué protocolos usar en todo su “Stack”

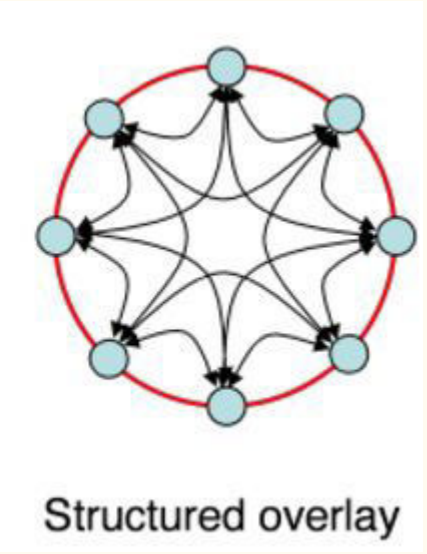
Redes: Cliente-Servidor / Peer To Peer



https://proyectoidis.org/wp-content/uploads/1969/04/p2p-network-vs-server.jpg

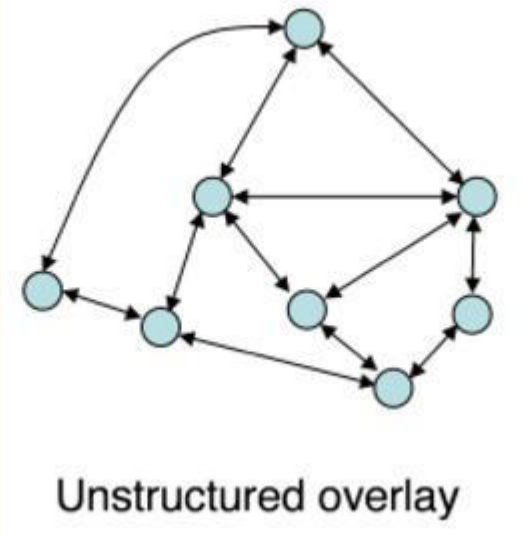
Redes P2P: Estructuradas

- Existen una organización la cual los nodos conocen
- Existe cierta “centralización”
- Más complejas de instalar y mantener
- Proveen acceso más simple a la información



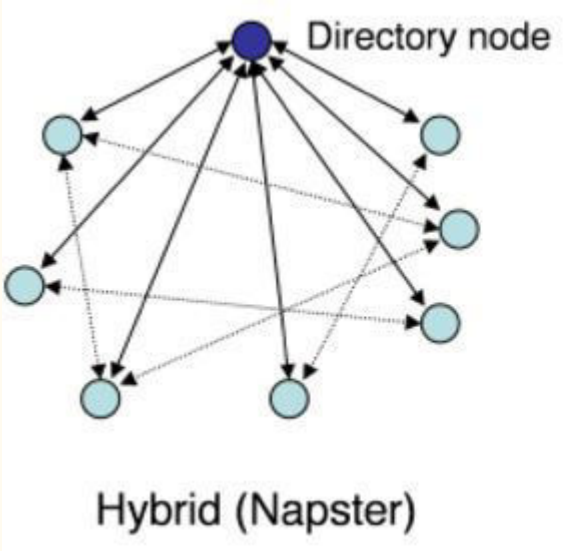
Redes P2P: No estructuradas

- La conexión entre los nodos se realiza de manera arbitraria
- Son más fáciles de construir y mantener
- Requieren de mayor cantidad de tráfico para funcionar



Redes P2P: Híbridas

- Mezcla de red P2P con arquitectura cliente-servidor
- Existe un “índice” mantenido por un servidor central
- Tienden a ser más performantes



Redes P2P y Blockchain, resumen

- La información se distribuye entre todos los nodos
- Todos los nodos son iguales
- No existe una autoridad central
- Un nodo nuevo puede obtener una copia completa gracias a otros nodos
- Todos los nodos pueden validar la información enviada por un nodo
- Permite confiar en “el resultado” sin confiar en “los participantes”

Conceptos principales

Conceptos principales

- Funciones de “Hash”
- Nonce
- Transacciones
- Bloques

Funciones de Hash

Funciones de Hash - Introducción

- Utilizadas por todas las variantes de tecnología Blockchain
- Se le aplica una función a una entrada y se obtiene una salida
  - La entrada es arbitraría (cantidad de datos, tipo, etc.)
  - La salida tiene un largo fijo
- Permite a cualquier persona tomar una entrada y obtener la misma salida
  - Valida que para la misma entrada se recibe la misma salida
  - Prueba que no hubo cambios en la entrada
- Ante el mínimo cambio en la entrada la salida cambia completamente
- Términos:
  - Entrada - Input
  - Salida (resultado) - Digest

Funciones de Hash - Propiedades

- Resistentes a pre-imágenes (Dado el digest de F(x), imposible obtener x)
- Resistentes a segunda pré-imagen
- Resistentes a colisiones
- Dado que hay infinitas entradas y un número finito de “Digests” las colisiones son posibles pero muy improbables
- Efecto avalancha
- Determinismo

Funciones de Hash - Usos en Blockchain

- Derivación de direcciones
- Creación de identificadores únicos
- Aseguramiento de la información contenida en bloques
- Aseguramiento del encabezado de bloque / block header

Algunos ejemplos de funciones de Hash

- MD5
- SHA1
- SHA2
- **Keccak**

Message Digest-Algorithm 5 (MD5)

- Diseñado por Ronald Rivest en 1995
- Muy utilizado para realizar verificaciones de archivos dada su rapidez
- Actualmente considerado como **NO** seguro
- Output de 128 bits (16 bytes)
- El output normalmente se expresa en formato hexadecimal (32 caracteres)

MD5("Generando un MDS de un texto") = e14a3ff5b5e67ede599cac94358e1028

Secure Hash Algorithm 1 (SHA1)

- Similar a MD5
- Salida de 160 bits (20 bytes)
- Actualmente “roto”
  - Se conocen ataques prácticos para generar colisiones
  - Investigación de Google+CWI
  - https://shattered.it/

SHA1(“Blockchain”) = efe3fbaa6db3f43cf45d8ee3fdb168cd448afa41

SHA2 (224, 256, 384, 512)

- Creado por la NSA y publicado por el NIST en 2001
- Similar en su naturaleza a otros algoritmos de la familia SHA
- 2^{256} posibles outputs
- https://sha256algorithm.com/

sha256(‘Blockchain’) = ef7797e13d3a75526946a3bcf00daec9fc9c9c4d51ddc7cc5df888f74dd434d1

Keccak / Secure Hash Algorithm 3 (SHA3)

- Keccak nombre original del algoritmo
- Sutiles diferencias entre Keccak y SHA3
- No reemplaza a SHA2
- Keccak es muy utilizado en Ethereum

Keccak(“Blockchain”) =
fa8871e962875d078135f1c5b27b0f184ab6f4dff8641dd81032226ea0ae9e8c

Preguntas

- ¿Qué tipo de Redes P2P recordás?
- ¿Qué es y para qué sirve una función de Hash?
- ¿Qué algoritmo de Hash se considera inseguro hoy en día?

Nonce

Nonce (Number used ONly onCE)

- Un número arbitrario que sólo se debe utilizar una vez
- Utilizado para forzar la variación de la entrada en una función de hash

$$\mathbf{Hash(NONCE + Data) = Digest}$$

Sindicato CEPETEL

Transacciones

SECRETARÍA TÉCNICA

Transacciones (TXs)

- Interacción entre partes
  - Transferencia de valores
  - Ejecución de programas
  - Transferencia de información
- Contenidas en **bloques**
- La información contenida en una TX depende de la implementación
  - Emisor
  - Clave Pública del emisor
  - Firma digital
  - Inputs y Outputs

Transacciones (TXs)

- Son la “columna vertebral” de las Blockchain
- En módulos futuros veremos sus propiedades en el contexto de cada blockchain

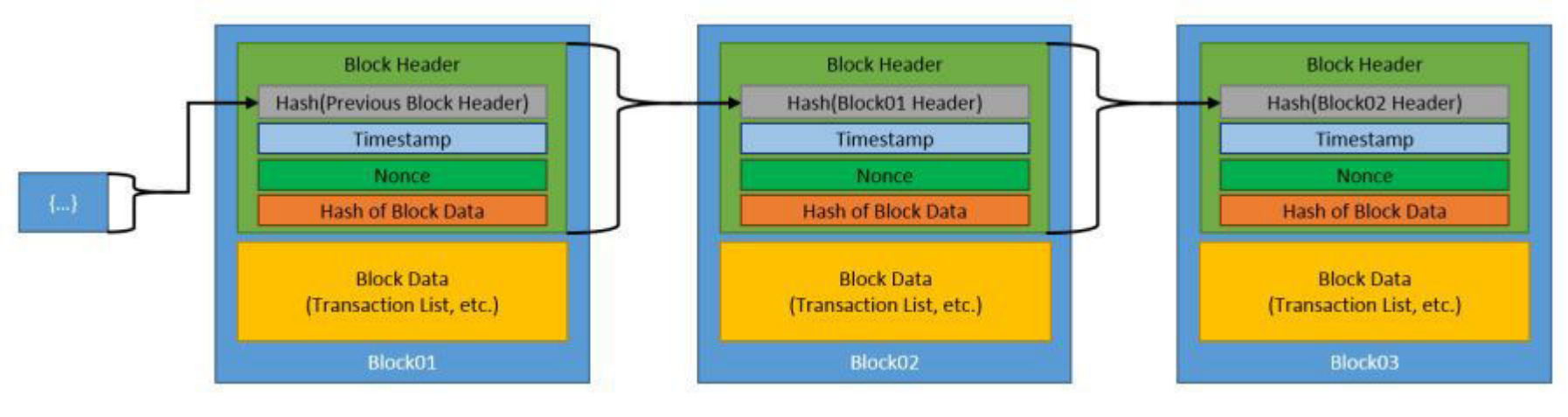
Bloques

Bloques

- Agrupación de TXs
- Contiene una cabecera o “header”
  - En esta parte se incluye la “metadata” del bloque
    - Timestamps
    - Información del bloque anterior
    - Número de bloque
    - Hash que representa la información incluida en el bloque
    - Nonce
- Las TXs incluidas son validadas previamente

Encadenando bloques

- Cada bloque tiene el hash del bloque anterior



Breve introducción a la criptografía

Conceptos principales

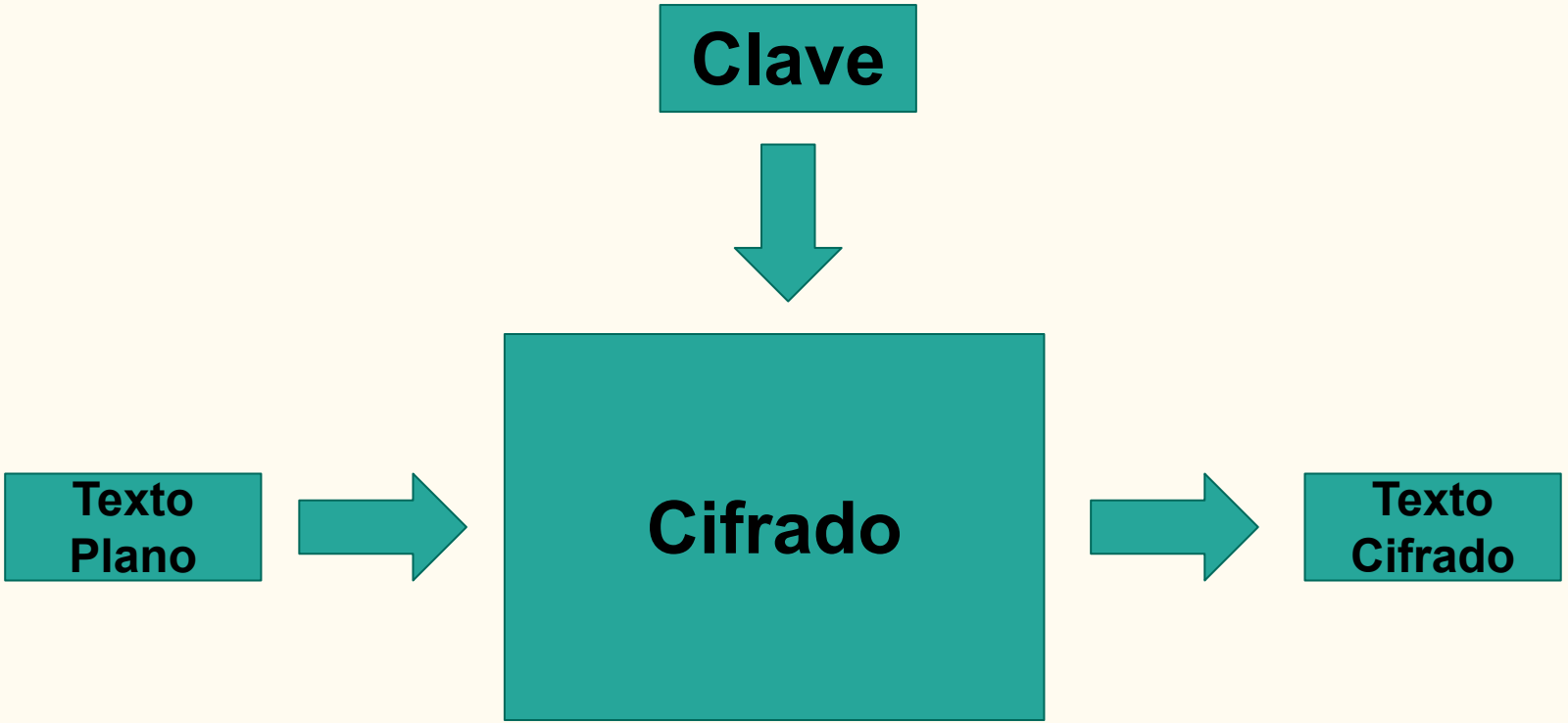
- Introducción
- Hashes (ya los vimos)
- Criptografía simétrica
- Criptografía asimétrica o de clave pública/privada
  - Cifrado
  - Firma

Introducción

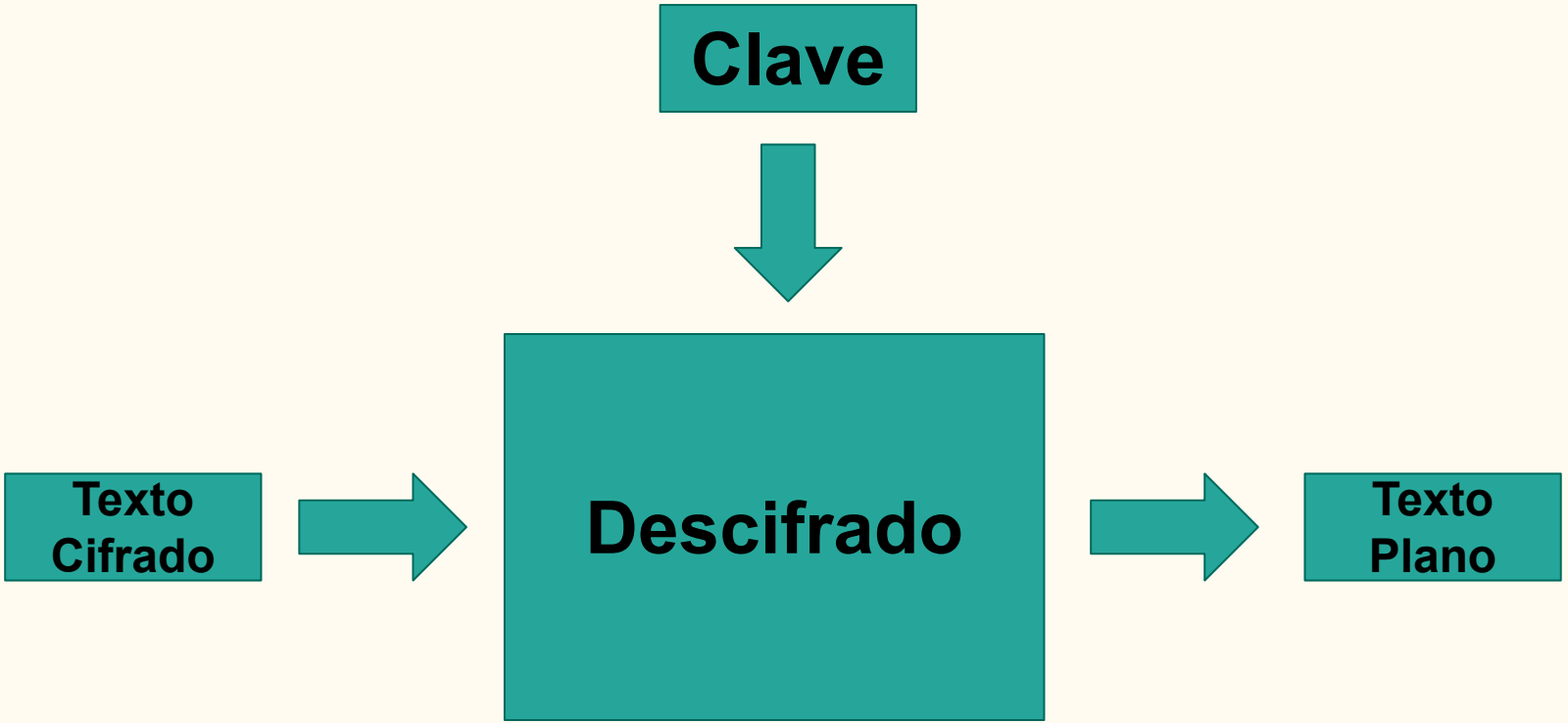
Criptografía - Introducción

- Métodos/protocolos para resguardar información privada
- Términos comunes
  - Encryption / Cifrar
  - Decryption / Descifrar
  - Cipher / Algoritmo de cifrado
  - Key / Clave
  - Plaintext / Texto plano
  - Ciphertext / Texto cifrado
- Provee confidencialidad a través del cifrado
- La seguridad del cifrado NO debe depender de que este sea secreto

Cifrado simétrico



Cifrado simétrico



Algoritmos de cifrado simétrico - Ejemplos

- Cesar (Caesar Cipher)
- Vigenere
- One-Time Pad
- AES
- 3DES

Ejemplo OTP

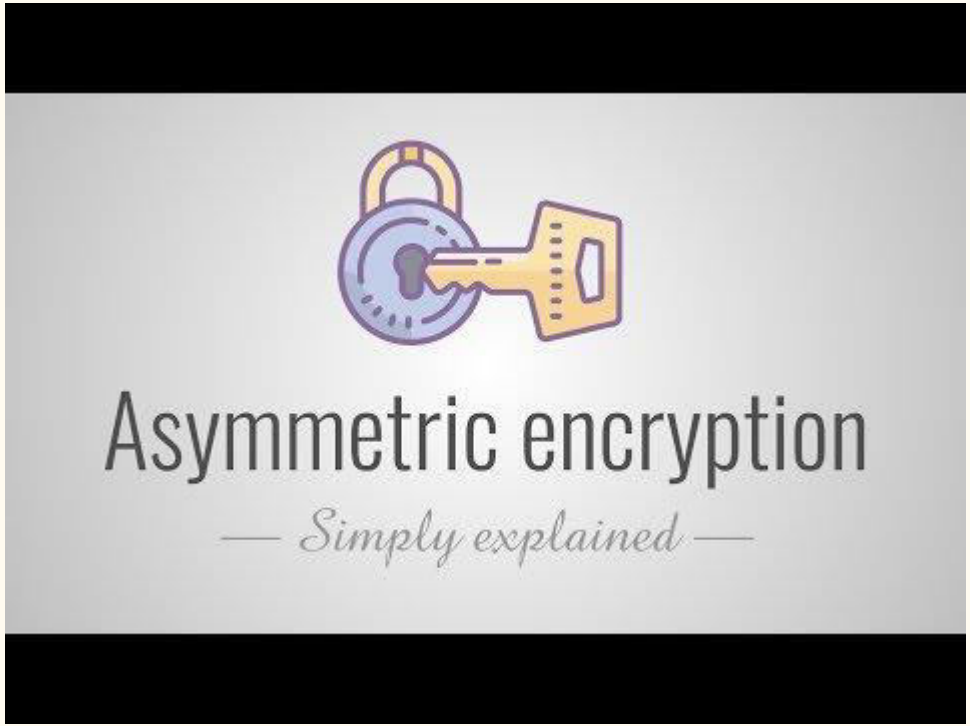
Key = “10110100”

P = “01101101”

C = “11011001”

Cifrado asimétrico

- Se utilizan dos claves
  - Clave pública (cifrado)
  - Clave privada (descifrado/firma)
- La clave pública puede derivarse de la privada (no así al revés)
- Se basa en la asimetría de ciertas operaciones matemáticas
  - Factorización de números primos grandes
- RSA es el uno de los algoritmos más conocidos
- Hoy en día se utiliza la criptografía de Curvas Elípticas
  - Permite obtener el par de claves pública/privada
- Este tipo de criptografía es fundamental para Blockchain


Asymmetric encryption
Simply explained

Clave pública - Derivación de direcciones

- Concepto de “Dirección” utilizado por muchas Blockchains
- Conjunto de caracteres alfanuméricos
- Derivada de la clave pública
- En redes no-permisionadas es posible crear una cantidad arbitraria de direcciones

Clave Pública -> Función de Hash -> Dirección

Métodos de consenso

Algunas preguntas

¿Por qué nodos que no confían entre sí y que compiten por producir un bloque compartirán información?

Si se producen dos bloques al mismo tiempo. ¿Cuál se considera válido como “siguiente bloque”?

Métodos de consenso

Métodos de consenso

- Determina quién publicará el siguiente bloque
  - Normalmente es una competencia entre distintos nodos
- Generalmente se premia al nodo ganador
  - Pago a través de criptomonedas
  - Pago por transacciones incluidas
- Cabe destacar que normalmente el objetivo es económico

Métodos de consenso

- Los usuarios que ingresan a la red, se ponen de acuerdo sobre el estado inicial de la misma
- Los nodos aceptan el método de consenso utilizado
- Todos los bloques subsiguientes son agregados después, deben ser válidos y respetar el algoritmo de consenso
- En la práctica todo esto está resuelto vía software
- En redes permisionadas algunas cosas pueden ser distintas
    - Solo algunos nodos pueden publicar nuevos bloques
    - Métodos de consenso menos seguros pero más rápidos

Prueba de Trabajo / PoW

Proof of Work (PoW)

- Para publicar el próximo bloque se debe resolver un problema computacionalmente complejo
- El problema a solucionar es difícil de resolver pero fácil de verificar

¿Ejemplos?

Proof of Work (PoW)

- Un ejemplo bastante común de problema a resolver es el siguiente:

“ Se requiere que el *digest* resultante de aplicar una función de hash al bloque sea menor que un valor X”

Proof of Work (PoW)

EJEMPLO PRÁCTICO

SHA256(“blockchain” + Nonce) = Digest que empiece con “000000”

Proof of Work (PoW)

SHA256(“blockchain” + Nonce) = Digest que empiece con “**000000**”

1. SHA256(blockchain0) = 0xbd4824...
2. SHA256(blochchain1) = 0xdb0b9c...
3. ...

10730895. SHA256(blockchain10730895) = 0x000000...

Proof of Work (PoW)

- Es sumamente común que varios nodos se agrupen en colectivos (pools)
- El tipo de trabajo a resolver debe poder ser dividido
- Veamos el siguiente ejemplo:
  - Nodo 1: validará Nonces del 0 al 1000000
  - Nodo 2: validará Nonces del 1000001 al 2000000
  - Nodo 3: validará Nonces del 2000001 al 3000000
  - Etc.
- De esta forma el trabajo puede ser distribuido y resuelto más rápidamente

Prueba de participación / PoS

Proof of Stake (PoS)

- Basado en la idea de que: “Cuanto más un usuario tenga invertido en un sistema, más querrá que el sistema funcione y menos querrá afectarlo”
- Está normalmente asociado a depositar algún tipo de valor en el sistema
- Dependiendo la implementación el valor puede ser fijo o no
- En este modelo no se requieren resolver problemas de cálculo intensivo
- Tienden a consumir menos energía

Proof of Stake (PoS)

- La selección de quien publicará el próximo bloque depende de la implementación
  - Aleatorio con peso
  - Aleatorio sin peso
  - En base a votos
  - Por antigüedad
  - Delegación
- Independientemente de la variante cada nodo debe “depositar” una cierta cantidad de algún recurso

Proof of Stake (PoS)

- Ejemplo: Ethereum luego del “Merge”

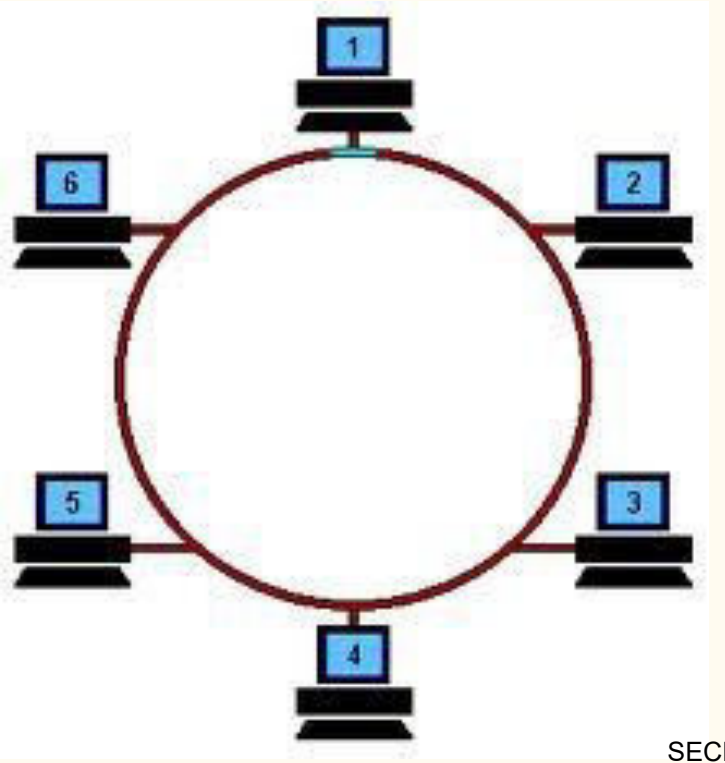
Proof of Stake (PoS)

Todos contra Todos / Round Robin

Round Robin

- Normalmente utilizado en redes permisionadas
- Existen turnos y la creación de bloques se ve repartida
- No se requiere resolver problemas
- Consume poca energía
- **Requiere que se confíe en todos los nodos participantes**

Ejemplo



- Topología Token Ring
- Un “Token” pasa de nodo en nodo
- Todos los nodos participan

Sindicato CEPETEL

Prueba de Autoridad / Proof of Authority

SECRETARÍA TÉCNICA

Proof of Authority

- También conocido como “Prueba de Identidad”
- Se basa en una confianza sobre la entidad detrás del nodo
- La entidad operando el nodo “se juega su reputación”
- Modelo utilizado en redes permisionadas

Sindicato CEPETEL

Prueba de Tiempo Transcurrido / Proof of Elapsed Time

SECRETARÍA TÉCNICA

Proof of Elapsed Time (PoET)

- Cada nodo espera un tiempo aleatorio para publicar su bloque
- Cuando un nuevo bloque es publicado cada nodo resetea su temporizador y el proceso comienza de nuevo
- Requiere de hardware específico para funcionar
- El software implementando el cliente puede interactuar con este hardware de forma segura
- El nodo puede probar la correcta operación mediante certificados firmados por el hardware

¿Preguntas?

Introducción a blockchain y billeteras virtuales - Módulo 2

Docente: Nahuel Sánchez

*Este documento fue realizado en concepto de capacitación en Formación Profesional y dictada para el **Sindicato CePETel** a contar del mes de mayo del año 2023.*

Introducción a “Blockchain”

—

Módulo 2: Bitcoin

Introducción al Módulo 2

- ¿Qué es Bitcoin?
- Historia
- ¿Cómo funciona Bitcoin?
- Claves, direcciones y billeteras
- Transacciones
- La red de Bitcoin
- La cadena de Bloques (Estructura de la Blockchain)
- Minado y consenso

¿Qué es Bitcoin?

- Conjunto de conceptos y tecnologías que forman un ecosistema de “dinero digital”
  - Unidad de valor
  - Protocolo
- Los usuarios pueden utilizar Bitcoin para las mismas cosas que utilizan monedas convencionales
- Los bitcoins pueden comprarse, venderse o cambiarse en “exchanges”
- Bitcoin es completamente descentralizado*, no existe el concepto de servidor central o empresa manteniendo la red

¿Qué es Bitcoin?

- Alrededor de cada 10 minutos se genera un nuevo bloque que valida las transacciones realizadas en esa ventana de tiempo
- El nodo que publicó el bloque recibe una recompensa pagada en Bitcoin
- El protocolo se “autoregula” y la dificultad para minar bloques se ajusta para respetar la ventana de 10 minutos
- Además la producción de Bitcoins también está regulada y disminuye cada 4 años
- Moneda deflacionaria

¿Qué es Bitcoin?

Al margen del uso financiero Bitcoin representa la culminación de décadas de investigación en criptografía y sistemas distribuidos. Bitcoin soluciona dos preguntas fundamentales relacionadas al “dinero electrónico”.

1. ¿Puedo confiar en que este dinero es original y no una copia?
2. ¿Puedo estar seguro de que no existe nadie más diciendo que este dinero le pertenece? (Problema del doble gasto)

Historia

- Bitcoin fue creado en 2008 por “Satoshi Nakamoto” a través de un paper llamado “Bitcoin: A peer-to-peer Electronic Cash System”
- Está basado en innovaciones previas:
  - B-money
  - HashCash
- La innovación clave fue la creación del algoritmo de prueba de trabajo (PoW) distribuido y poder “consensuar” sobre el estado de la red
- La red comenzó a utilizarse en 2009
- Satoshi Nakamoto desapareció en 2011

¿Cómo funciona Bitcoin?

- Ejemplo completo, comprando un café con Bitcoin. Extraído del libro Mastering Bitcoin
- Vamos a ver paso a paso a alto nivel una transacción completa que simula la compra de un café y el sucesivo pago en Bitcoins
- Alice, la compradora, posee un software llamado “billetera o wallet” que le provee de una dirección en la red Bitcoin y su balance es de 0.10 BTC.

Ejemplo - Comprando un café con Bitcoin

Alice decide comprar un café en el bar de Bob y pagar en Bitcoin. Bob le muestra el siguiente QR al momento de pagar:

Ejemplo - Comprando un café con Bitcoin

bitcoin:1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA?\

amount=0.015&\

label=Bob%27s%20Cafe&\

message=Purchase%20at%20Bob%27s%20Cafe

- Dirección Bitcoin: “1GdK9UzpHBzqzX2A9JFP3Di4weBwqgmoQA”
- Monto: 0.015
- Un identificador del receptor del monto
- Una descripción del pago

Comprando un café con Bitcoin - Transacción

- Alice usa su teléfono (con la aplicación de billetera de Bitcoin) para escanear el QR
- Alice aprueba la transacción y a los pocos segundos Bob recibe el pago
- Alice acaba de realizar y aprobar una **Transacción**
- Cada TX tiene un costo asociado (fee)
- Podemos pensar que las TXs mueven valor desde su entrada (de donde viene el dinero) hacia su salida (a donde va)

Comprando un café con Bitcoin - Transacción

- La billetera de Alice supo “armar” esta TX sin ningún problema
- Normalmente las billeteras poseen una “base de datos” que lleva la cuenta de los Bitcoin no gastados
  - Distintos tipos de Wallets (o clientes) llevan distintos tipos de registro y almacenan más o menos información.
- Usando distintos métodos es posible para una wallet “liviana” “hacer preguntas” a la red
  - JSON RPC API

Comprando un café con Bitcoin - Transacción

Ejemplo de llamada para obtener un balance

```
$ curl https://blockchain.info/unspent?active=1Cdid9KFAaatwczBwBttQcwXYCpvK8h7FK
{
        "unspent_outputs":[

                {
                        "tx_hash":"186f9f998a5...2836dd734d2804fe65fa35779",
                        "tx_index":104810202,
                        "tx_output_n": 0,
                        "script":"76a9147f9b1a7fb68d60c536c2fd8aeaa53a8f3cc025a888ac",
                        "value": 10000000,
                        "value_hex": "00989680",
                        "confirmations":0
                }

        ]
}
```

Comprando un café con Bitcoin - Publicando la TX

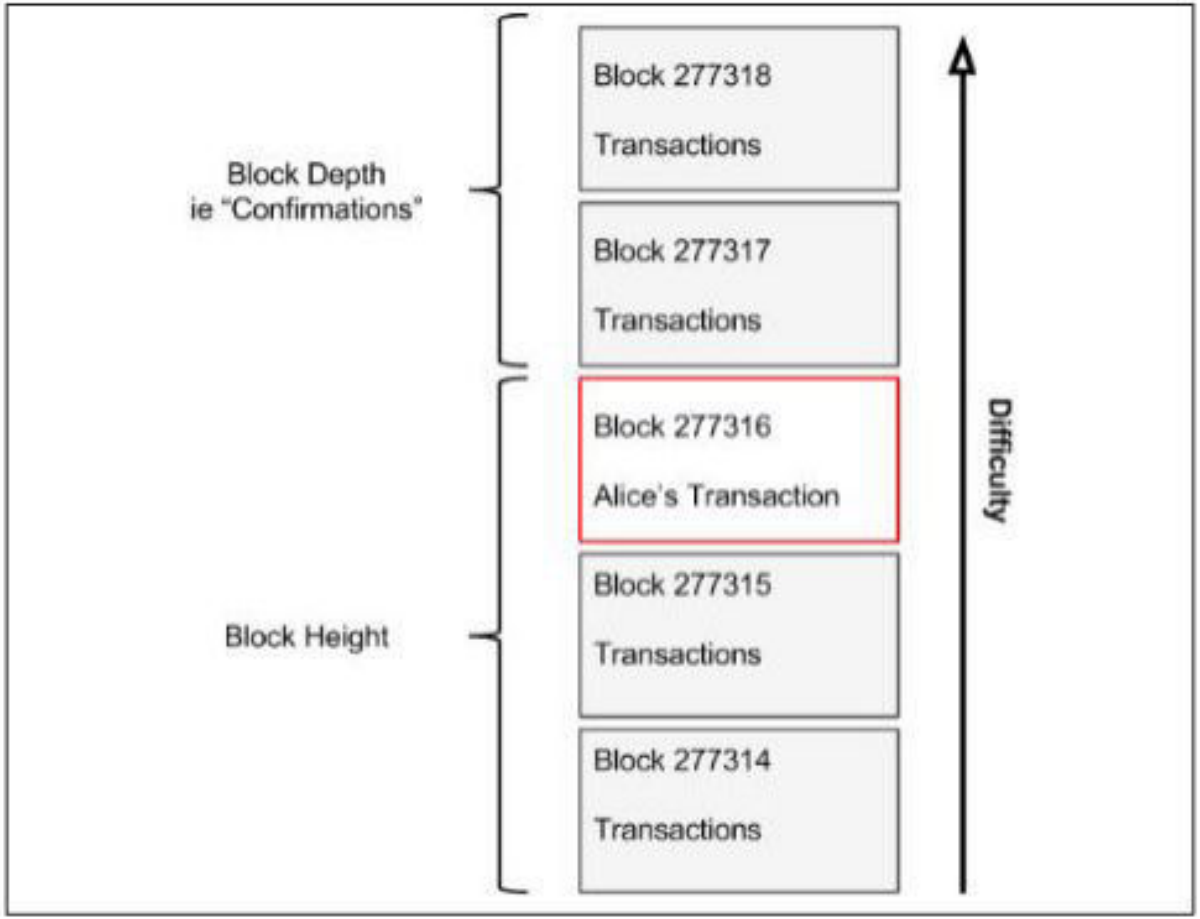
- La transacción está lista y firmada por la billetera y solo falta “publicarla”
- Como vimos anteriormente Bitcoin funciona a través de una red P2P
- La billetera de Alice puede enviar la TX a cualquier otro nodo y este la continuará reenviando a otros
- Finalmente en algún momento (generalmente segundos) la billetera de Bob “verá” la TX y podrá comprobar que es válida
- De todas formas la TX no es incluida en un bloque hasta que no es “minada”

Comprando un café con Bitcoin - Minado

- El minado puede pensarse como un “Sudoku” enorme
- Se ajusta la dificultad automáticamente
- Es complejo resolverlo, pero fácil validar una solución
- El algoritmo de PoW se basa en calcular un hash SHA256 de la cabecera del bloque hasta que el hash cumple ciertas condiciones
- Las nuevas TXs se agregan a bloques los cuales deben ser minados

Comprando un café con Bitcoin - Minado

- Volviendo a nuestro ejemplo, la TX de Alice fue tomada del pool por el minero Jing.
- La misma fue incluida en un bloque y este bloque fue minado por Jing utilizando hardware especializado (ASICs)
- Tiempo después, otros bloques son minados “sobre” el bloque que contiene la TX de Alice, confirmando esta cada vez con más seguridad


Block Depth
ie "Confirmations"
Block Height
Block 277318
Transactions
Block 277317
Transactions
Block 277316
Alice's Transaction
Block 277315
Transactions
Block 277314
Transactions
Difficulty


BITCOIN WHITEBOARD TUESDAY
99 BITCOINS
FROM "SEND" TO "RECEIVED"
HOW BITCOIN TRANSACTIONS WORK

Comprando un café con Bitcoin - Resumen

1. Nuevas TXs se propagan a todos los nodos de la red
2. Cada nodo toma las TXs y las agrega a un bloque nuevo
3. Cada nodo trabaja en encontrar el PoW para el bloque creado
4. Cuando el nodo encuentra la solución, la hace pública a todos los nodos
5. El resto de los nodos lo valida y solo lo acepta si todas las TXs incluídas son válidas
6. Los nodos aceptan el bloque usando su hash para calcular el hash del siguiente bloque

Comprando un café con Bitcoin - Explorando un bloque

Ejemplo práctico:

https://blockchair.com/es/bitcoin/block/765625

Claves, direcciones y billeteras

Claves, direcciones y billeteras

- La tenencia de bitcoins se establece a través de:
  - Claves digitales
  - Una dirección de Bitcoin
  - Firmas digitales

Claves digitales

- No se almacenan en la red
  - Almacenadas en archivos de forma local *
- Son completamente independientes del protocolo
- Se pueden generar de forma independiente
- Proveen varias propiedades interesantes
  - Prueba de identidad
  - Seguridad criptográfica
  - Prueba de “tenencia”

Firma digital

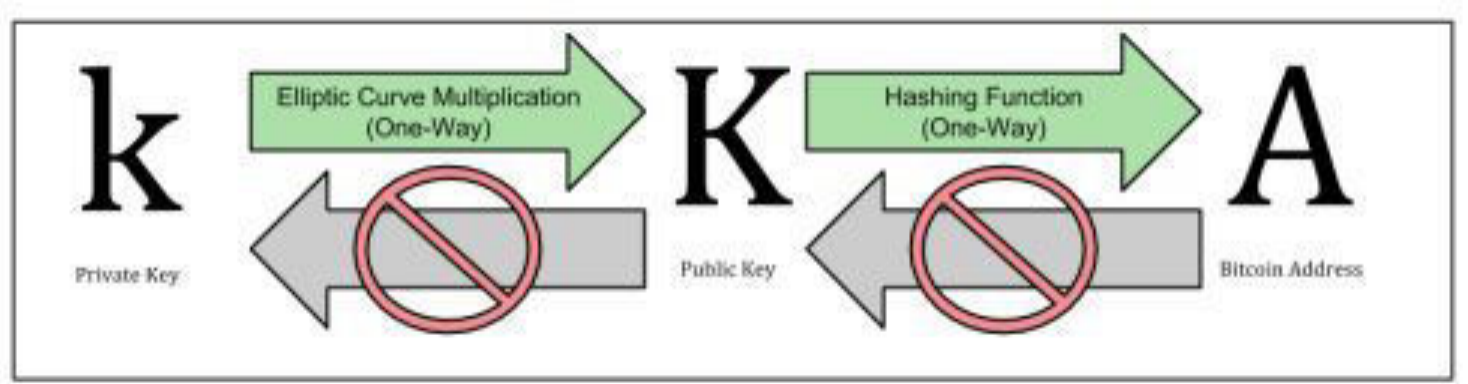
- Toda TX requiere una firma para ser válida
- Para realizar esta firma se requiere acceso a las claves digitales
  - Clave privada
- Podemos pensar en la siguiente analogía:
  - Clave Pública: Cuenta de Banco
  - Clave Privada: PIN de la cuenta
- Normalmente los usuarios finales no ven estas claves
  - Abstracción por parte de la billetera o wallet
- Normalmente la clave pública se representa via la “dirección de Bitcoin”
  - Existen excepciones: Algunas direcciones son de “scripts”

Firma digital

- La firma digital se basa en la criptografía asimétrica
- Consiste en una clave privada (generada de una forma específica) y una clave pública derivada de la misma
- La clave pública nos permite recibir Bitcoins
- La clave privada firmar TXs y por ende enviar Bitcoins
- Al momento de gastar Bitcoins:
    - El dueño presenta su clave pública y una firma (que es distinta en cada ocasión) derivada de la clave privada en una TX

Clave Privada y Clave Pública

- Cada wallet contiene un par de claves



La clave privada

- Número aleatorio elegido al azar
- Utilizada para crear firmas y validar transacciones
- Debe ser secreta y estar protegida
- Para generarla se deben tener en cuenta los siguientes puntos:
  - Tener una buena fuente de “aleatoriedad” o entropía
  - Elegir un número entre 1 y 2^{256}

La clave privada

- Normalmente el proceso para obtener este número es el siguiente:

1. Se inicializa de forma segura un generador de números aleatorios (criptográficamente seguro) *
2. A esta entrada se le aplica una función de hash (sha256) que producirá un número
3. Si el número es menor a 2^{256} se tiene una clave válida, sino, se repite el proceso

La clave pública

- Generada a partir de la clave privada y multiplicación de curva elíptica

$$K = k * G$$

K = Clave pública

k = Clave privada

G = “Punto Generador”

Dirección de Bitcoin

- Generada a partir de la clave pública
- Las direcciones generadas a partir de una clave pública consisten en números y letras que empiezan por “1”.
- Ejemplo: “1thMirt546nngXqyPEz532S8fLwbozud8”
- Para crear la dirección a partir de la clave pública se utiliza
  - SHA256
  - RIPEMD160
- Dirección = RIPEMD160(SHA256(Clave Pública))
- Se utiliza normalmente un “encoding” llamado “Base58Check”

Base58Check

- Forma práctica de representar números muy largos
- Utiliza menos símbolos
- “Subset” de Base-64
  - Se removió el 0 (cero), O (letra “o” mayúscula), l (letra “l” minúscula), I (letra “i” mayúscula) y los símbolos “/”, “+” y “\”
- Incluye comprobación de errores a través de un “checksum”

Base58Check

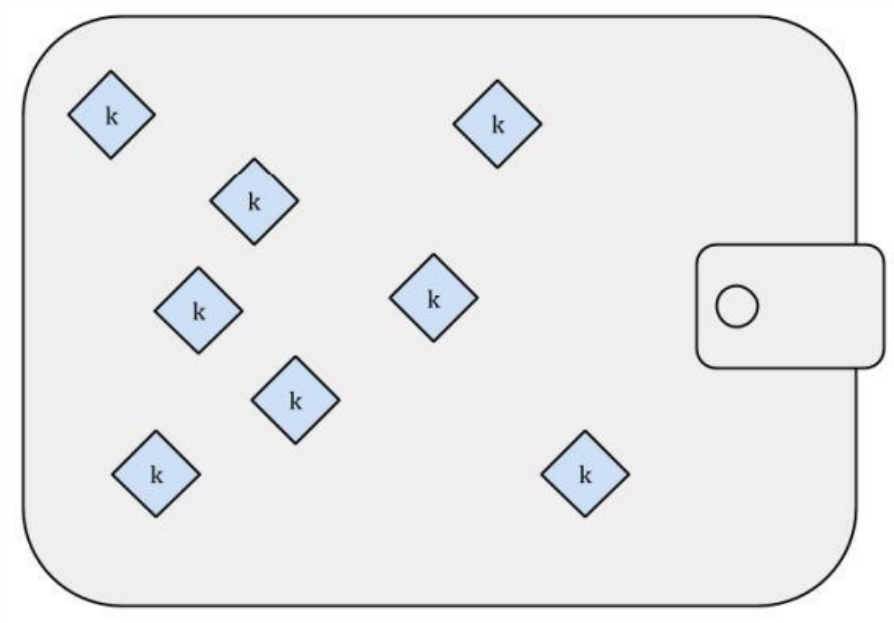
Value	Character	Value	Character	Value	Character	Value	Character
0	1	1	2	2	3	3	4
4	5	5	6	6	7	7	8
8	9	9	A	10	B	11	C
12	D	13	E	14	F	15	G
16	H	17	J	18	K	19	L
20	M	21	N	22	P	23	Q
24	R	25	S	26	T	27	U
28	V	29	W	30	X	31	Y
32	Z	33	a	34	b	35	c
36	d	37	e	38	f	39	g
40	h	41	i	42	j	43	k
44	m	45	n	46	o	47	p
48	q	49	r	50	s	51	t
52	u	53	v	54	w	55	x
56	y	57	z				

Decimal version	Leading symbol	Use
0	1	Bitcoin pubkey hash
5	3	Bitcoin script hash
21	4	Bitcoin (compact) public key (proposed)
52	M or N	Namecoin pubkey hash
128	5	Private key (uncompressed pubkey)
128	K or L	Private key (compressed pubkey)
129-135	5, K, L, or M	Private key (Electrum-defined[1] and now deprecated[2])
111	m or n	Bitcoin testnet pubkey hash
196	2	Bitcoin testnet script hash

Billeteras o “Wallets”

- Contenedores de Claves privadas (no de Bitcoins).
- Existen de distintos tipos
  - No-Determinísticas
  - Determinísticas
- No-Determinísticas
  - Implementación más antigua
  - Conjunto de claves privadas generadas de forma aleatoria
  - Difíciles de mantener
  - Requieren backups frecuentes
  - Desalientan el uso de múltiples direcciones

Billetera no-determinística



Billeteras Determinísticas

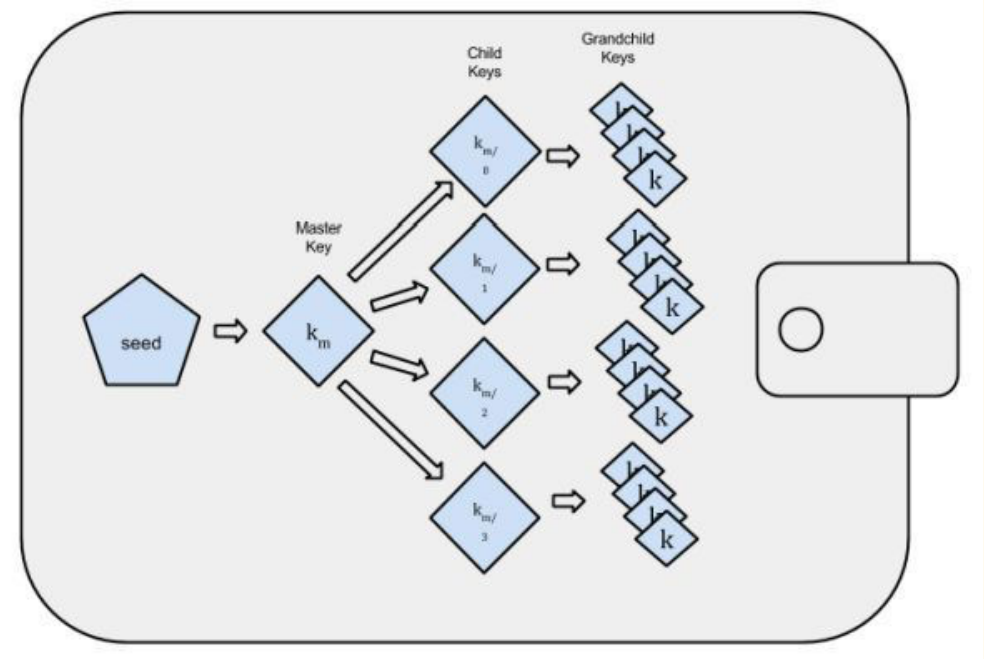
- Las claves privadas se derivan de una semilla o “seed”
- Se utiliza una función de hash en conjunto al seed + otra información
- Conociendo el “seed” se pueden derivar todas las Private Keys
  - Solo hace falta hacer copia de seguridad del mismo
- Son las más recomendadas actualmente

Mnemónicos

- Listado de palabras utilizadas para representar el “seed”
- Se utilizan de 12 a 24 palabras
- Conociendo estas palabras es posible generar las claves privadas
  - ¡SE DEBEN PROTEGER SIEMPRE!
- BIP0039 define la forma de generar estas palabras

Billeteras Determinísticas con jerarquía

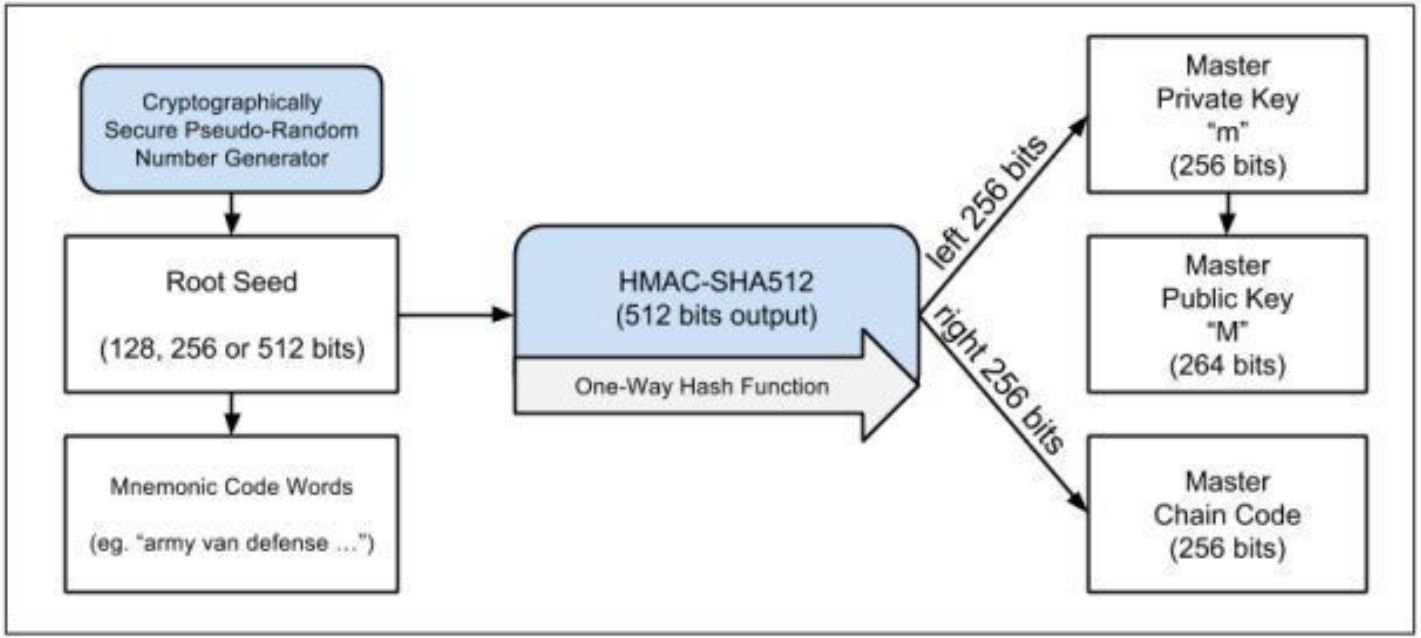
- Definidas en el standard BIP0032/BIP0044



Billeteras Determinísticas con jerarquía

- Son las más avanzadas
- Permiten crear estructuras similares a las que mantienen organizaciones
  - Claves por sector, o sucursal
- Es posible crear claves públicas sin poseer la clave privada
  - Útiles para recibir pagos en entornos inseguros
- Se crean a partir de una semilla

Billeteras Determinísticas con jerarquía

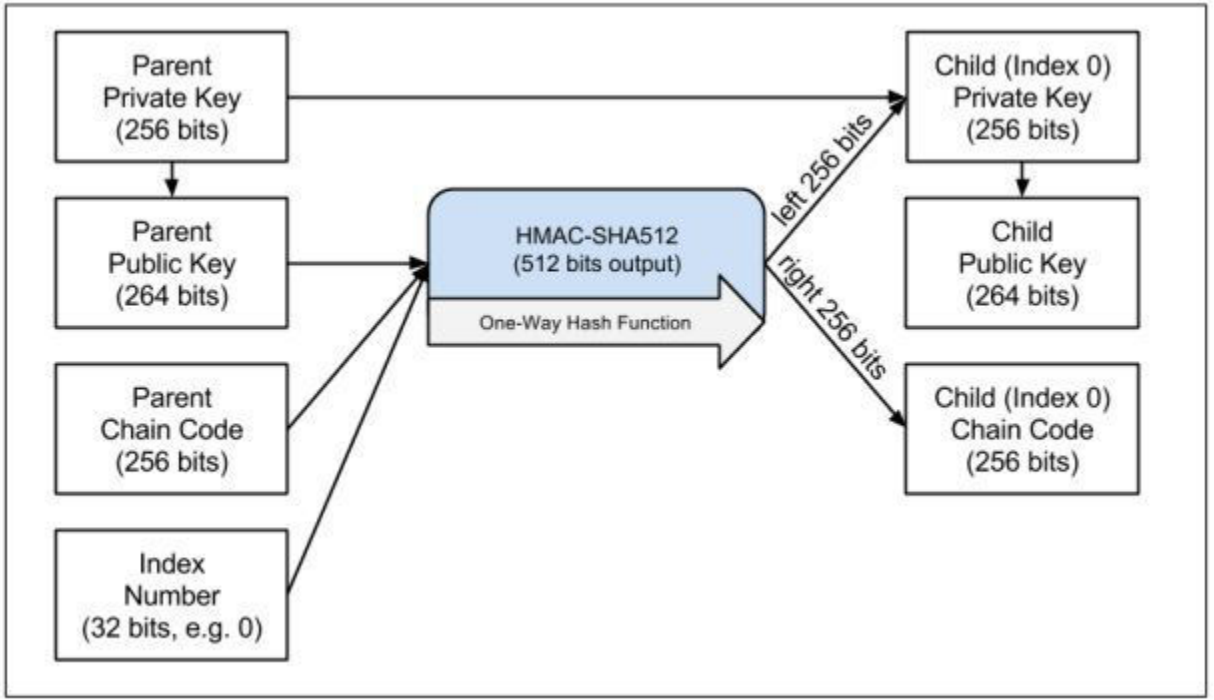


Billeteras Determinísticas con jerarquía

Derivación de claves “hijas”

- Se utiliza una función “child key derivation” (CKD), la cual utiliza hashes y combina tres elementos:
  - Una clave (pública o privada)
  - Una semilla (Chain Code)
  - Un índice (Index)
- El proceso funciona de la siguiente manera

Billeteras Determinísticas con jerarquía



Billeteras Determinísticas con jerarquía

- Las claves privadas “hijas” son iguales a las claves privadas generadas de forma aleatoria
- Solamente es posible derivar claves teniendo la clave Padre y el “chain code”
- Al conjunto de “Chain code” y claves privadas “hijas” se lo denomina “Clave extendida” (Extended key)

Cifrado de claves privadas

Cifrado de claves privadas

- Las claves privadas deben ser SIEMPRE secretas
  - Esto es complejo de mantener en la práctica
- Algunas dificultades
  - Backups
  - Migración de billeteras
  - Facilidad de uso
- Esto dio lugar al standard BIP0038
  - Propone un formato común para cifrar claves privadas
- Utiliza una contraseña (fuerte)
- El algoritmo de cifrado utilizado es AES

Pay To Script Hash (P2SH)

Pay To Script Hash (P2SH)

- Hasta ahora sabemos que las direcciones “comunes” de Bitcoin empiezan con un “1”
- Si enviamos Bitcoins a este tipo de direcciones la única forma de gastarlos es teniendo la clave privada asociada
- Existe otra alternativa, direcciones que comienzan por “3”
- Estas direcciones son conocidas como “Pay-To-Script Hash”
  - Requieren algo más que la clave privada para poder gastar Bitcoins
  - Los requerimientos son definidos al momento que la dirección es creada
- Se crean a partir de un “script” de transacción
- El ejemplo más común de script es el de billetera “multifirma”

Vanity Addresses

Vanity Addresses

- Direcciones que contienen texto con “sentido”
  - 1LoveBPzzD72PUXLzCkYAtGFYmK5vYNR33
- No tienen otro uso particular
- Difíciles de generar
- No existe un mecanismo más allá de la fuerza bruta

Transacciones

Transacciones

- Parte fundamental del sistema. Todo gira entorno a que las TXs puedan ser creadas, propagadas y validadas correctamente
- Son estructuras de datos que transfieren valor entre partes en la red
- Cada TX es una entrada en el “libro contable” de Bitcoin

Transacciones - Ciclo de vida

1. Creación (origination)
2. Firmado por una o más firmas
3. Broadcast a la red
4. Validación y propagación por parte de todos los nodos
5. Minado e inclusión en la cadena de bloques

Transacciones - Creación

- Analogía con un cheque
- TX expresa la intención de transferir dinero
- Puede crearse “online” o “offline”

Transacciones - Propagación

- Una vez creada y firmada la TX esta debe llegar a la red
- El emisor no necesita confiar en los nodos *
- Los nodos no necesitan establecer ninguna clase de identificación
- La transmisión de TXs puede hacerse sobre canales inseguros (no-cifrados)
- Cuando un nodo recibe la TX procede a validarla. Si es válida, la propaga
- La red está diseñada para propagar TXs de forma segura

Transacciones - Estructura

Tamaño	Campo	Descripción
4 bytes	Versión	Especifica la versión de la TX
1-9 bytes	Contador Entradas	Cantidad de entradas
Variable	Entradas	Una o más entradas
1-9 bytes	Contador Salidas	Cantidad de salidas
Variable	Salidas	Una o más salidas
4 bytes	Tiempo de Bloqueo	Timestamp

Transacciones - Salidas y Entradas

- Las TX se basan en “Salidas de TX No Gastadas” o UTXO
- UTXO son cantidades de BTC específicas de un dueño
- La red mantiene el estado de todas las UTXO
- Las UTXO componen el “saldo” de una dirección
- Las billeteras calculan el balance sumando todas las UTXO para una determinada dirección

Transacciones - UTXO

- Las UTXO pueden ser de valores arbitrarios
- Una vez creadas NO pueden ser divididas
- Las billeteras deben buscar entre las UTXO del usuario las que mejor se ajusten al monto a pagar
- Las UTXO consumidas por una TX son llamadas “Inputs”
- Las UTXO creadas por una TX son llamadas “Outputs”
- Existe una excepción llamada “Coinbase TX”, corresponde al pago del premio al nodo que “minó” el bloque

Transacciones - UTXO

- Las UTXO consisten de dos partes
  - Cantidad de Bitcoins
  - Script de bloqueo (Locking Script)
- Más adelante veremos el lenguaje de script usado

Transacciones - UTXO

Ejemplo práctico, revisando las UTXO de una cuenta:

https://blockchain.info/unspent?active=1Dorian4RoXcnBv9hnQ4Y2C1an6NJ4UrjX

Transacciones - UTXO

Script de bloqueo (Encumbrances)

- Definen las condiciones a cumplir para gastar la UTXO
- Normalmente solamente permiten gastar la UTXO a una determina dirección
- En nuestro ejemplo del café Alice le pagó a Bob, se generó una UTXO con un script que solo permite gastar esos Satoshis a Bob

Transacciones - Inputs

- Son “punteros” a UTXOs
- Incluyen las condiciones necesarias (scripts) para desbloquear las UTXOs

Tamaño	Campo	Descripción
32 Bytes	TX Hash	Puntero a la TX que contiene la UTXO a gastar
4 bytes	Index de Salidas	Índice de la UTXO a ser gastada
1-9 bytes	Tamaño script	Tamaño del script de desbloqueo en Bytes
Variable	Script de desbloqueo	El script para desbloquear la/s UTXO
4 bytes	Número de secuencia	Deshabilitado

Transacciones - Tarifas (Fees)

- Costo que se paga a los mineros por “asegurar” el trabajo
- Casi todas las TXs generan un “Fee”
- Los Fees se utilizan para:
  - Incentivar el trabajo de los mineros
  - Desalentar las TX “Spam”
- Los Fees de una TX van al minero que “minó” la TX
- Los Fees influyen en la prioridad de una TX

Transacciones - Tarifas (Fees)

- Cómo vimos antes, la estructura de una TX no tiene campo “Fee”
- Los fees se deducen como la diferencia entre las entradas y las salidas
- Ejemplo:

Si tenemos que pagar 1 BTC y tenemos una UTXO con 20 BTC, al momento de pagar debemos crear una UTXO de 19 BTC a modo de cambio, caso contrario los 19 BTC serán considerados “Fee”.

Transacciones - Lenguaje de Script

- Los clientes de la red validan TXs ejecutando un script
- El más común es el “Pay-to-Public-Key-Hash” script
  - Para TXs del estilo Alice paga X a Bob
- Este es el ejemplo más simple (pero no el único)
- El lenguaje de Script permite construcciones mucho más complejas

Transacciones - Lenguaje de Script

- El motor de validación de transacciones utiliza dos tipos de script
  - Locking Scripts
  - Unlocking Scripts
- Para cada “Input” de una TX el cliente de bitcoin obtendrá todas las UTXO y sus “Locking Scripts”. A partir de esto se tomarán todos los “Unlocking Scripts” y se ejecutarán ambos
- Primero se ejecutará el “Unlocking Script”. Si este se ejecuta sin errores se procede a ejecutar el “Locking Script” utilizando como input la salida del “Unlocking Script”

Transacciones - Lenguaje de Script

- El lenguaje de script se llama “Script”
- Basado en la estructura de pila o “Stack”
- Notación polaca inversa (2 3 + -> 5)
- Turing Incompleto
- No conserva estado

La Red de Bitcoin

La Red de Bitcoin

- Red P2P - Todos los nodos son iguales
- Topología “chata”
- La decisión de la arquitectura refleja algunos principios fundamentales
  - Decentralización
  - Robustez y seguridad
- El término de “Red de Bitcoin” hace referencia a todos los nodos ejecutando el protocolo P2P de Bitcoin
- Además del Protocolo P2P, existen otros protocolos secundarios:
  - Stratum

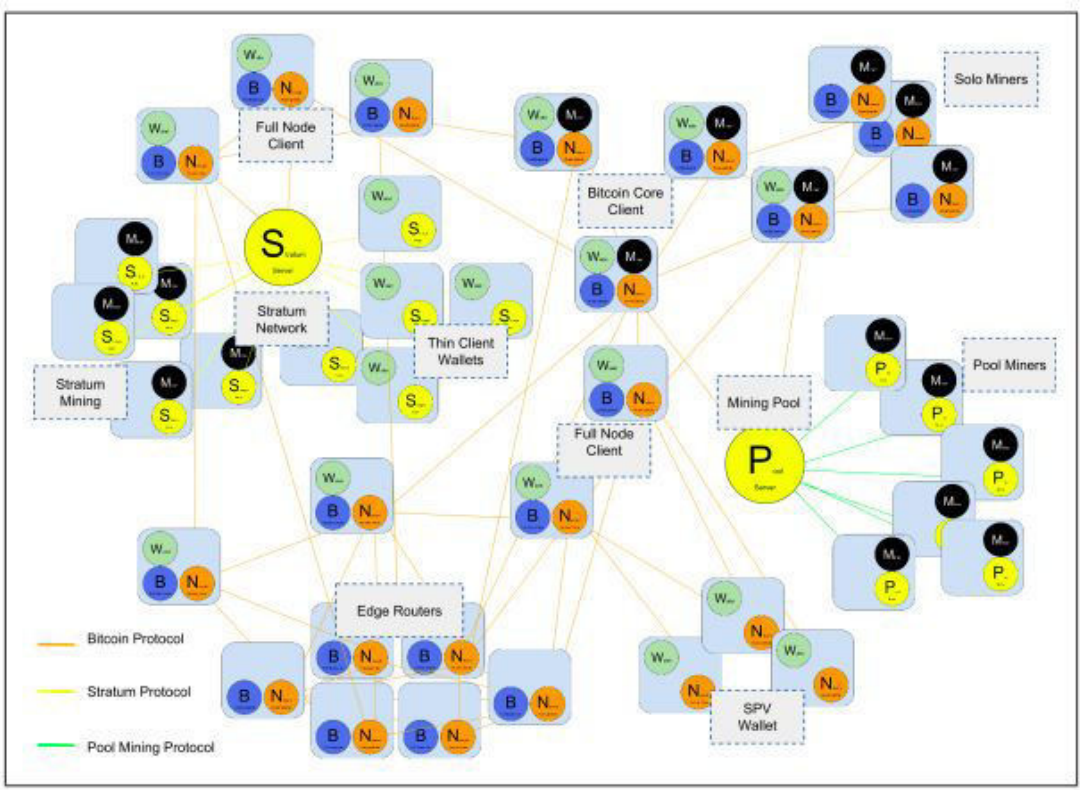
La Red de Bitcoin - Tipos de nodos y roles

- Todos los nodos son “iguales” pero pueden tener distintos “roles”
- Todos los nodos incluyen funcionalidad de “ruteo”, validan TXs y las propagan
- Existen nodos llamados “Full Nodes” los cuales además mantienen una copia completa de la “Blockchain”
  - Estos nodos pueden validar TXs sin ningún otro requerimiento
- Otros sólo mantienen una parte y validan TXs usando SPV
  - Simple Payment Verification
- Los nodos “mineros” utilizan hardware especial para resolver los problemas de PoW

La Red de Bitcoin - Tipos de nodos y roles

- **Cliente de Referencia**: Wallet, Miner, Router y backup completo
- **Nodo Full**: Backup completo, Router
- **Solo Miner**: Funciones de Minería, Routing y una copia completa de la Blockchain
- **Cliente liviano (SPV):** Wallet y Routing, no posee copia de la Blockchain
- **Pool Protocol Servers:** Sirven de nexo entre la red de Bitcoin y otros protocolos
- **Mining nodes:** Pueden realizar tareas de minería utilizando otros protocolos

La Red de Bitcoin



La Red de Bitcoin - Descubriendo Nodos

- El descubrimiento es el primer proceso que debe realizar un nodo cuando se inicializa
- Utiliza el puerto TCP 8333
- Realiza un “handshake” que contiene:
  - Versión del protocolo
  - Servicios soportados por el Nodo
  - Timestmap
  - IP del nodo remoto
  - IP local

La Red de Bitcoin - Descubriendo Nodos

¿Pero... ¿Cómo un nodo “encuentra” a otro para iniciar este proceso?

La Red de Bitcoin - Descubriendo Nodos

- Si bien todos los nodos son iguales, existen algunos con “reputación”
- Estos nodos se llaman “Nodos Semilla”
- Permiten un rápido descubrimiento del resto de nodos de la red
- Su uso es optativo

La Red de Bitcoin - Descubriendo Nodos

- Cuantos más pares (otros nodos) conozca un nodo mejor
- Un nodo recordará sus últimas conexiones y tratará de repetirlas
- Los nodos periódicamente envían información a modo de “keep-alive”
- Si un nodo no envía nada por más de 90 minutos se considera “desconectado”

Nodos completos / Full nodes

- Son nodos que mantienen una copia actualizada de toda la información contenida en la Blockchain
- Independientemente verifican TODAS las TXs desde el inicio de los tiempos
- Son los nodos “más puros”
- Tardan días en sincronizarse por primera vez y requieren mucho espacio
- El 90% de estos nodos utilizan el cliente “Standard” llamado “Satoshi”

Nodos Livianos / SPV nodes

- Son nodos que están destinados a correr en hardware limitado
- No mantienen una copia de la Blockchain
- Empiezan a ser los más comunes
- Solamente descargan la cabecera de cada bloque
  - El resultado es una “Blockchain 1000 veces más liviana”
- Deben confiar en otros nodos para poder verificar TXs
  - “Les consultan” por información específica

La Red de Bitcoin - Transaction Pools

- Casi todos los clientes mantienen una lista de TXs que no han sido “minadas”
  - Son TXs válidas
- Esta lista de TXs se almacena en memoría y no se persiste
- Los nodos comparten esta lista con sus “nodos vecinos”

La Blockchain

Estructura de la Blockchain

- Lista enlazada de bloques que contienen las TXs
- Puede ser almacenada como archivo o en una BD
  - Google LevelDB
- Cada bloque hace referencia al anterior mediante su “hash”
- Cada bloque puede tener un solo “padre” pero (momentáneamente) múltiples hijos

Estructura de un bloque

Size	Field	Description
4 bytes	Block Size	The size of the block, in bytes, following this field
80 bytes	Block Header	Several fields form the block header (see below)
1-9 bytes (VarInt)	Transaction Counter	How many transactions follow
Variable	Transactions	The transactions recorded in this block

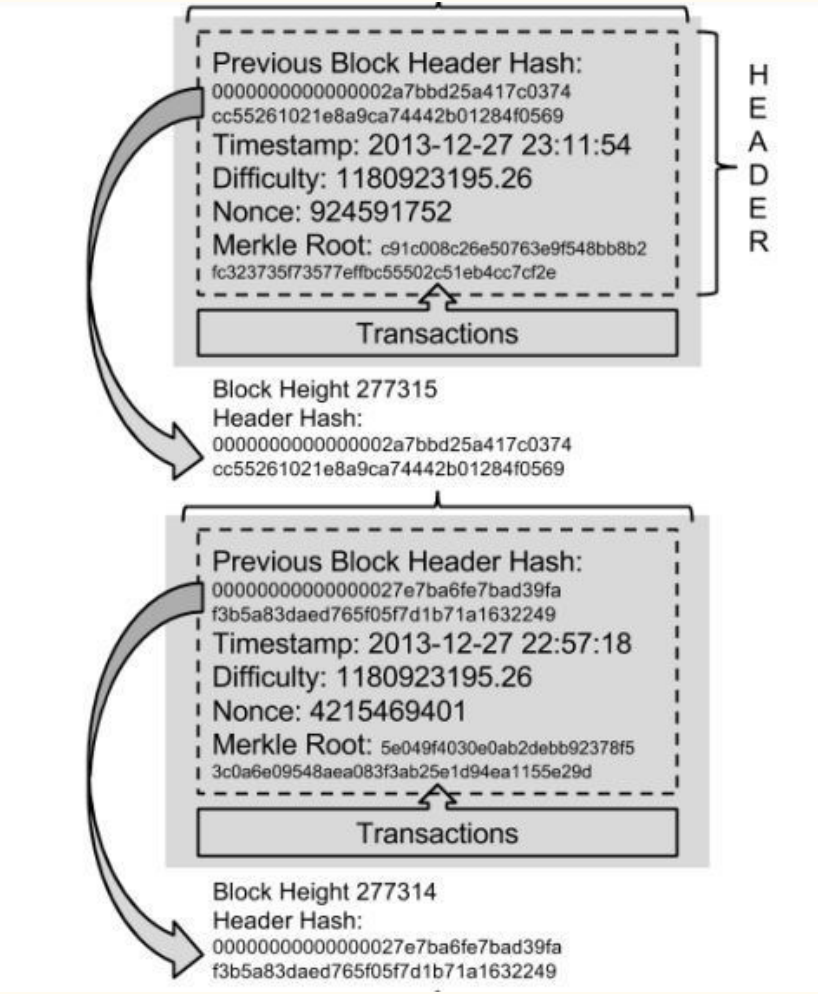
Encabezado de bloque / Header

- Consiste en tres sets de “metadata”
  - Referencia al bloque anterior
  - Dificultad, timestamp y Nonce
  - Raiz del arbol de Merkle

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the Merkle-Tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Difficulty Target	The proof-of-work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the proof-of-work algorithm

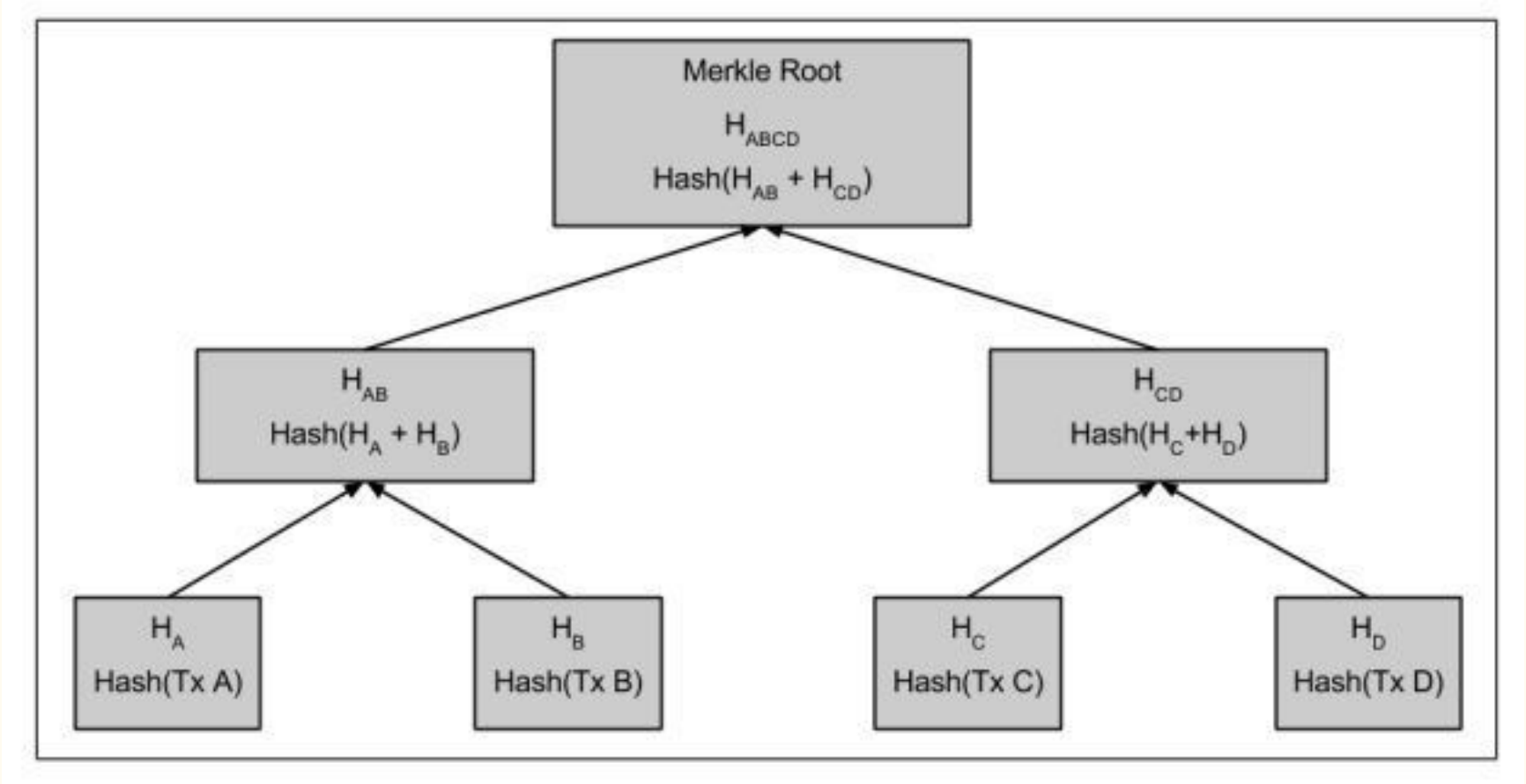
Identificador de bloque

- Para identificar el bloque podemos utilizar su hash
- Como alternativa tenemos la “altura de bloque”, relativa a la posición del bloque en la cadena
- El hash identifica inequívocamente a un bloque, su altura no necesariamente


Previous Block Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Timestamp: 2013-12-27 23:11:54
Difficulty: 1180923195.26
Nonce: 924591752
Merkle Root: c91c008c26e50763e9f548bb8b2
fc323735f73577effbc55502c51eb4cc7cf2e
HEADER
Transactions
Block Height 277315
Header Hash:
0000000000000002a7bbd25a417c0374
cc55261021e8a9ca74442b01284f0569
Previous Block Header Hash:
00000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249
Timestamp: 2013-12-27 22:57:18
Difficulty: 1180923195.26
Nonce: 4215469401
Merkle Root: 5e049f4030e0ab2debb92378f5
3c0a6e09548aea083f3ab25e1d94ea1155e29d
Transactions
Block Height 277314
Header Hash:
00000000000000027e7ba6fe7bad39fa
f3b5a83daed765f05f7d1b71a1632249

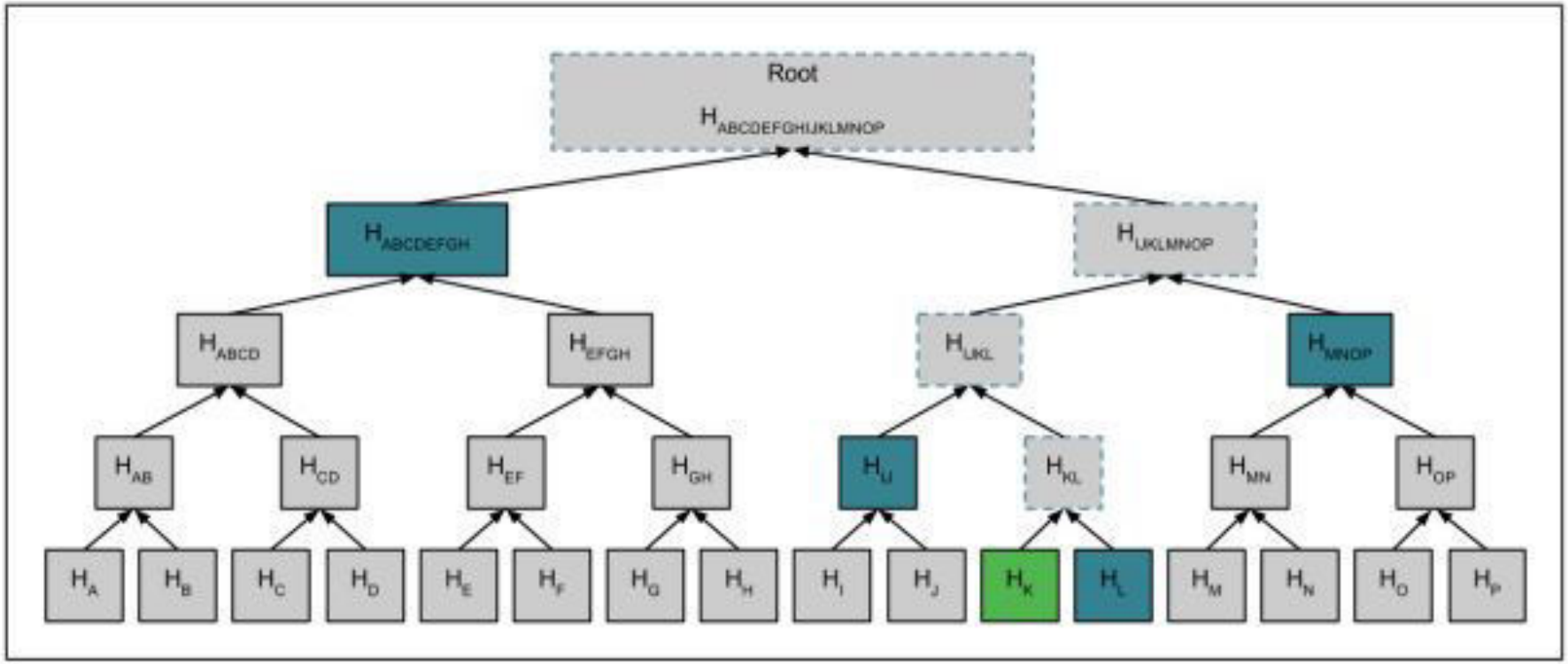
Merkle Trees / Árboles de Merkle

- Se utilizan para sumarizar las TXs en un bloque
- Generalmente son utilizados para resumir y verificar grandes cantidades de información
- Son árboles binarios que contienen hashes
- Para el hash se utiliza SHA256 dos veces
- Recursivamente se “hashean” pares de nodos hasta que solo tenemos uno, la raíz o “root”
- Permiten verificar datos de forma muy eficiente
- Se debe balancear


Merkle Root
H_{ABCD}
$Hash(H_{AB} + H_{CD})$
H_{AB}
$Hash(H_A + H_B)$
H_{CD}
$Hash(H_C + H_D)$
H_A
Hash(Tx A)
H_B
Hash(Tx B)
H_C
Hash(Tx C)
H_D
Hash(Tx D)

Merkle Trees - Verificando inclusión de TXs

- Para probar si una TX está incluida en un bloque un nodo debe crear un “camino de autenticación” o “Merkle Path”
- Este camino conectará la TX a verificar con la raíz


Root
$H_{ABCDEFGHIJKLMNOP}$
$H_{ABCDEFGH}$
$H_{IJKLMNOP}$
H_{ABCD}
H_{EFGH}
H_{IJKL}
H_{MNOP}
H_{AB}
H_{CD}
H_{EF}
H_{GH}
H_{IJ}
H_{KL}
H_{MN}
H_{OP}
H_A
H_B
H_C
H_D
H_E
H_F
H_G
H_H
H_I
H_J
H_K
H_L
H_M
H_N
H_O
H_P

Minado y Consenso

Minado

- Proceso por el cual se “crean” Bitcoins
- Además, es el mecanismo que asegura la red
- Los nodos mineros proveen poder de cómputo a cambio de la posibilidad de ser recompensados

Minado

- Los mineros crean bloques nuevos cada 10’
- Las TXs que son agregadas a la Blockchain se consideran “confirmadas”
- Los mineros reciben dos tipos de recompensa
  - Recompensa por creación de bloques
  - Pago de “aranceles” por TX
- El minado es similar a la “Emisión monetaria”
  - Cada 4 años disminuye la cantidad de recompensas
  - 50 BTC en 2009, 25 BTC, 2012, 12.5 BTC en 2016, 6.25 en la actualidad

Minado

Consenso

- Mecanismo por el cual todos los nodos se “ponen de acuerdo” sobre el estado de la información
- El consenso se logra de forma espontánea
  - No hay una elección activa del mismo
  - Todos los nodos siguen ciertas reglas
- Se logra a través de cuatro procesos independientes

Consenso

Verificación independiente de cada TX, por parte de los “Full Nodes” siguiendo una serie de reglas preestablecidas

- La sintaxis y estructura de la TX debe ser correcta
- Las entradas y salidas no deben estar vacías
- Los valores utilizados son adecuados
- La TX no supera un tamaño establecido
- ...

Consenso

Agregado de las TXs a nuevos bloques, probando la resolución de problemas complejos

- Los nodos mineros toman las TXs y compiten por crear nuevos bloques
- La llegada de un nuevo bloque indica la resolución de un problema
- Los mineros toman las TXs del “memory pool” y tratar de crear nuevos bloques

Consenso

Construcción del bloque

Size	Field	Description
4 bytes	Version	A version number to track software/protocol upgrades
32 bytes	Previous Block Hash	A reference to the hash of the previous (parent) block in the chain
32 bytes	Merkle Root	A hash of the root of the Merkle-Tree of this block's transactions
4 bytes	Timestamp	The approximate creation time of this block (seconds from Unix Epoch)
4 bytes	Difficulty Target	The Proof-of-Work algorithm difficulty target for this block
4 bytes	Nonce	A counter used for the Proof-of-Work algorithm

Construcción del bloque

- El minero incluye una TX especial que es la que paga su recompensa
- Debe calcular la raíz del árbol de Merkle en base a las TXs agregadas
- Se completan el resto de campos y se procede a resolver el algoritmo de PoW variando el Nonce

Consenso

Verificación independiente de cada nuevo bloque

- Cada nodo de la red independientemente verifica la validez de cada nuevo bloque publicado
- Esto valida que solo los nodos “honestos” ganen sus recompensas
- Además, evita las trampas

Consenso

Selección independiente por parte de cada nodo de la cadena con la mayor cantidad de trabajo realizado

- Los nodos mantienen tres sets de bloques:
  - Los de la “cadena principal”
  - Los que generan “cadenas secundarias”
  - Y bloques “huérfanos”
- Se considera como cadena principal a aquella que tiene más bloques unidos y por ende mayor trabajo realizado
- A veces se minan bloques casi al mismo tiempo y esto da lugares a cadenas secundarias

Forks

¿Preguntas?

Introducción a blockchain y billeteras virtuales - Módulo 3

Docente: Nahuel Sánchez

*Este documento fue realizado en concepto de capacitación en Formación Profesional y dictada para el **Sindicato CePETel** a contar del mes de mayo del año 2023.*

Introducción a “Blockchain”

—

Módulo 3: Ethereum

Introducción al Módulo 3

- ¿Qué es Ethereum?
- Conceptos básicos
- Clientes
- Wallets y Transacciones
- Smart Contracts, Solidity
- Tokens
- Oráculos
- Aplicaciones descentralizadas (DAPPs)

¿Qué es Ethereum?

- Podemos definir a Ethereum como la computadora distribuida del mundo
  - Infraestructura Open-Source distribuida
  - Ejecuta programas llamados “Smart Contracts”
- Permite a desarrolladores crear aplicaciones distribuidas con funciones económicas embebidas
- Ethereum comparte algunas similaridades con Bitcoin
  - Red P2P
  - Tolerancia a fallos
  - Algoritmos de consenso

¿Qué es Ethereum? - Ethereum vs Bitcoin

- El propósito fundamental es distinto
  - Bitcoin: Ser una alternativa de pago
  - Ethereum: ser una alternativa de propósito general que se ejecuta de forma distribuida
- Si bien existe el “Ether” como moneda, su uso se concibió para pagar por la ejecución de programas
- Como vimos anteriormente el lenguaje de script de Bitcoin fue limitado a propósito
- Ethereum es Turing completo

¿Qué es Ethereum?

- Los lineamientos básicos de Ethereum fueron presentados por Vitalik Buterin en su “Ethereum Yellow Paper”
- Existe una versión más amena de ese paper llamado “Beige Paper”
- Ethereum cuenta con los componentes típicos que ya aprendimos anteriormente:
  - Red P2P
  - Transacciones
  - Reglas de consenso
  - Una cadena de bloques
  - Algoritmo de PoW / PoS
  - Clientes

¿Qué es Ethereum? - Componentes

- Red P2P: Red sobre la cual funciona Ethereum, utiliza el puerto TCP 30303 y un protocolo llamado DEVp2p
- Reglas de consenso: Permiten llegar a un acuerdo común sobre el estado de la blockchain
- Transacciones: Mensajes que se envían a la red
- Máquina de estados: Las transiciones son procesadas por la EVM, una máquina virtual basada en el tipo de dato “Pila” que ejecuta programas (Smart contracts)
- Estructuras de datos: Mantienen almacenada la información en cada nodo

¿Qué es Ethereum?

- Ethereum aparece como solución a un problema
  - Se basa en el potencial de Bitcoin pero resuelve su falta de versatilidad
- Construir sobre la tecnología de Bitcoin no es fácil y escalable
  - Limitaciones técnicas
- En 2013 Vitalik comparte un paper que contiene las bases de Ethereum
  - Lenguaje Turing completo
  - De propósito general
- Ethereum se diseñó y creó con el propósito de servir de plataforma para aplicaciones de propósito general distribuidas

¿Qué es Ethereum?

- Etapas de desarrollo de Ethereum
  - Frontier
  - Homestead
  - Metropolis
  - Serenity
- “Hard-Forks” / Bifurcaciones
  - Ice Age
  - DAO
  - Tangerine Whistle
  - Spurious Dragon
  - Byzantium
  - Constantinople
  - Istanbul
  - Muir Glacier

Frontier

- Bloque #0
- La etapa inicial de Ethereum
- Duró desde el 30 de Julio de 2015 a Marzo del 2016

Ice Age

- Bloque #200.000
- El primer “hard-fork” que introdujo dificultad exponencial en el minado para pasar a un algoritmo de PoS cuando fuera necesario

Homestead

- Bloque #1.150.000
- La segunda parte del desarrollo de Ethereum, lanzado en Marzo del 2016

DAO

- Bloque #1.192.000
- Un momento clave en la historia de Ethereum (¿El código es ley?)
- Se produce la división entre Ethereum Classic y Ethereum
- Se decide devolver los fondos a las víctimas de un hackeo

Tangerine Whistle

- Bloque #2.463.000
- Otro hard-fork, se deciden cambiar ciertos parámetros de cálculo de comisiones (gas) y limpiar el “estado”

Spurious Dragon

- Bloque #2.675.000
- Hard-fork, introdujo más protecciones contra ataques de DoS
- Introdujo protección contra ataques de “replay”

Metropolis Byzantium

- Bloque #4.370.000
- El tercer paso en el desarrollo de Ethereum, comenzó en Octubre de 2017
- Agregó funcionalidad de bajo nivel y se incrementó la dificultad de minado

Constantinople / St. Petersburg

- Bloque #7.280.000
- Planeada como continuación del upgrade anterior. Horas antes de su aplicación se detectaron bugs críticos que tuvieron que ser arreglados

Istanbul

- Bloque #9.069.000
- Hard-fork con las mismas ideas que los dos anteriores (actualizaciones y aumento de dificultad)

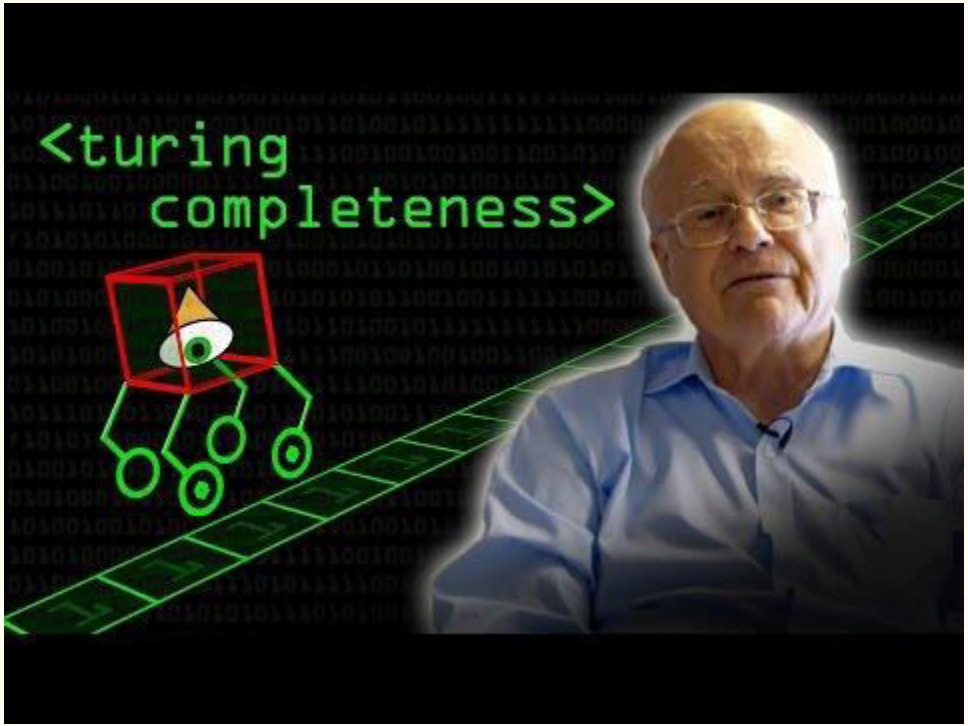
Muir Glacier

- Bloque #9.200.000
- Hard-fork con las mismas ideas que los dos anteriores (actualizaciones y aumento de dificultad)

Ethereum es Turing Completo

- Este término hace referencia a Alan Turing, un matemático Británico considerado el padre de la ciencia de la computación
- En 1936 creó un modelo de una computadora que consiste en una máquina de estados y procesa símbolos leyéndolos desde una “cinta”
- A partir de esto pudo probar que existen clases de problemas que no son computables. Específicamente probó que el “halting problem” no era computable

Ethereum es Turing Completo



Ethereum es Turing Completo

- Esto es un arma de doble filo
  - Provee versatilidad
  - Puede traer grandes problemas
- Puede derivar en consecuencias no pensadas
- Pensar cuáles podrían ser las consecuencias de que un programa “se cuelgue” en la Blockchain de Ethereum
- Turing probó que no es posible conocer el comportamiento de un programa simulandolo
- Al mismo tiempo cada nodo debe ejecutar cada TX y correr cada Smart contract
- Para evitar esta clase de problemas o ataques se introdujo el concepto de ***Gas***

¿Qué es Ethereum? - Gas

- La EVM lleva la cuenta de la ejecución de los programas. Cada instrucción tiene un costo asociado
- Cuando una TX desea ejecutar un Smart Contract debe incluir el Gas adecuado
- Si la ejecución “sale más cara” que el Gas disponible la ejecución se termina
- De este modo se limita la ejecución de programas en la EVM
- El Gas se paga con “Ether”

¿Qué es Ethereum? - Gas

- La EVM lleva la cuenta de la ejecución de los programas. Cada instrucción tiene un costo asociado
- Cuando una TX desea ejecutar un Smart Contract debe incluir el Gas adecuado
- Si la ejecución “sale más cara” que el Gas disponible la ejecución se termina
- De este modo se limita la ejecución de programas en la EVM
- El Gas se paga con “Ether”

Aplicaciones Descentralizadas (DAPPs)

- La potencia de Ethereum permite a desarrolladores crear aplicaciones que funcionan ejecutándose en la EVM
- Una Aplicación descentralizada es la suma de uno o más contratos y una aplicación de “front-end”
- Veremos más sobre esto en futuras secciones

Web3

- En 2004 apareció el término “Web 2.0”
  - Aplicaciones “Responsivas”
  - Interactividad
- La idea de aplicaciones distribuidas (DApps) se cree que en el corto plazo revolucionará la red (otra vez)
- El término para explicar esta revolución es “Web3”, la idea de una nueva generación de aplicaciones web mezcladas con tecnología Blockchain

Conceptos básicos

Conceptos básicos

- Ether: Es la moneda de Ethereum, también llamada ETH
- Se divide en partes más pequeñas, la menor llamada “Wei”
- 1 ETH = 1 x 10^{18} wei
- Existen otras unidades también:
  - Finney = 10^{15}
  - Szabo = 10^{12}
  - Shannon = 10^{9}
  - ...

Conceptos básicos

- Ethereum también utiliza el concepto de Wallet o billetera
- La billetera es la conexión a la red de Ethereum
- También utiliza el concepto de clave pública y privada
- Existen muchas variantes, las más comunes son:
  - Metamask
  - Jaxx
  - Otras

Conceptos básicos - Seguridad de la Private key

Independientemente de la billetera a utilizar es clave siempre proteger el acceso a las claves privadas

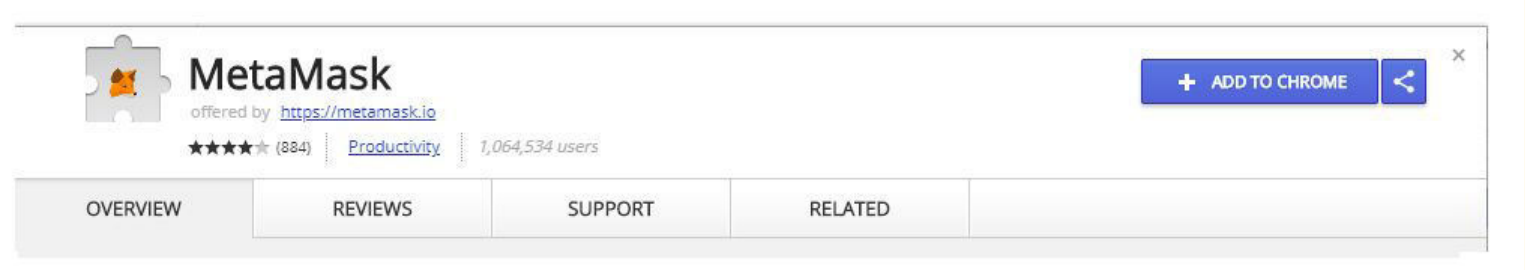
Si se pierde el acceso a la clave privada, se pierde acceso a cualquier saldo que se tuviera y lo mismo sucede con el acceso a los Smart Contracts

Conceptos básicos - Seguridad de la Private key

- No improvisar, usar métodos verificados para almacenar la PK
- La cantidad de cuidado debe ser directamente proporcional a la importancia de la cuenta
- La mayor seguridad se consigue utilizando “cold” wallets pero no es necesario para todas las cuentas
- Nunca almacenar la PK en texto plano de forma digital
- Siempre realizar copia en papel y lápiz de las palabras generadoras

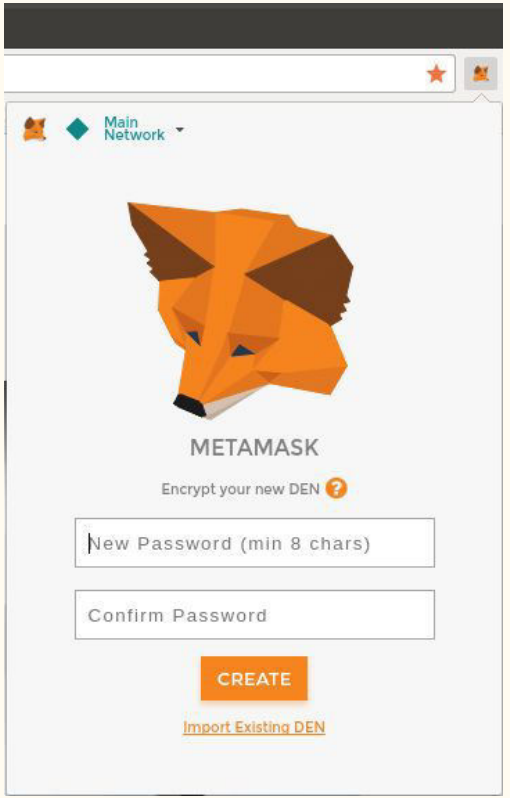
Conceptos básicos - utilizando Metamask

- Metamask es una aplicación de billetera compatible con varios navegadores y sistemas operativos
- Puede instalarse como extensión



Conceptos básicos - utilizando Metamask

- Una vez instalada podemos crearnos nuestra primer cuenta
- Utilizaremos una contraseña para evitar que cualquiera pueda acceder a la cuenta
- Metamask creará una cuenta y nos mostrará la lista de palabras que utilizó como “seed”



Conceptos básicos - utilizando Metamask

- Afortunadamente existen varias redes distintas, con distinto propósito
- Para realizar pruebas y aprender podemos utilizar las redes de “test” sin tener que gastar dinero real
- Las redes son:
  - Main Ethereum Network - La red real, el Ether equivale a dinero
  - Sepolia - Red de Test recomendada para desarrollo
  - Goerli - Red de Test recomendada para testing de validadores y staking
  - Rikeby - Red de Test (próxima a deprecarse)
  - Kovan - Red de Test (próxima a deprecarse)
- Metamask permite elegir la red a donde conectarse

Conceptos básicos - utilizando Metamask

- Para conseguir Ether de prueba existen los “faucets” o “grifos”
  - No siempre es fácil conseguir Ether de prueba aunque no tenga valor monetario
- Veremos un ejemplo de uso

- Faucet: https://sepoliafaucet.net/

Conceptos básicos - EOAs y Contratos

- La cuenta que creamos en Metamask se denomina Externally Owned Account (EOA)
- Las EOAs tienen la particularidad de tener una clave privada utilizada para firmar TXs
- También existen cuentas que no poseen una clave privada. Estas son “controladas” por los contratos inteligentes
- Los contratos que se ejecutan en la EVM tienen su propia dirección y pueden enviar y recibir Ether
- Para ejecutar un contrato debemos ejecutar una TX e incluir el Gas y los datos necesarios

Conceptos básicos - EOAs y Contratos

- Dado que las cuentas asociadas a contratos no poseen claves privadas, estos no puede iniciar transacciones (no pueden firmar la TX)
- Las EOAs puede iniciar transacciones y los contratos reaccionar a esto

Conceptos básicos - EOAs y Contratos

Ejemplo práctico. Instalando y usando Metamask

Clientes Ethereum

Clientes

- Un cliente para la red Ethereum es un programa que implementa la especificación definida en el “Yellow Paper” y se comunica con otros clientes en la red
- Existen distintas implementaciones pero todas deben respetar la implementación para poder interoperar
- Ethereum es Open Source y lo mismo sucede con la mayoría de sus implementaciones
- A diferencia de Bitcoin, Ethereum posee una especificación “formal”

Clientes

- Como sucede con Bitcoin, en Ethereum también existen distintos tipos de clientes
- Nodo completo o “Full Node”
  - Ventajas
    - Valida todas las TXs
    - No dependen de 3ras partes
    - Puede “deployar” contratos en la red
    - Puede “hacer consultas” a la red por si mismo
  - Desventajas
    - Requiere mucho espacio y procesamiento
    - Tardará días en sincronizarse
    - Debe ser mantenido regularmente

Clientes

- Nodos Livianos o “Light Nodes”
  - Ventajas
    - Más fáciles de instalar
    - Requieren menos recursos
    - Pueden “consultar” a la red
  - Desventajas
    - No mantienen una copia de la Blockchain
    - Deben confiar en otros nodos
- Nodos Archivo o “Archive Nodes”
  - Ventajas
    - Poseen todo el histórico de transacciones disponible rápidamente
  - Desventajas
    - Requieren aún más espacio que un “Full Node”

Billeteras y Transacciones

Billeteras y Transacciones

- Como vimos en el módulo anterior una de las claves a considerar cuando elegimos que billetera usar es el balance entre facilidad y privacidad
- Al igual que en la red de Bitcoin, las billeteras en Ethereum SOLO guardan una o más claves privadas (y no monedas o “tokens”)
- También aplican los conceptos que vimos de billeteras no-determinísticas y determinísticas

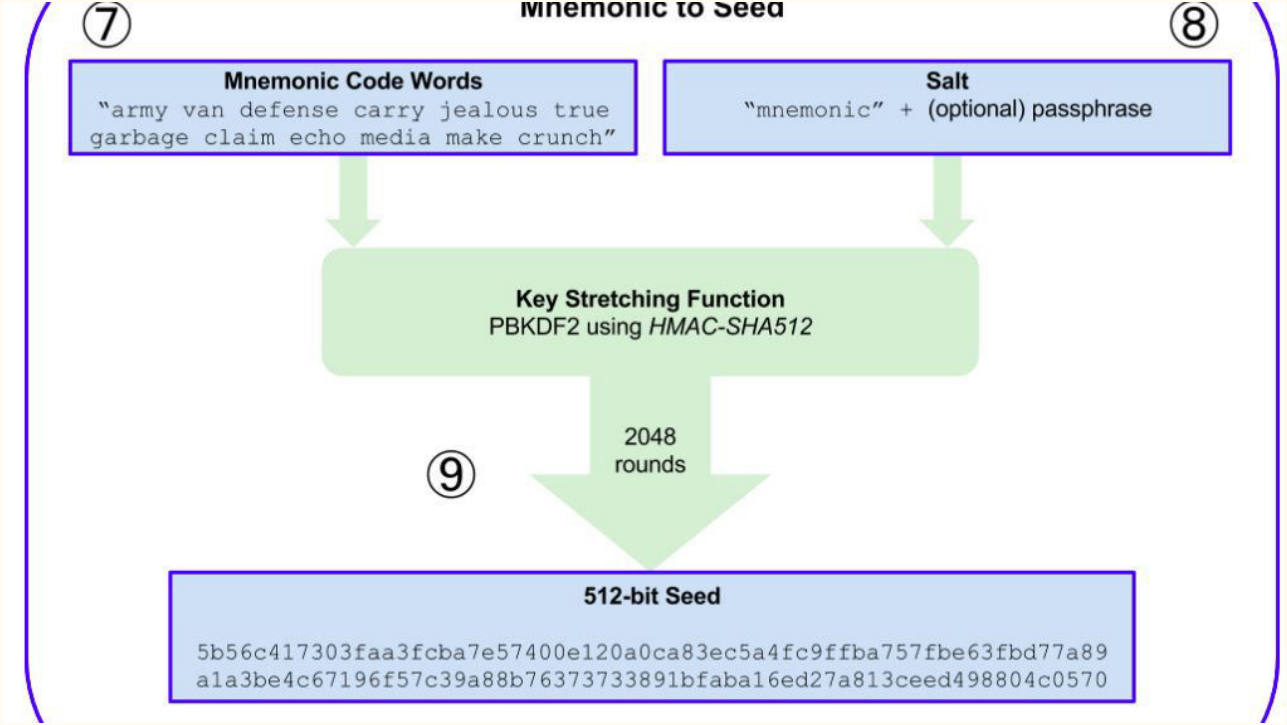
Billeteras y Transacciones

- En Ethereum también tenemos el concepto de “Seed” para inicializar una billetera determinística y el uso de “mnemónicos” (BIP-39)
- Ejemplo de semilla en hexadecimal: “FCCF1AB3329FD5DA3DA9577511F8F137”
- Ejemplo de semilla en mnemónico de 12 palabras: “wolf juice proud gown wool unfair wall cliff insect more detail hub”

De mnemónico a Semilla

1. Las 12 palabras representan entropía (bits aleatorios)
2. Esta entropía se utiliza como entrada de un algoritmo llamado PBKDF2 que deriva una longitud mayor de entropía (512 bits)
3. Además se puede utilizar una palabra secreta para sumar (aún más) entropía
4. La función PBKDF2 se ejecuta 2048 veces
5. Esto genera nuestra semilla “final” de 512 bits que será utilizada por nuestra billetera

De mnemónico a Semilla

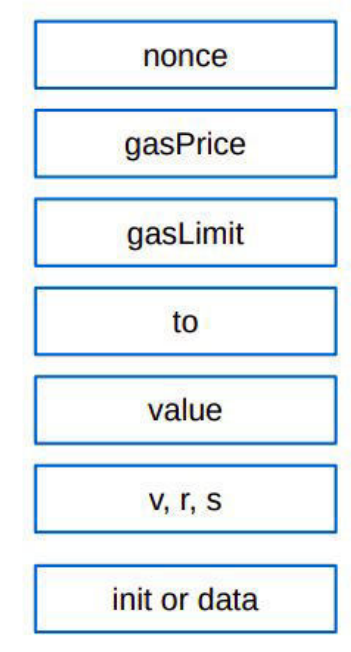


Transacciones

- Las TX son mensajes firmados originados por una EOA que son transmitidos en la red de Ethereum y guardados en la Blockchain
- Otra forma de pensar en las TX es verlas como la única operación que puede disparar un cambio de estado en la Blockchain o ejecutar un contrato en la EVM
- Ethereum es una máquina de estados global y las TXs son su “tick”

Transacciones - Estructura

- Nonce: Número de secuencia
- Gas Price: Cantidad de Ether que el emisor está dispuesto a pagar por unidad de gas (en wei)
- Gas Limit: Máxima cantidad de gas que la TX puede consumir
- to: Receptor de la TX
- Value: Cantidad de Ether enviada (en wei)
- v,r,s: Componentes de la firma digital
- Data: Información incluida en la TX



Transacciones - El NONCE

- Protección contra ataques de “repetición”
- Funciona en dos escenarios:
  - Permite “ordenar” la ejecución de TXs
  - Evita que sea posible “copiar” TXs
- Las billeteras mantienen registro del último NONCE utilizado, además puede consultarse a la red
- Cabe destacar los siguientes escenarios:
  - Salto o “gap” en los NONCES
  - NONCES duplicados
- Tener cuidado con la concurrencia

Transacciones - Gas

- El Gas es el “combustible” de la red Ethereum
- El Gas no es “Ether”, pero se paga utilizando Ether
- El Gas permite pagar el costo de los recursos que utiliza una TX
- El campo “gasPrice” permite determinar cuántos Wei (Ether) está dispuesto a pagar el que envía la TX por unidad de Gas
- A mayor pago, más rápida la ejecución de la TX será (más recompensa para el nodo que la procese)

Transacciones - Gas

- El campo “GasLimit” determina la cantidad máxima de unidades de Gas que la ejecución de una TX puede usar
- TXs de pagos (transferir Ether de una cuenta a otra) consume la suma fija de 21.000 unidades de Gas
- Para calcular cuánto será el costo hacemos: 21.000 x “GasPrice”
- En cambio si nuestra TX ejecutará un contrato la cantidad necesaria de Gas puede estimarse pero no al 100%

Transacciones - Receptor

- Especificado en la TX
- Es una cadena de 20 bytes. Puede ser una EOA o la dirección de un contrato
- No se realizan validaciones, cualquier dirección es considerada válida
- Si se envía Ether a direcciones inválidas este será “quemado”, esto significa que no podrá gastarse jamás

Transacciones - Valor y Datos

- La parte “más importante” de las TXs
- Todas las combinaciones son posibles (Sin valor y con datos, sin datos y con valor, etc)
- Una TX solo con “Valor” es un pago
- Una TX solo con “Datos” es una invocación a un contrato

Transacciones - Valor y Datos

- Cuando realizamos un pago a una EOA, Ethereum grabará un “cambio de estado”, agregando el balance a la nueva cuenta
- En cambio, si el destino es un contrato, Ethereum tratará de invocar la función especificada en el campo “Datos”. Si no existe dicha función se ejecutará la función “fallback”. Si esta función no está presente se tratará de aumentar el balance del contrato. Finalmente si esto no es posible la transacción será revertida

Transacciones - Enviando información a un contrato

- La información que enviemos a un contrato será interpretada por la EVM de la siguiente manera:
  - Selector de función (4 bytes)
  - Argumentos de la función

Transacciones - La creación de contratos

- Transacción especial
- La creación de un contrato se logra enviando una TX a la dirección 0x00 (ZERO ADDRESS)
- La TX contendrá como “datos” el código del contrato a ser creado (deploy) compilado en bytecode

Smart Contracts, Solidity

Smart Contracts - Introducción

Podemos definir un “Contrato inteligente” cómo: “Programas no modificables que se ejecutan de manera determinística en el contexto de la máquina virtual de Ethereum”

- Los smart contracts son “simples” programas de computadora
- Una vez “deployados” no pueden modificarse (como todo, son datos en la blockchain)
- La ejecución es la misma para cualquiera que lo ejecute
- Existen ciertas restricciones en el contexto en el que se ejecutan

Smart Contracts - Ciclo de vida

- Escritos en un lenguaje de alto nivel (Solidity)
- Compilados a “bytecode” antes de ser “deployados” en la Blockchain
- Son representados por una dirección
- Son “ejecutados” por TX, nunca pueden correr “por sí solos” o “en Background”
- Las TXs son atómicas
- Si bien los contratos no pueden cambiar una vez “deployados”, pueden ser destruidos (SELFDESTRUCT) (Gas negativo)

Smart Contracts - Solidity

- Lenguaje procedural con sintaxis similar a JavaScript, C++ o Java
- Es el lenguaje más común para el desarrollo de Smart Contracts
- Distintos tipos de datos, el más importante “Contract”
- Utiliza “Eventos” para comunicar información al mundo exterior
- Existen distintos entornos de desarrollo, el más común Remix IDE

Smart Contracts - Solidity

Ejemplo práctico: Revisando el código de un contrato con Remix IDE

Tokens

Tokens

- Simbolizan formas de “dinero digital”. Este es su principal uso pero no el único
  - Monedas
  - Recursos
  - Acceso
  - Votación
  - Identidad
  - Otros
- Existen “fungibles” y “no fungibles”

Tokens

- Hoy en día la gran mayoría de proyectos se lanzan con algún tipo de Token asociado
- Normalmente los utilizan cómo tokens de “utilidad” o “equidad”
- Los de “utilidad” son aquellos utilizados para ganar acceso a algún servicio, aplicación, o recurso
- Los de “equidad” representar participación en el proyecto y por ende dividendos
- Los Tokens nacieron antes que Ethereum, por ejemplo, Bitcoin es un Token
- Los Tokens son distintos al Ether
    - El balance de un Token está guardado en un Contrato

Tokens

- Se basan en el estándar ERC-20 (Tokens fungibles)
- Tienen ciertas operaciones y propiedades “obligatorias”
  - totalSupply
  - balanceOf
  - Transfer
  - ...
- Mantiene el balance en una estructura llamada “mapping” (diccionario)
- Ejemplos comunes:
  - USDT
  - USDC
  - DAI

Tokens

Ejemplo práctico, revisando el Token USDT mediante Etherscan

Oráculos (Oracles)

Oracles

¿Por qué son necesarios los oráculos?

Oracles

- La EVM no tiene forma de obtener información del “mundo exterior”
- Información del exterior solo puede llegar a la Blockchain como “datos” en una TX
- La principal tarea de los oráculos es proveer información del exterior (precios, condiciones, etc) a la Blockchain de Ethereum
- Veamos algunos ejemplos:
  - Entropía (recordemos que la Blockchain es determinística)
  - Información sobre precios para cálculos de cambios
  - Índices de riesgo
  - Información del clima

Oracles

- Tienen tres funciones principales:
  - Obtener datos desde una fuente “off-chain”
  - Transferir la información “on-chain” a través de una TX firmada
  - Poner la información en el almacenamiento de un Smart Contract
- Los oráculos se dividen en tres categorías:
  - Pedido-Respuesta
  - Publicación-Subscripción
  - Lectura inmediata

Oracles - immediate read

- Son aquellos que proveen información para ser consumida “en el momento”
  - ¿Esta persona es mayor de 18 años?
  - ¿Qué temperatura hace en Buenos Aires?
- Almacenan los resultados en el almacenamiento del contrato
- Cualquier cliente de la Blockchain puede consultarla

Oracles - Publicación-Subscripción

- Similar a los antiguos “Feeds RSS”
- En este caso el oráculo es “actualizado” por otro contrato “on-chain” o actualizado por un servicio “off-chain”
- Quién desea recibir la información se suscribe y cuando hay actualizaciones las recibe
- Ejemplos
  - “Feeds” de precio
  - Información del clima
  - otras

Oracles - Pedido/Respuesta

- Son los más complejos
- Debido a que el volumen de información es muy grande, se espera que los clientes consulten por datos específicos
- Normalmente funcionan de la siguiente manera:
  - Una EOA ejecuta una TX asociada al Oráculo
  - Una vez validado el pedido y procesado el oráculo genera un “evento” aviando que los resultados están listos
  - El proceso que requería la información lee el evento o el almacenamiento del Oráculo

Aplicaciones descentralizadas (DApps)

DApps

- Una DApp es una aplicación (casi) completamente descentralizada
  - Backend
  - Frontend
  - Storage
  - DNS
- Entre las ventajas de las DApps podemos resaltar
  - Resistencia a la censura
  - Transparencia
  - Resiliencia

DApps - Backend

- En las DApps los smart contracts funcionan de backend y contienen la lógica de negocio y el almacenamiento
- El “backend” puede estar constituido por uno o más smart contracts
- Dado que los contratos tienen un límite en su tamaño, muchas veces aplicaciones complejas necesitan de varios contratos para toda la funcionalidad que implementan

DAPPs - Frontend

- Normalmente son aplicaciones web desarrolladas en JavaScript
- Esto permite a desarrolladores web tradicionales utilizar herramientas comunes para desarrollarlas (JS, Node, CSS, etc.)
- También pueden ser aplicaciones móviles

DAPPs - Almacenamiento

- Pueden utilizar el sistema de archivos llamado IPFS (Interplanetary File System)
- Es un sistema de almacenamiento distribuido
- Pensado para reemplazar a HTTP

DAPPs - Ejemplos

Ejemplo práctico, revisando algunas DAPPs

Introducción a blockchain y billeteras virtuales - Módulo 4

Docente: Nahuel Sánchez

*Este documento fue realizado en concepto de capacitación en Formación Profesional y dictada para el **Sindicato CePETel** a contar del mes de mayo del año 2023.*

Introducción a “Blockchain”

—

Módulo 4: Otros aspectos

Introducción al módulo 4

- Exchanges
- DeFi
- DAOs
- Los Tokens “más conocidos”
- “Shitcoins” y la explosión de ICOs
- NFTs
- Otras Blockchains

Exchanges - Introducción

- Plataformas (centralizadas o descentralizadas) que permiten comprar, vender o intercambiar crypto-monedas o dinero fiat
- También se utilizan para realizar trading
- Se denomina dinero fiat al dinero tradicional (Dolares, Pesos, Yuanes, etc.)
- Pueden pensarse como un puente entre el “mundo real” y “crypto”

Exchanges - Centralizados

- Son los más simples de utilizar
- Normalmente están regulados
  - Obedecen normas y reglamentaciones gubernamentales *
  - En su mayoría se requiere realizar KYC
- Son altamente centralizados
- No siempre son seguros/confiables
  - FTX.com
- Tienen poder sobre todos los activos que manejan
  - “Not your keys, not your crypto”
- En su mayoría permiten operaciones P2P (no confundir con redes P2P)

Exchanges - Centralizados

- Algunos ejemplos *
  - Binance (binance.com)
  - KuCoin (kucoin.com)
- Binance
  - El exchange número uno a nivel mundial
  - Requiere KYC para cualquier tipo de operación
  - Soporte para intercambio P2P
  - Gran utilización en Argentina (altos niveles de volumen)
  - Videos explicativos en español (Link)
- KuCoin
  - No requiere KYC para operaciones diarias hasta 1 BTC
  - Interfaz poco amigable

Exchanges - Descentralizados

- Son DAPPs creadas por empresas o grupos de personas
- Normalmente son un poco más complejos de utilizar que su variante centralizada
- Permiten el intercambio de cryptomonedas entre los usuarios de la plataforma
- No están controlados por ninguna entidad/empresa
- Normalmente no son regulados
- No permiten cambiar dinero fiat por dinero “crypto”
- No requieren KYC
- Se rigen por los contratos inteligentes

Exchanges - Descentralizados

- Algunos ejemplos *
  - Uniswap (app.uniswap.com)
  - Pancake Swap (pancakeswap.finance)
- Uniswap
  - Uno de los primeros DEX (Decentralized EXchange)
  - Interfaz intuitiva
  - Alta liquidez
  - Permite realizar “swaps” y otro tipo de operaciones

Exchanges - Ejemplos prácticos

Revisando Binance y Uniswap

DeFi: Breve intro

DeFi

El sistema financiero tradicional

- Tiene décadas de avances tecnológicos
- Una gran evolución en 1970 con los ATMs
- El siguiente gran salto fue con Internet
- Paypal, Robinhood, TransferWise (Fintechs)
- Altamente centralizado y controlado por unas (pocas) entidades
- Construido sobre tecnología obsoleta en forma de silos

DeFi

Finanzas Descentralizadas

- Se basa en los smart contracts y toda la tecnología y seguridad que vimos hasta ahora
- Posee las mismas cualidades que la tecnología Blockchain
- Provee alternativas para la mayoría de operaciones e instrumentos tradicionales (TradFi)
- Existen todavía muchos desafíos por sortear
    - Escalabilidad
    - Confianza / Seguridad
    - Facilidad de uso
    - Adopción masiva

DeFi

DAOs

DAOs - Introducción

- Decentralized Autonomous Organization (Organización Autónoma Descentralizada)
- Modelo de gestión común en el ecosistema relacionado a tecnología Blockchain
- Bitcoin fue considerada la primera DAO ya que:
  - Cuenta con un conjunto de reglas programadas
  - Funciona de forma autónoma
  - Se coordina a través de un protocolo de consenso
- Ethereum y los smart contracts hicieron posible la “masificación” de las DAOs

DAOs

- Conjunto de reglas programadas (Contrato Inteligente)
- Financiación
  - Tokens
  - Derecho a voto
- Una vez “Deployadas” se convierten en organizaciones autónomas e independientes regidas por las reglas previamente determinadas
  - Contratos inteligentes de código abierto
  - Normalmente todo se somete a votación a través de propuestas

DAOs

- La primera DAO se llamaba simplemente “La DAO”, fue creada por una empresa alemana
- La idea de esta empresa era un “Airbnb” descentralizado
- Fue la operación de “crowdfunding” más exitosa de la historia (150M)
- Sufrió el primero “hackeo” de la historia y tuvo fuertes repercusiones
    - El código como ley
    - La inmutabilidad del código en la Blockchain
    - Ethereum “hard fork”

DAOs

Ejemplo práctico, revisando Compound

Los “Tokens” más conocidos

Tokens

- Cómo vimos anteriormente un token es la representación de “algo” en la Blockchain
- Normalmente será dinero, pero también hay tokens representando:
  - Acciones (ejemplo mAAPL, mirrored Apple)
  - Metales preciosos
  - Granos (Agrotoken)
  - Arte (NFTs)
- Existen dos grandes estándares
  - ERC-20 (Fungible)
  - ERC-721 (No Fungibles)

Tokens

- En el caso de las “criptomonedas” existen distintas categorías
  - Estables
  - No Estables
  - Altcoins (shitcoins)
  - Monedas “nativas”
- Se distingue a veces entre “moneda nativa” y “token”
  - Ejemplos de monedas nativas: Bitcoin en la red de Bitcoin, Ether en Ethereum
- Altcoins son aquellas criptomonedas creadas sobre un protocolo normalmente se llama de esta forma aquellas con poco “market-cap”

Tokens - Stable coins

- Tratar de mantener paridad con algún activo subyacente
  - Típicamente el Dólar
- Existen de distinto tipo
  - Colateralizadas con dinero fiat
  - Colateralizadas con criptoactivos
  - Algorítmicas
- Colateralizadas con dinero Fiat
  - Son las más comunes
  - Teóricamente mantienen reservas en efectivo o similares en entidades financieras

Tokens - Stable coins

- Colateralizadas con criptoactivos
  - Utilizan Smart Contracts en formato de bóvedas
  - Normalmente deben ser “sobrecolateralizadas” debido a la volatilidad de las criptomonedas
  - Ejemplos:
    - DOC, colaterizada con Bitcoin
- Algorítimicas
  - Funcionan a través de un algoritmo que mantiene la paridad
  - No mantienen una colateralización completa con otro activo
  - Responden a las leyes de oferta y demanda. Cuando el precio comienza a subir se emiten nuevas monedas para bajarlo y viceversa
  - Todavía requieren mucha investigación (Colapso de UST)

Tokens - Stable coins

- Ejemplos de Stablecoins
  - USDT
  - USDC
  - DAI
- USDT
  - Creada y mantenida por la empresa Theter
  - Mantiene la paridad con el dólar
  - Sospechada de tener muchos problemas y falta de colateralización real
- USDC
  - Creada por la empresa Circle
  - USD Coin
  - Más transparente que USDT pero menos utilizada
  - Paridad 1:1 con el dólar

Tokens - Stable coins

- DAI
  - Creada por la DAO MakerDAO
  - Colateralizada con criptomonedas (Ether, USDC, y otras)
  - Completamente descentralizada
  - Paridad 1:1 Dólar

“Shitcoins” y la explosión de ICOs

“Shitcoins” y la explosión de ICOs

- ICO es el acrónimo de Initial Coin Offering (similar a IPO)
- La burbuja de las “shitcoins” o “altcoins” comenzó a gestarse en 2017
- Se creó como producto del mercado alcista de crypto (principalmente Bitcoin)
- La falta de regulación y facilidad de creación contribuyeron fuertemente
- Las redes sociales contribuyeron a dar confianza a proyectos sin sustento técnico
  - Reddit
  - Telegram
  - Foros
- Se calcula que se invirtieron alrededor de 15.000 Millones durante 2017

“Shitcoins” y la explosión de ICOs

- Shitcoin define a proyectos que son copias de otros y no aportan ninguna innovación
- Su principal objetivo es obtener inversiones para luego desaparecer
- Si bien es posible “invertir” en este tipo de proyectos es muy riesgoso y normalmente las ganancias no justifican el riesgo

“Shitcoins” y la explosión de ICOs

NFTs

NFTs

- Non Fungible Token (Token No Fungible)
- Representan valores “únicos”, el ejemplo más común son “obras de arte”
  - Propiedades
  - Cine
- Permiten probar que un usuario es dueño de un determinado bien
- Se basan en el estándar EIP-721
- Han sido producto de gran controversia y actividades fraudulentas
  - “Wash Trading” (Volumen de compra/venta falso)
- Se “tradean” en plataformas de la misma manera que las criptomonedas
- Existen colecciones con gran popularidad
  - BAYC

NFTs

Ejemplo práctico, explorando un NFT

Otras Blockchains

Otras Blockchains

- Además de Bitcoin y Ethereum existen muchas otras Blockchains
- Muchas de ellas son “copias” de Ethereum o están basadas en tecnologías muy similares
- Entre ellas podemos citar:
  - BNB Chain (Previamente Binance Smart Chain)
  - Polygon (Previamente MATIC)
  - Tron Network
- Cada una de ellas tiene propiedades y objetivos distintos

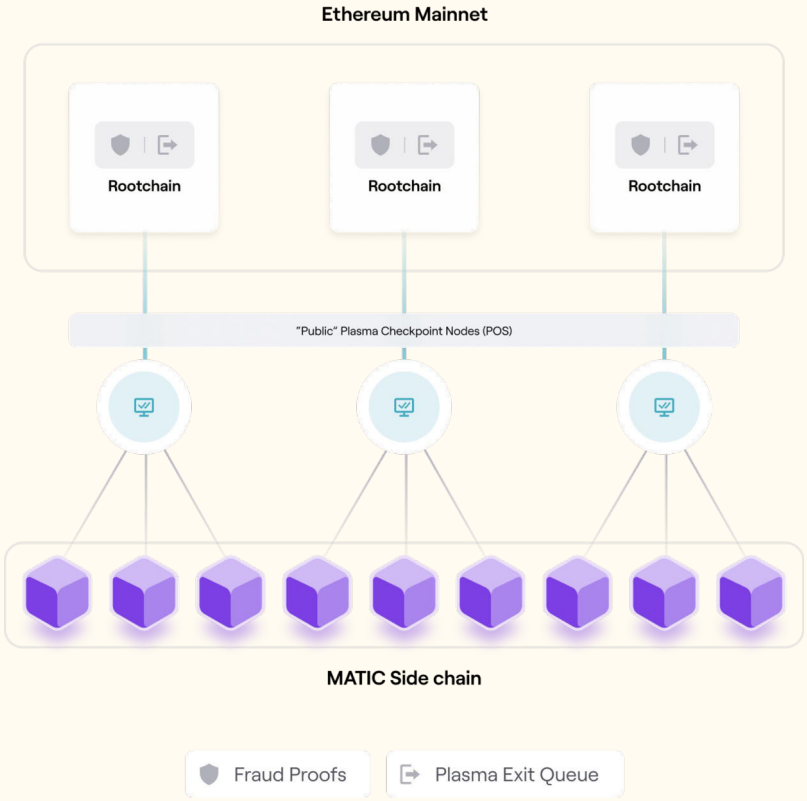
BNB - Binance Smart Chain

- Compatible con Ethereum, creada en 2020
- Su principal fin fue “abaratar” el costo de las TXs
- Está fuertemente integrada al exchange
- Su “token” nativo se llama BNB
- Utiliza una mezcla de PoS y PoA como método de consenso
- Cuenta con 21 nodos validadores
- Los costes promedio de una TX son de 0.35 USD vs Ethereum con 5.2 USD

Polygon

- Solución de Escalabilidad que funciona “sobre” Ethereum
- Es una red “aparte” con sus propias reglas que “de vez en cuando” graba su estado en la Blockchain de Ethereum
- Polygon - PoS aumenta considerablemente la cantidad de TXs posibles de forma concurrente
- Es compatible con Ethereum y los Smart Contracts

Polygon


Ethereum Mainnet
Rootchain
Rootchain
Rootchain
"Public" Plasma Checkpoint Nodes (POS)
MATIC Side chain
Fraud Proofs
Plasma Exit Queue

TRON

- Red alternativa a Ethereum (Compatible)
- Soporta Smart Contracts
- Se diferencia de Ethereum en:
  - Algoritmo de consenso (DPOS)
  - EVM (Compatible con Ethereum pero con algunas diferencias)
- Transacciones muy baratas

Otras Blockchains

Ejemplo práctico, utilizando el explorador de bloques de redes alternativas

Introducción a blockchain y billeteras virtuales - Módulo 5

Docente: Nahuel Sánchez

*Este documento fue realizado en concepto de capacitación en Formación Profesional y dictada para el **Sindicato CePETel** a contar del mes de mayo del año 2023.*

Introducción a “Blockchain”

—

Módulo 5: Billeteras virtuales

Introducción al módulo 5

- Introducción
- Tipos
- Beneficios / ¿Cómo gana dinero una Billetera Virtual?
- Arquitectura
- Usos y aplicaciones
- Implementación
- Aspectos de seguridad

Un poco de historia

- En 1997 Coca cola permite comprar latas mediante un SMS (Helsinki).
- Durante 1999 se lanza Paypal, si bien no es una billetera virtual, tuvo un gran impacto en el comercio on-line.
- En 2003 se lanza AliPay, hoy en día la plataforma de pago más grande del mundo.
- En 2011 Google lanza “Google Wallet”.
- En 2014 Apple lanza “Apple Pay”.
- “Android Pay” y “Samsung Pay” nacen en 2015.

¿Qué es una billetera virtual?

Son aplicaciones que almacenan uno o más métodos de pago en un dispositivo móvil. Algunas permiten almacenar otro tipo de información (Pasajes, registros médicos, pasaportes, etc).

Billetera Virtual, Billetera digital, Pagos móviles

- Billetera móvil / Billetera digital / Billetera virtual / Billetera electrónica

- Pagos móviles

Tipos de billeteras móviles

Podemos dividir las distintas billeteras móviles en dos grandes grupos:

- Aplicaciones que almacenan información de pago
- Aplicaciones de Proveedores de Servicios de Pago (PSPs)

Aplicaciones que almacenan información de pago

- Almacenan información que permite realizar pagos de forma segura
  - Tarjetas Débito/Crédito
  - Usuarios y contraseñas
  - Otros
- NO proveen ningún tipo de cuenta
- En pocas palabras, permiten utilizar tarjetas físicas de forma digital
  - Pagos con código QR
  - Pagos con tecnología NFC
- Se integran con servicios de pago (Google Wallet/Google Pay)
- Ejemplos
  - Google Wallet
  - Apple Pay
  - Samsung Wallet

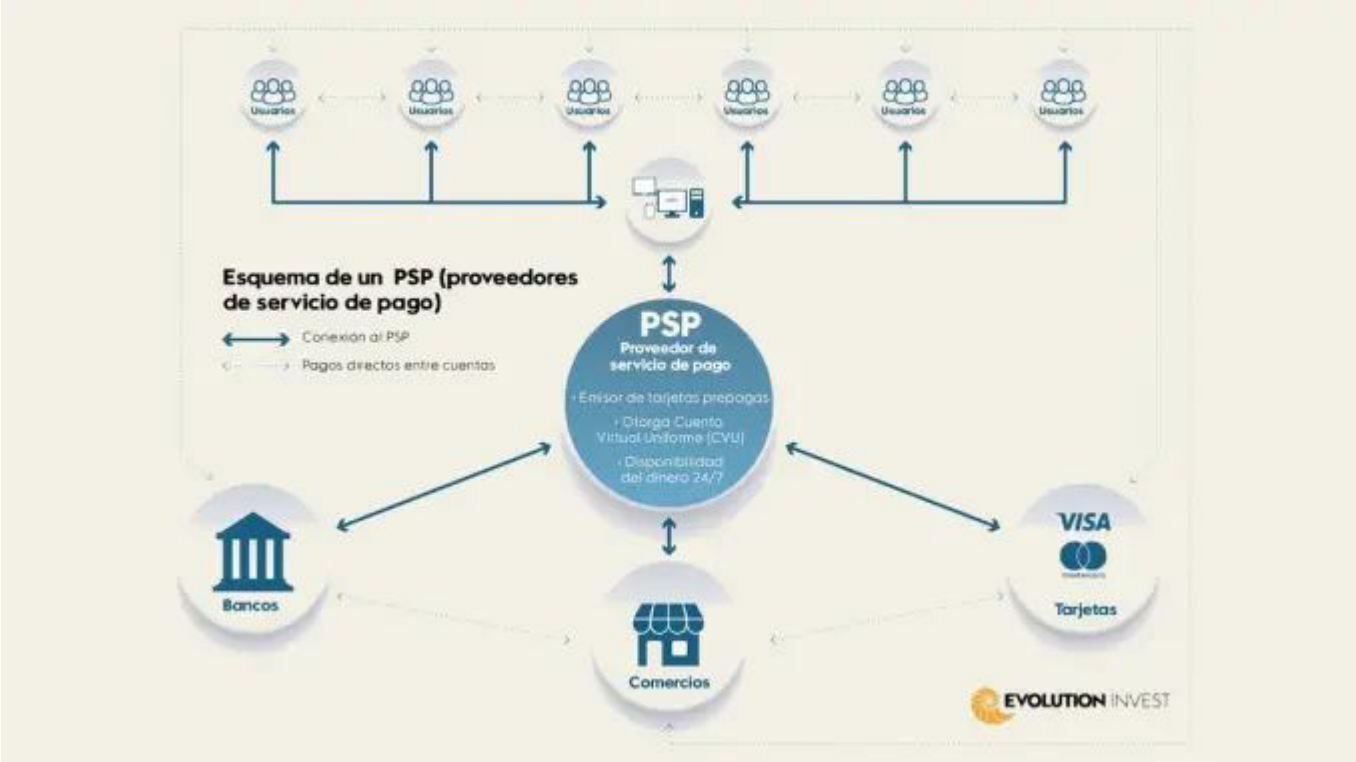
Aplicaciones de Proveedores de Servicio de Pago

- Permiten realizar pagos utilizando cuentas de pago
- Están reguladas por el BCRA (COMUNICACIÓN “A” 7495)
- Las empresas que las desarrollan no se consideran “Entidades Financieras”
- Las entidades financieras también pueden proveer este tipo de servicio
- La COMUNICACIÓN “A” 7462 las define como: “Billetera Digital”, “Billetera Virtual”, “Billetera Electrónica”
- Ejemplos: Mercado Pago, Personal Pay, Claro Pay, UALA, ...
- Un ejemplo interesante: MODO (Iniciador)

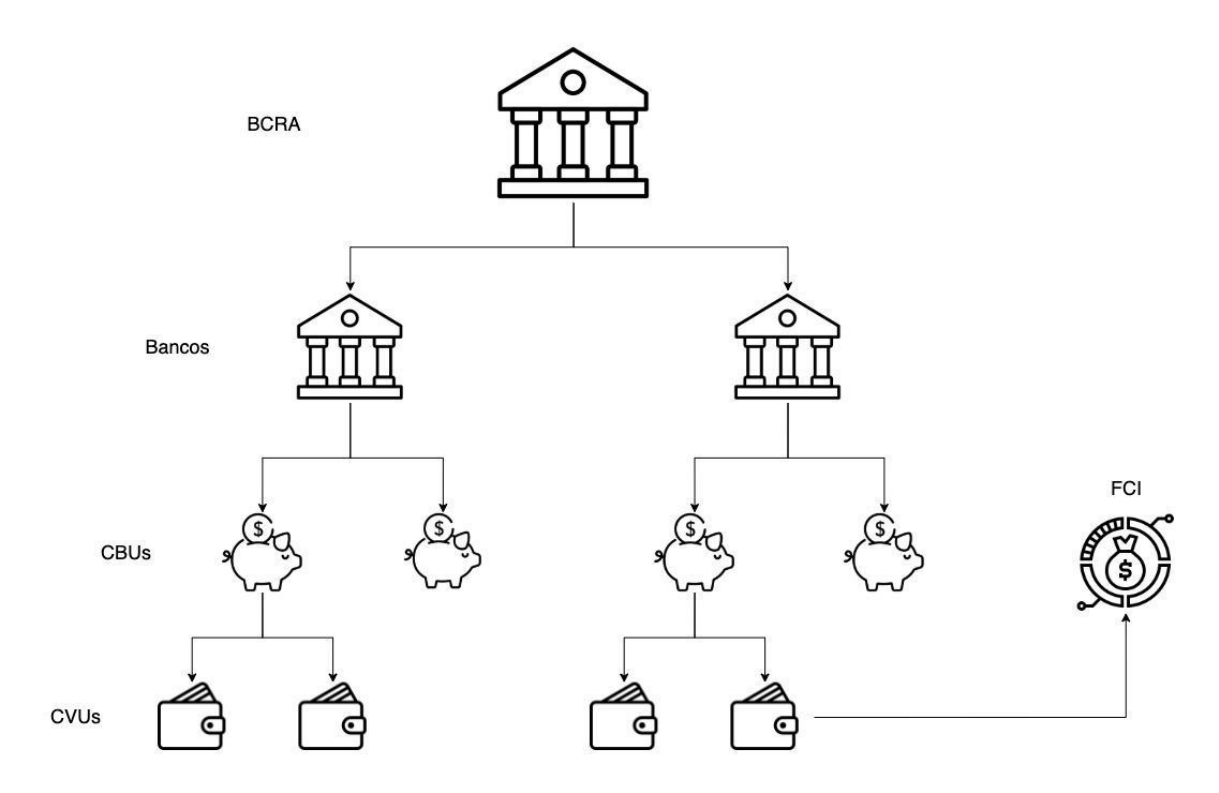
Aplicaciones de Proveedores de Servicio de Pago

- Estas aplicaciones ofrecen cuentas de pago identificadas por el CVU
- Permiten realizar pagos, recibir y enviar dinero, realizar inversiones, etc
- Dado que tenemos una “cuenta” podemos tener un saldo en la aplicación
- Algunas de estas aplicaciones nos permiten pedir tarjetas de débito prepagas

Arquitectura



Arquitectura



PSPs, Entidades financieras y Activos digitales

A partir de la COMUNICACIÓN “A” del 04/05/2023 los PSPs que ofrezcan cuentas de pago no podrán ofrecer a sus clientes la compra o venta de activos digitales.

Por ejemplo, Mercado Pago, en otros países de la región permite comprar y vender criptomonedas desde su aplicación

Ventajas de las Billeteras virtuales

- Facilidad de pago
- Aplicación multipropósito (Pagos, Créditos, Recargas, Promociones)
- Impulsan y facilitan el gasto
- Ingresos por Transacción
- Ingresos modelo “Marketplace” (Los vendedores o locales pagan comisiones)
- Modelo de “Billetera-as-a-Service”
- Venta de información de uso (compras, uso de promociones, etc)

Ventajas y usos más comunes

- Enviar y recibir dinero “al instante”
- Inversiones en FCIs (Money Markets)
- Descuentos y promociones
- Pago de servicios e impuestos
- Recargas de servicios
- Mayor seguridad contra fraudes *

Implementación de una billetera virtual

- Billetera “White-Label”
- Desarrollo a medida
- Integración con servicios (Google Pay, Apple Pay, otros)

Billetera “White-Label”

- Utilizan una plantilla a la cual le agregan la “marca”
- Normalmente proveen los mismos servicios a todos los clientes
- Permiten realizar customizaciones, normalmente con un costo adicional
- Son las más simples de utilizar
- El tiempo de desarrollo es menor que en otros casos
- Proveen un “SDK” (Software Development Kit)

Desarrollos a medida

- En estos casos las empresas contratan (o tienen) un equipo de desarrollo propio
- Las soluciones en este caso son específicas para las necesidades
- Requieren tiempos de desarrollo más largos
- Tienden a ser la elección de empresas grandes (MercadoPago, Personal Pay, otros)
- Desarrollan soluciones para Android y iOS

Integración con servicios

- En este caso el producto busca utilizar soporte del sistema operativo del teléfono (Android Wallet/Google wallet, Apple Wallet, Samsung Wallet)
- Cada sistema operativo provee su SDK
- No se limita a billeteras virtuales “Tradicionales”
- Puede utilizarse para membresías, pasajes, etc.


9:30
5G
State of California
Covid-19 Vaccination Card
Name
Jason Doe
Date of birth
1984-08-20
Manufacture
Pfizer
Dose
2021-04-13
Manufacture
Pfizer
Dose
2021-05-13
Only visibleto you
Details


9:41
Done
Fleet
GATE
62
SAN FRANCISCO
SFO
NEW YORK
LGA
SCHEDULED
2:40
FLIGHT
AP 2214
SEAT
33A
GROUP
B
PASSENGER
LIZ CHETELAT

Amenazas más comunes

- Phishing
- Ingeniería social
- Malware/Virus
- Configuración incorrecta del OS/Smartphone

Phishing

- Consiste en convencer al usuario a bajar/instalar código malicioso
  - Aplicaciones del Marketplace
  - Ingreso a sitios malignos
  - Otros
- Una de las amenazas más comunes y efectivas
- Dificil de combatir
  - Teléfono actualizado
  - Educación

Ingeniería social

- También requiere convencer a la víctima
- En este caso de compartir información sensible
  - Usuario / Contraseña
  - PIN
  - Otra
- Últimamente ganó popularidad
  - Estafas WA
  - Estafas de MercadoLibre / MercadoPago
- Requiere de educación y alerta para evitarlo

Malware / Virus

- Aplicaciones maliciosas que roban datos
- Normalmente se aprovechan de problemas de seguridad en el OS
- A medida que pasa el tiempo la probabilidad de infección decae
  - Son removidas de los mercados de aplicaciones
  - Los smartphones son actualizados
- Una vez instalados son difíciles de detectar
- Para evitarlos se requiere una configuración segura del teléfono y prácticas sanas

Configuraciones inseguras

- Sin duda de los principales problemas
- Se basan en las malas prácticas y/o “comodidad” de los usuarios
  - Contraseñas débiles
  - PIN fácil de adivinar
  - Patrones de bloqueo simples
- A medida que avanza la tecnología se vuelven menos comunes
  - Configuraciones seguras “por defecto”
  - Sistemas Operativos más seguros
  - Actualizaciones

Consideraciones de Seguridad en el desarrollo

- Autenticación y autorización
- Transmisión de la información
- Almacenamiento
- Resistencia a “ingeniería inversa”

Autenticación y autorización

- Es el primer paso, ¿Cómo ganamos acceso a la aplicación?
  - Usuario y Contraseña
  - PIN
  - Segundo factor de autenticación
    - SMS
    - Llaves por hardware (Hardware keys)
  - Biometría
  - Otros (Combinaciones)

Autenticación y autorización

- Usuario y contraseña + Un segundo factor de autenticación
  - Llave por hardware
  - Biometría
- Los SMS no se consideran un 2do método de autenticación seguro
  - Robo del teléfono
  - SIM swapping

Transmisión de la información

- La información debe transmitirse a través de un canal seguro
  - Cifrado TLS
- La información en texto plano puede ser interceptada y modifica (MiTM)
- La aplicación debe forzar el uso de configuraciones seguras


Profe
Sang
SSL
SECURE
CONNECTION
https://
http://

Almacenamiento Seguro

- La aplicación debe utilizar las APIs de OS para almacenar información de forma segura
- Los OS modernos incluyen funcionalidad para almacenar información de forma segura
- Android
  - Secure Storage
  - Secure key management
  - Almacenamiento “por aplicación
- iOS
  - Data protection API
  - Keychain

Resistencia a la “ingeniería inversa”

- El desarrollo en Android puede ser en:
  - Java
  - Kotlin
- En Apple / iOS
  - Swift
  - Objective-C
- Todas las aplicaciones son “susceptibles” de ser inspeccionadas
  - Entender cómo funcionan
  - Interoperabilidad
  - Fallas
- El proceso se denomina “Reverse engineering”

Resistencia a la “ingeniería inversa”

- En el mundo ideal, las aplicaciones no deberían confiar en la “seguridad por obscuridad”
- Sin embargo en la práctica...
  - Secretos almacenados
  - Uso de funcionalidad privilegiada
  - Otros
- Protección de la propiedad intelectual
- Existen herramientas que dificultan la tarea de ingeniería inversa
- El proceso se denomina “ofuscación”